

***How to Use
FileMaker Pro
Plug-ins***

Contents

Introduction.....	3
Installing Plug-ins	4
<i>Supported FileMaker Versions.....</i>	<i>4</i>
<i>Installing Plug-ins Locally.....</i>	<i>5</i>
Using Plug-ins.....	8
Checking for Plug-ins.....	13
Using Plug-ins with Bound Solutions.....	14
Auto Plug-in Update.....	15
<i>What is it?.....</i>	<i>15</i>
<i>Server Side Configuration</i>	<i>15</i>
<i>AutoUpdate Folder.....</i>	<i>17</i>
<i>Auto Update Plug-in</i>	<i>18</i>
<i>Using the Auto Update Feature.....</i>	<i>19</i>
<i>Automating the Process</i>	<i>21</i>
Conclusion.....	22
Credits	22
Contact Us.....	22

Introduction

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 understand how many developers approach plug-in development.

In this document we will cover how to install plug-ins onto your computer for use within FileMaker Pro. Next, we will cover the basics of using and interacting with plug-ins. We will then look at ways for your database to determine if a plug-in is installed. Following that, we will look at how to add plug-ins to bound solutions created with FileMaker Pro Developer. Finally, we will take a look at the Auto Plug-in Update feature introduced in the 5.5 version of FileMaker Server to learn how to easily distribute plug-ins to all your computers in multi-seat installations.

Installing Plug-ins

Supported FileMaker Versions

Before you can use a plug-in, you need to install it. Before we go into how to install a plug-in, let's look at which versions of FileMaker support plug-ins, so we know which versions of FileMaker we can install into.

Product	Capable of using plug-ins?	Special Notes
FileMaker Pro 3.0 and prior	No	
FileMaker Server 3.0	No	
FileMaker Pro 4.0/4.1	Yes	
FileMaker Pro Developer 4.0	No*	Can create bound solutions that use plug-ins
FileMaker Pro 5.0	Yes	
FileMaker Pro Unlimited 5.0	Yes	
FileMaker Pro Developer 5.0	No*	Can create bound solutions that use plug-ins
FileMaker Server 5.0	No	
FileMaker Pro 5.5	Yes	Can download plug-in updates from Server 5.5†
FileMaker Pro Unlimited 5.5	Yes	Can download plug-in updates from Server 5.5†
FileMaker Pro Developer 5.5	Yes	Can download plug-in updates from Server 5.5† Can create bound solutions that use plug-ins
FileMaker Pro Server 5.5	No	Can distribute plug-ins to 5.5 guests†

* FileMaker Pro Developer 4.0 and 5.0 do not actually contain a special "developer" version of FileMaker Pro, so there is nothing to use a plug-in with. However, they both can create bound solutions that can use plug-ins.

† See the [Auto Plug-in Update](#) section later in this document.

As you can see, not every version of FileMaker can use plug-ins. Most notably, the FileMaker Pro Server applications cannot use plug-ins. There are two common misconceptions about plug-ins in network environments that use FileMaker Server. The first is the thought that you can put a plug-in on the server machine somewhere and it would then be available to every FileMaker Pro guest computer that logs into the server. However, this will not work. The FileMaker Server application only hosts FileMaker Pro databases; there is no way that it could host plug-ins. If you think of plug-ins as small applications that FileMaker Pro talks to, then hopefully you will see that if you tried to share that same application across many computers, you would run into all sorts of file sharing and memory sharing problems.

The second misconception about plug-ins in network environments that use FileMaker Server is the thought that since the database uses the plug-in, the plug-in must be on the same machine as the database file. Which would mean that somehow whenever you run the script that uses the plug-in, the script is actually running on the FileMaker Server machine and therefore it must have the plug-in loaded. However, this is also not true. When you run a script in FileMaker, that script runs on your computer, regardless of where the actual database file is, and therefore talks to the plug-in you have installed on your computer.

The next thing to notice is the FileMaker Pro 4.0 Developer and FileMaker Pro 5.0 Developer versions. Unlike the name suggests, these do not actually come with a special "developer" version of FileMaker Pro, but are instead just a suite of tools and resources for the FileMaker Pro developer. Since there is not an actual version of FileMaker Pro, there is nothing to use a plug-in with. However, these FileMaker Pro Developer versions do come with a Binding Tool that allows you to bind your database solution into a runtime form that can be distributed without a full copy of FileMaker Pro, and these bound solutions can use plug-ins.

Finally, FileMaker, Inc. introduced a new Auto Plug-in Update feature in their 5.5 suite of applications. This allows you to place the plug-in files in a special folder on the machine running FileMaker Server, and then any FileMaker Pro guest machine that logs into the server can download those plug-ins to the local machine to use them. For more information about this new feature, see the [Auto Plug-in Update](#) section later in this document.

Installing Plug-ins Locally

To get started with plug-ins, you will want to install them into FileMaker Pro on your local machine. In general, plug-ins consist of a single file that you copy to a specific place on your hard drive. After obtaining the plug-in, close FileMaker Pro and then locate the folder on your hard drive that contains your FileMaker Pro application. For Macintosh, this is usually in the Applications folder; while on Windows, it is usually in the Program Files folder. Next to the FileMaker Pro application, you should find a folder named FileMaker Extensions (on Macintosh; See Figure 1) or System (on Windows; See Figure 2). Copy the plug-in file into this folder to install it into FileMaker Pro.



Figure 1. Installing on Macintosh



Figure 2. Installing on Windows

Some plug-in vendors provide Installer Applications that either make the installation process easier, or perform a more complex install. Always read the ReadMe file and/or Documentation that came with the plug-in for proper installation.

After installing the plug-in, open FileMaker Pro. On Windows and Mac OS 9, go to the Edit menu, select Preferences, and then Application. On Mac OS X, go to the FileMaker Pro menu, select Preferences, and then Application. You should now see the Application Preferences dialog. Change to the Plug-ins tab and you should see a list of all the plug-ins that are currently installed (See Figure 3). If you select one of the plug-ins, a short description will appear below and the Configure button will enable, allowing you to bring up the configuration dialog for the plug-in if it has a configuration dialog.

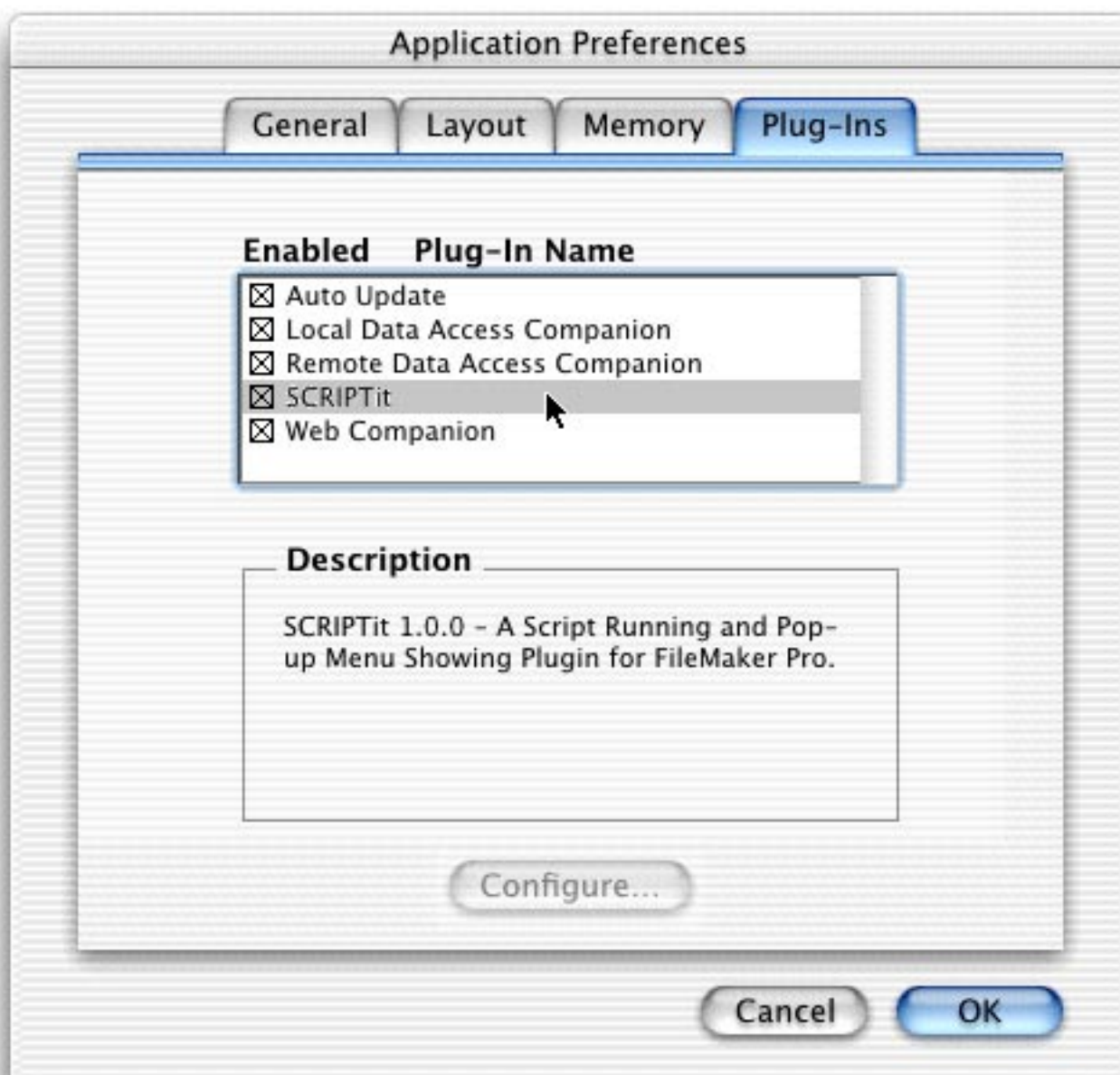


Figure 3. Application Preferences dialog

Using Plug-ins

Your database interacts with plug-ins using External Functions in FileMaker Pro Calculations. Calculations appear in many places within FileMaker Pro including Calculation Fields, the Set Field Script Step, the Paste Result Script Step, Auto Entered Calculations, Validation Calculations, and new with FileMaker Pro 5.5, Privilege Calculations. The most common method to interact with a plug-in is through the Set Field Script Step. It is easy to use; plus, you can see the results of the plug-in action in the field 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 located in the Help menu.

In order to try out the Set Field Script Step and plug-ins, we need a database to work with. So, open up FileMaker Pro and create a new database. When creating a new database, you are immediately presented with the Define Fields dialog. So, create a new Text field named Result. This field will hold the results from the plug-in when you use one of the plug-in's External Functions. Now click Done to close the Define Fields dialog.

If you are going to be working with any text that will contain single quotes, double quotes, or apostrophes (most likely you will be), then the first thing to do after creating a new database is to turn off Smart Quotes. When Smart Quotes are enabled, it turns all regular quotes ("" and '') to "smart" quotes ("” and “”). However, there is no way for a plug-in to determine if your database is using smart quotes or not. So, when text that contains smart quotes is transmitted to a plug-in, the plug-in will misinterpret the quotes as being some other character. So, save yourself some trouble and always turn off Smart Quotes after creating a new database that will use a plug-in. You do this by opening the Document Preferences Dialog. On Mac OS 9 and Windows, go to the Edit menu, select Preferences, and then select Document. On Mac OS X, go to the FileMaker Pro menu, select Preferences, and then select Document. Uncheck the Use smart quotes checkbox, and press the OK button.

To use a Set Field Script Step, open up the ScriptMaker™ by selecting it from the Script menu. Type a name for your script in the Script Name field at the bottom of the dialog and press the Create button. Scroll the list on the left until you see the Set Field Script Step and double-click it (See Figure 4). This will insert a Set Field Script Step into the script box on the right. At the bottom, you should now see two buttons labeled Specify Field... and Specify....

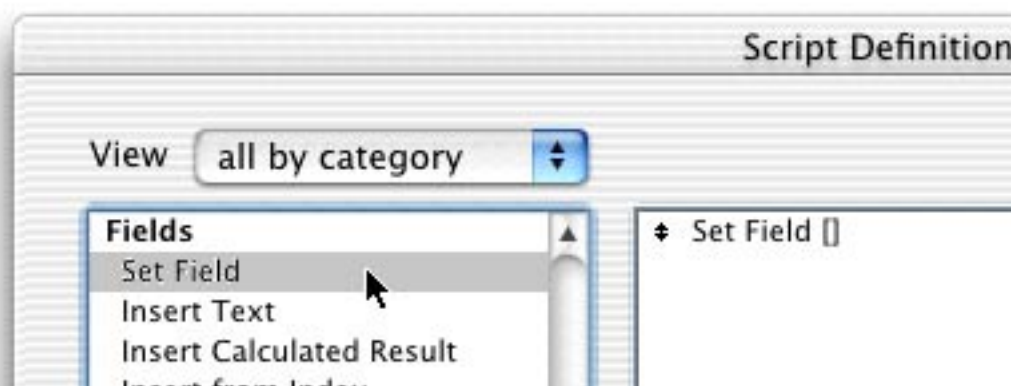


Figure 4. Inserting a Set Field Script Step

Press the Specify Field... button and select the Result field you created above (See Figure 5). While specifying a field is optional, you will want to specify a field the majority of the time. If you do not specify a field, then when FileMaker Pro performs this script, the Set Field Script Step will set the active field (the field with the cursor in it). If you did not specify a field and you did not click into a field before performing the script, then FileMaker Pro would ignore the Set Field Script Step as if it were not there. So, generally, you will always want to specify a field.

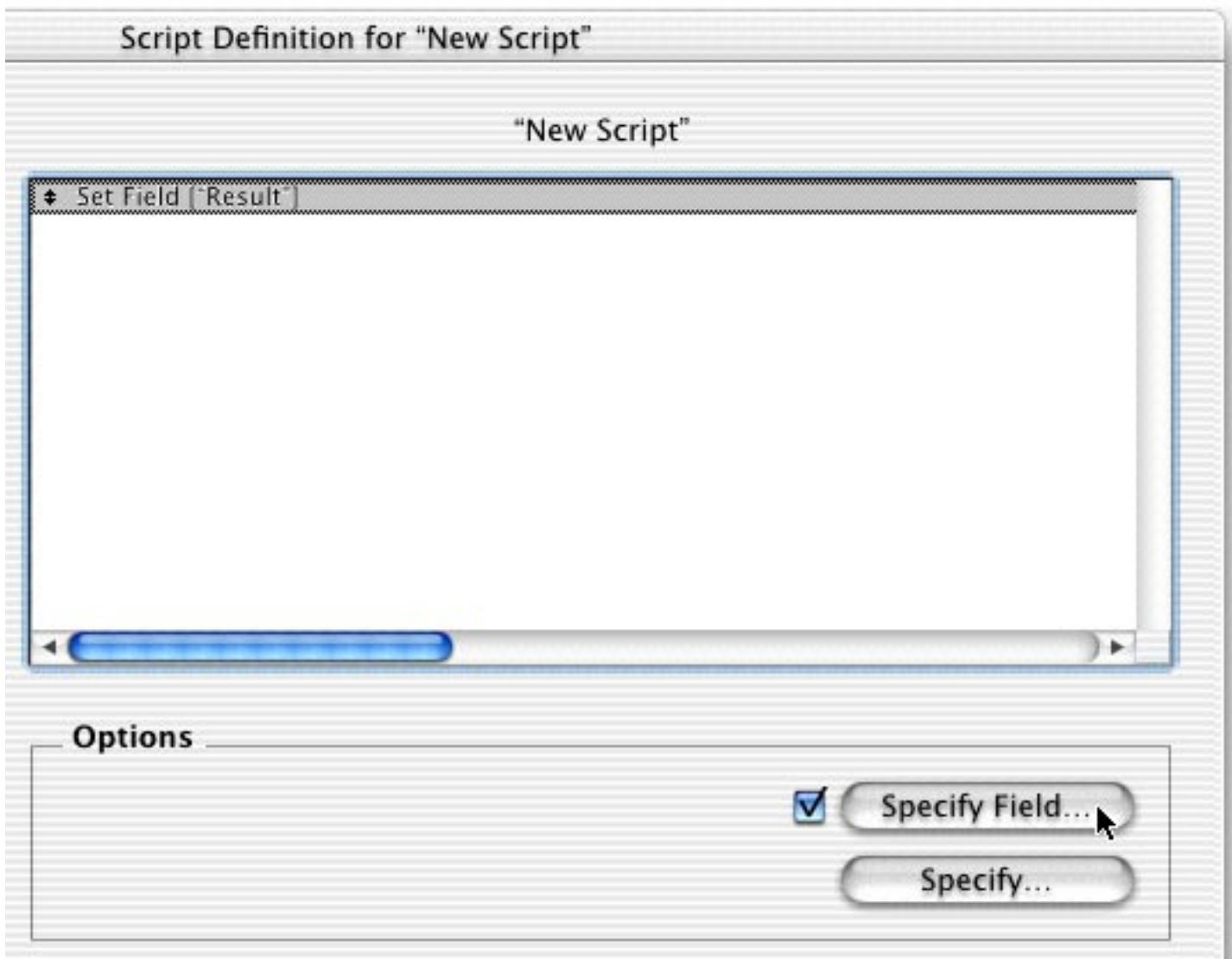


Figure 5. Using the Specify Field... button

Next, press the Specify... button to bring up the Specify Calculation dialog. Though the calculation dialog box is limited in space, FileMaker Pro has made it somewhat easier to deal with functions by creating groups of related functions and allowing you to view those groups individually. To view a list of all the External Functions of all the plug-ins currently installed, select External Functions from the View list at the top, right of the dialog (See Figure 6).

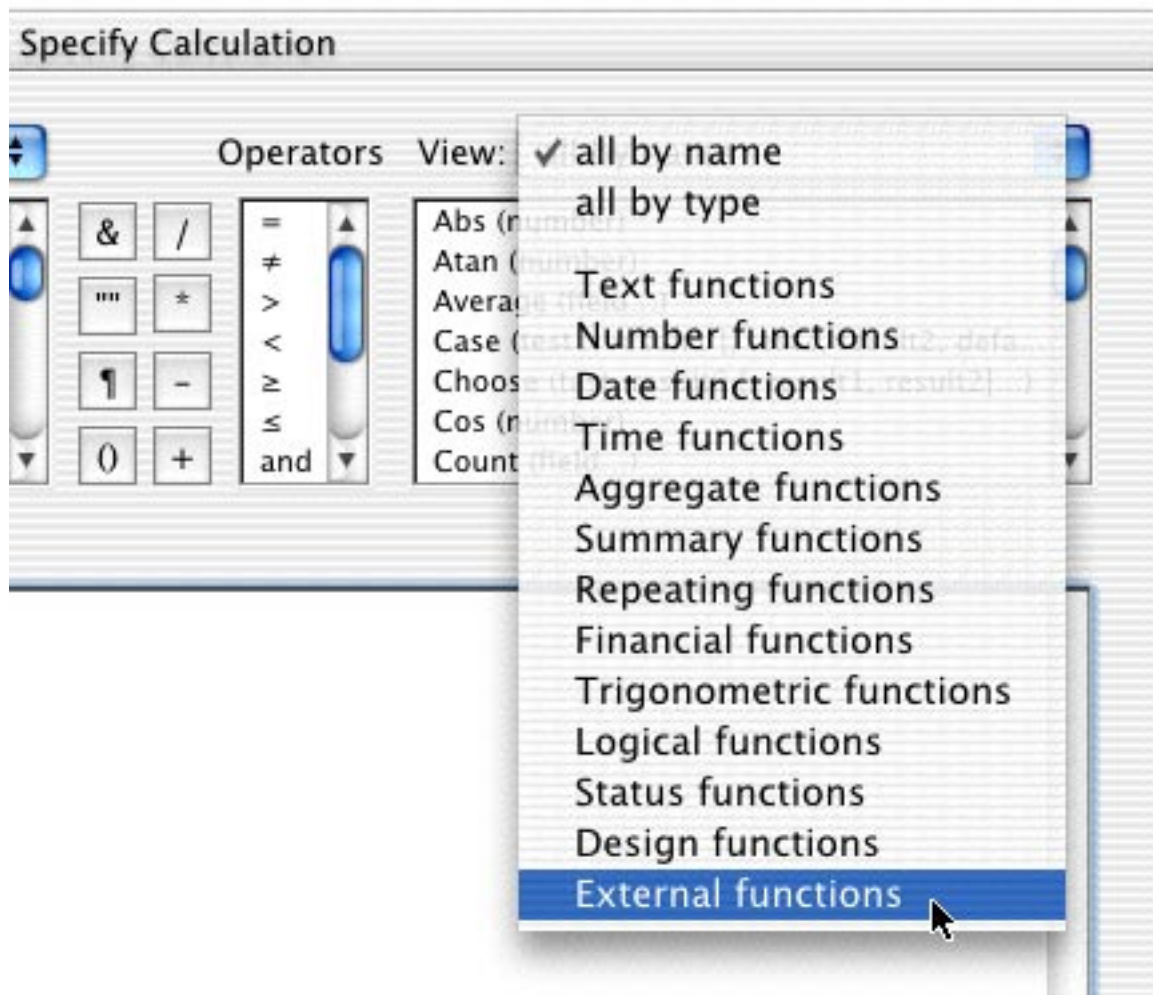


Figure 6. Selecting View External Functions

You should now see the External Functions for all the plug-ins that are currently installed and enabled. These functions are grouped by the plug-in name. For instance, all the Web Companion External Functions are listed under the Web Companion name.

Choosing to View External Functions can greatly increase the ease of your script writing because the External Functions that you need are right there at your finger tips. In fact, we highly recommend that you always insert External Functions into your calculations by double-clicking on them in this list. This will ensure that you have the correct spelling of the function name. The reason you want to make sure you have the correct spelling is because if you misspell a function name, or if you do not put capital letters where they should be, then FileMaker Pro will completely ignore the function as if it were not there, and will not report any errors to you.

The External Functions look like the following in your calculations:

External ("PluginNamePrefix-FunctionName", parameter)

For example take the Web Companion Web-Version function, which returns the version of Web Companion that you are using. When you double-click the Web-Version External Function in the External Function list, it inserts this into your calculation:

External ("Web-Version", parameter)

To actually use the function though, you need to give it a parameter. Since the Web-Version function does not need any special information, you can just set the parameter to the empty string ("") in order to use the function (See Figure 7):

External ("Web-Version", "")

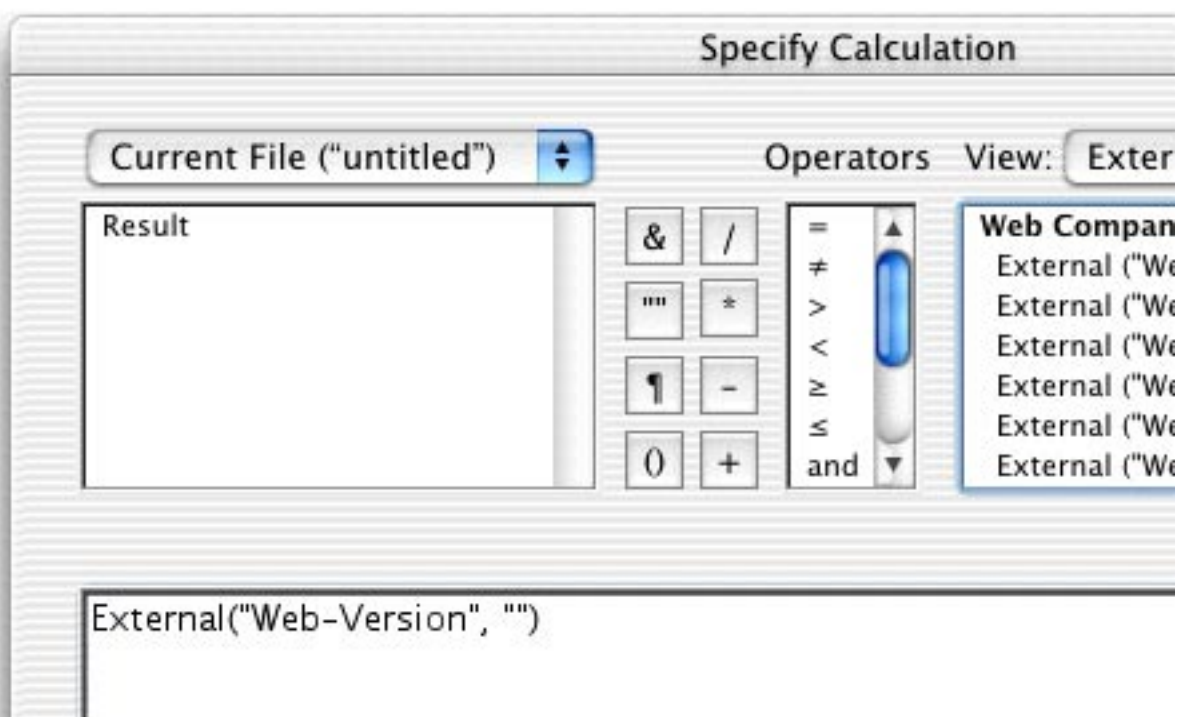


Figure 7. Using the Web-Version function

The parameter of the External Function is where you put the information for that function. For example, if you are using SMTPit (our email sending plug-in) and you need to set your From Email Address you would need to use the SMTP-FromAssign function. The information you need to give the SMTP-FromAssign function is your email address. One way is to use a literal value like "you@yourdomain.com":

External ("SMTP-FromAssign", "you@yourdomain.com")

You could also have a text field or a global field named FromField to hold your From Email Address value. If you did, then you can use that field like this:

External ("SMTP-FromAssign", FromField)

The difference between using a field to hold the value and putting the literal value directly into the calculation is the double-quotation marks around the parameter. FileMaker Pro will interpret anything that is not in double-quotes as a field in your database.

You can also mix literal values with fields in your database by concatenating them together. You do this by inserting an ampersand (&) between the values. For example, if you have a field named Contact Name and a field named Contact Email, you could use it in the SMTP-ToAssign function like so:

```
External ("SMTP-ToAssign", Contact Name & " <" & Contact Email & ">")
```

FileMaker Pro will concatenate the values in those two fields with the literal values " <" and ">". If your Contact Name field contained Frank Smith, and your Contact Email field contained frank@thesmiths.com, then FileMaker Pro would turn the above into:

```
External ("SMTP-ToAssign", "Frank Smith" & " <" & "frank@thesmiths.com" & ">")
```

FileMaker Pro will then concatenate it all together to form this:

```
External ("SMTP-ToAssign", "Frank Smith <frank@thesmiths.com>")
```

FileMaker Pro would then send all of that to the SMTPit which would use the parameter to set the To Email Address.

Checking for Plug-ins

Developing a database that uses a plug-in to make the database better is great, but only if the plug-in is actually installed. If your database is trying to use a plug-in that is not installed, FileMaker Pro does not warn you in any way, and will not report any errors. In the exact same way that FileMaker Pro will ignore misspelled function names, it will ignore any External Function calls to plug-ins that are not loaded. To alleviate this potential problem, your database needs to make sure that the plug-in is actually loaded before allowing the user to use the database.

Since there is not any sort of "IsPluginInstalled" Design or Status function in FileMaker Pro, you have to try calling one of the plug-in's functions and inspecting the results you get back. The easiest way to do this is to call the plug-in's Version function. While it is not mandatory for a plug-in developer to have a Version function in their plug-in, most developers include one. If the plug-in you are working with does not have a Version function, contact the plug-in developer and ask them the best way to determine if the plug-in is installed.

For all the plug-ins that do have a Version function, you can simply call this function and examine the results. If nothing is returned, then most likely the plug-in is not installed or not enabled. In this case, you would probably want to bring up an error message for the user asking them to make sure the plug-in is installed and enabled. If something is returned from the Version function, then you will know that the plug-in is installed and you can continue on with the database operation.

Let's look at an example script that checks for the existence of a plug-in:

```
Set Field [PluginResult, External("Scrp-Version", "")]
If [PluginResult = ""]
    Show Message [This database requires the SCRIPTit plug-in, but it does not
        appear to be installed. Please make sure the plug-in is installed and
        enabled before attempting to use this database.]
End If
```

This is a very simple script. First, the PluginResult field is set with the results from the Scrp-Version function. Next, it looks at the PluginResult field, and if there is nothing in it, then it brings up an error message using the Show Message script step. If your database does not work at all without the plug-in, or if it would pose a security risk to allow the database to run without a certain plug-in installed, then before the End If script step, you could insert a Close script step or even a Quit Application script step. That way, if the plug-in is not installed, then your database would immediately close before the user could use it.

The best time to check for installed plug-ins is when the database is first opened. You can define a "startup" script for your database that is run whenever someone opens your database. In FileMaker Pro, open up the Document Preferences dialog for your database. (On Mac OS 9 and Windows, open the Edit menu, select Preferences, and then Document. On Mac OS X, open the FileMaker Pro menu, select Preferences, and then Document.) On this dialog you will see a section labeled When Opening "mydatabase". In that section, you should see a checkbox labeled Perform Script with a drop down list next to it. You can check that checkbox and select the plug-in checking script that you created above.

Using Plug-ins with Bound Solutions

If you own one of the Developer's versions of FileMaker Pro, you can use the binding tool to bind your database solution into a runtime solution that does not require a full-featured copy of FileMaker Pro. However, the FileMaker Developer Tool application does not ask you if your database solution requires any plug-ins. If your solution does require plug-ins, then after binding the solution, you have to manually add in the plug-ins, which, fortunately, is a fairly simple thing to do.

After you have used the Developer Tool to bind your solution, you should have a folder containing your runtime solution and your solution's databases. In this folder, create a new folder. On Macintosh, name this folder "FileMaker Extensions". On Windows, name this folder "System". Copy the plug-in(s) into this folder and when your runtime solution is opened, it will have access to the plug-ins. If this seems familiar to installing plug-ins to work with a normal installation of FileMaker Pro, it should be. Basically, all you are doing is mimicking a regular installation of FileMaker Pro by creating the FileMaker Extensions or System folder, and then copying the plug-in(s) into that folder.

In general all of the FileMaker Pro Plug-ins on the market will work with a bound solution. The most common exception to this is all of FileMaker's "Companion" plug-ins, like the Web Companion. Basically, plug-ins have to be specifically designed to not work with bound solutions if the plug-in developer does not want someone distributing the plug-in with bound solutions. This is exactly what FileMaker, Inc. has done with their "Companion" plug-ins.

There may be other plug-in developers who have designed their plug-ins to not work with bound solutions, so if you try it and it does not work, then contact the plug-in developer. Also be aware that some plug-ins may have certain functions or features that may not work in the runtime solution environment. One of the more common problems on Windows is with any plug-in that attempts to use the ActiveX interface, which is not available in a runtime solution application. So, if you are trying to use a plug-in in a runtime solution and it does not work, or if parts of the plug-in do not work, contact the plug-in developer for any possible work-arounds they might have for you.

Auto Plug-in Update

What is it?

Realizing that it can be a hassle to distribute and update plug-ins in multi-seat scenarios, FileMaker, Inc. decided to build in the Auto Plug-in Update feature in its FileMaker 5.5 suite of applications. This excellent feature allows you to put plug-ins in a special folder on the FileMaker Server machine and then use the new Auto Update Plug-in on your FileMaker Pro client machines to automatically download and install the plug-in on the local machine.

This is actually quite easy to set up as well. We will start with the FileMaker Server side of the equation; looking at how to configure it, and then where to place the plug-ins. Next, we will examine the Auto Update plug-in on the FileMaker Pro Client side. Finally, we will look at some simple scripts to check plug-in versions and download updates.

Server Side Configuration

Configuring the Server is fairly simple, but it is slightly different for the four platforms that support FileMaker Server. On Mac OS X, open the FileMaker Server Config application to configure the server running in the background. Go to the FMServer Config menu and select Preferences. On the Preferences dialog that pops up, go to the Files tab. On that tab, you should find a checkbox titled Allow FileMaker Pro guests to download updates automatically (See Figure 8). Make sure this check box is checked. If it was not checked, then after pressing OK on the Preferences dialog, you will need to shut down and restart the FileMaker Server daemon running in the background for it to take effect.

On Mac OS 9, open the FileMaker Server application (if it is not already open). Go to the Edit menu and select Preferences. On the Preferences dialog that pops up, go to the Files tab. On that tab, you should find a checkbox titled Allow FileMaker Pro guests to download updates automatically. (This looks almost exactly like the OS X dialog, except for the Aqua interface; See Figure 8.) Make sure this checkbox is checked. If it was not checked, then after pressing OK on the Preferences dialog, you will need to shut down and restart the FileMaker Server application for it to take effect.

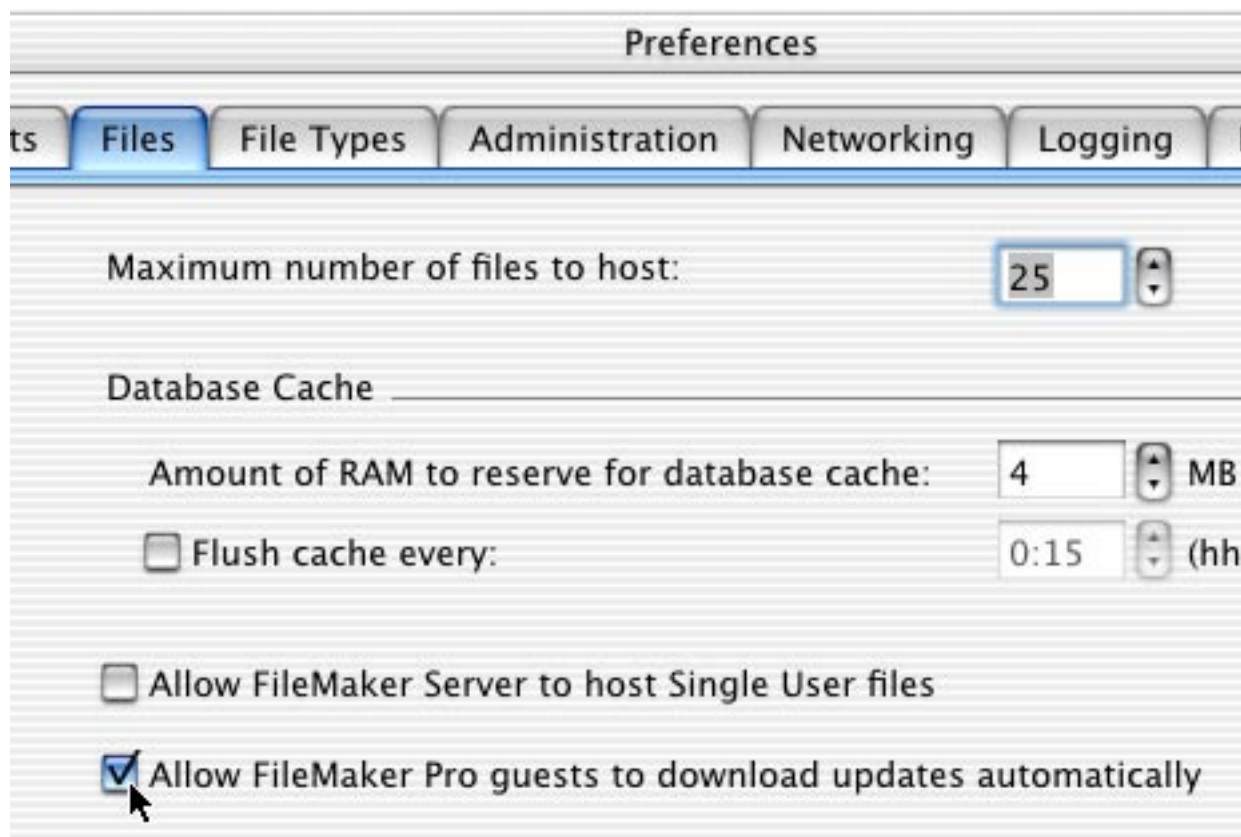


Figure 8. Turning on Auto Plug-in Update in OS X

On Windows, open the FileMaker Server Console. Right-click on FileMaker Server in the pane on the left, and choose Properties. On the Properties dialog that pops up, go to the Files tab. On that tab, you should find a checkbox titled Allow FileMaker Pro guests to download updates automatically (See Figure 9). Make sure this check box is checked. If it was not checked, then after you press OK on the Properties dialog, you will need to stop and restart the FileMaker Server Service for it to take effect.

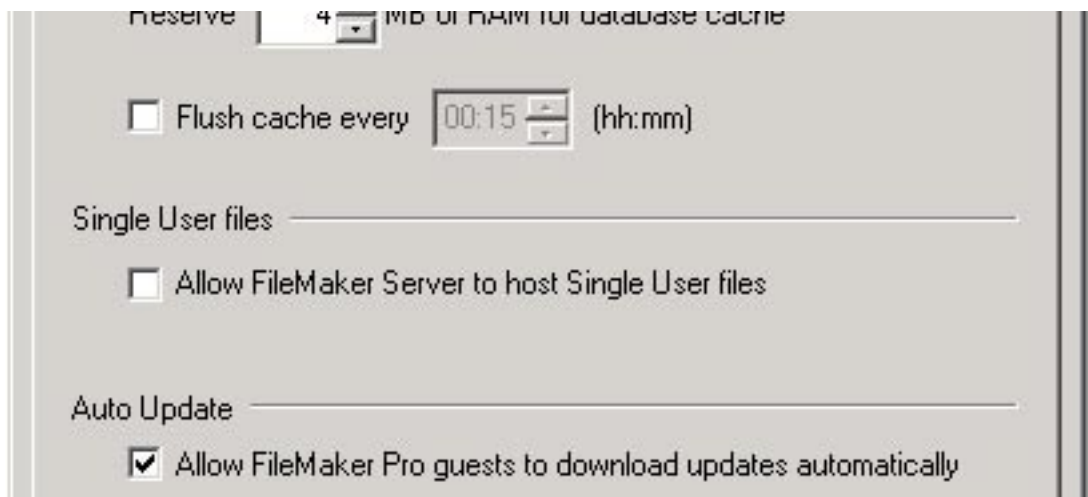


Figure 9. Turning on Auto Plug-in Update in Windows

On Linux, open up a console window and open up the `/etc/fmserver.conf` configuration file in your favorite linux editor. You will need super user privledges, so log in as root with `su`, or use `sudo` when opening your editor. Search through this configuration file until you see an option titled `UseAutoUpdate`. If it is not already, change this to read `UseAutoUpdate ON`. Save and close the configuration file. If FileMaker Server is not running, then start the daemon and it will load the configuration file. If FileMaker Server is running, then you can just reload the configuration file with `fmserverd reload` on the command line.

AutoUpdate Folder

After you have configured the server, you need to place the plug-ins in a special folder for FileMaker Server to find them to allow your guest computers to download them. This folder is named `AutoUpdate`, and you can have more than one depending on your needs. If your server hosts several sets of databases for several different solutions, and you have each of these sets in their own folder on the server, then each of these sets can have their own `AutoUpdate` folder. This allows each set of databases to only download the plug-ins that they need, or that they have access to.

When a FileMaker Pro guest attempts to download a plug-in, the first place the FileMaker Server application looks is in the `AutoUpdate` folder that is in the same folder as the database that the guest has open. If the server cannot find the requested plug-in there, then it will look in the default `AutoUpdate` folder that is in the same folder as the FileMaker Server application (OS 9, Win); the FileMaker Server Config application (OS X); or the `/var/fmserver` directory (Linux). For the examples in the rest of this document, we are going to use this default `AutoUpdate` folder.

If you look into that default `AutoUpdate` folder, you will probably see a few files dealing with the Server Administration plug-in. These are examples that you can look at to help figure out how the Auto Plug-in Update feature works. They are installed by default when you install the FileMaker Server application.

If you take a look at the files in that `AutoUpdate` folder, you will see that they come in pairs. For example, there is a file called `"FMPSADM.FMX"` and a file called `"FMPSADM.FMX.txt"`. These two files work together. The first is the actual plug-in file for Windows systems, which is why it has the `".FMX"` extension on the end. Windows Systems need that extension to properly identify that the file is a FileMaker Plug-in file. The `"FMPSADM.FMX.txt"` file is a simple one-line text file that will contain the version information for the `"FMPSADM.FMX"` file.

If you open up the `"FMPSADM.FMX.txt"` file in a text editor (like BBEdit on Mac, or Notepad on Windows, pico on Linux), you will see this:

FMPSADM 5.5

As you will find out later, it really does not matter what you put in here, just as long as you know what it is and can program your FileMaker Pro script to correctly handle what you do have in there.

Another thing that you will notice is that the Mac plug-in files have the extension `".bin"` on the end. This is because the Mac plug-in files have both a "Data Fork" and a "Resource Fork" that make up the file. That "Resource Fork" can get lost if transferred to a Windows computer without first "compressing" or "encoding" the file in some way like BinHex encoding, MacBinary encoding, Stuffing, or Zipping. In this case, all of the Mac plug-in files should be

MacBinary encoded before being placed in the AutoUpdate folder. You can use a program like Stuffit Deluxe to MacBinary encode the plug-in files, or if you do not own a program that can MacBinary encode the plug-in files, you can use the Mac version of the Auto Update plug-in to do it as described below.

Since the FileMaker Server application already installed these example files, we will just use them to learn how the Auto Update feature works. We are done with the server for now; let's look at the client machines.

Auto Update Plug-in

On your FileMaker Pro client machine, open up the Application Preferences dialog. (On Mac OS 9 and Windows, open the Edit menu, choose Preferences, and then choose Application. On Mac OS X, open the FileMaker Pro menu, choose Preferences, and then choose Application.) Go to the Plug-ins tab, and make sure that you see the Auto Update plug-in and that it has a check or X next to it. If it is not there, then make sure you are using version 5.5 or higher of FileMaker Pro. If you are using version 5.5 or higher, and it still is not there, then you may need to install it from the FileMaker Pro CD.

Let's look at the External functions provided to us in the Auto Update plug-in:

FMSAUC-Version - This function will tell you the Version of the Auto Update plug-in itself. You can use this function to make sure the Auto Update plug-in is installed as explained in the [Checking for Plug-ins](#) section earlier in this document.

FMSAUC-FindPlugin - This is one of the main functions you will use. This function takes the filename of the plug-in you are trying to update and will return whatever you put in that one-line text file on the server.

FMSAUC-UpdatePlugin - This function is the function that actually downloads and installs the plug-in on the local machine. It too takes the filename of the plug-in you are trying to update.

FMSAUC-SaveVersion - This is a simple function that you will probably never use. It simply creates one of those one-line text files based on the information you give it. It just saves it to your local FileMaker Pro application folder and that is it. You can do the same thing yourself with a text editor. It would have been useful had it actually saved the file on the server along with a copy of the plug-in, but it does not.

FMSAUC-SaveAsMacBin - This is another function that you will probably not use, unless you do not own a copy of Stuffit Deluxe. It is only available in the Mac version of the Auto Update plug-in, and it simply takes the plug-in file you specify and MacBinary encodes it.

Using the Auto Update Feature

Now that we know what functions are available, let's create a simple database to test this Auto Update feature out. After you know how it works, you will most likely want to implement it in the "opener" or "startup" script for your database. You can expand on the example script in the [Checking for Plug-ins](#) section above to include a check to see if there is a new version of the plug-in on the server to download. For now, though, let's just see how it works with a simple database.

Open up FileMaker Pro and create a new database. Create three new text fields and name them something like "Local Version", "Remote Version", and "Download Result". Click Done to close the Define Fields dialog and then open up the ScriptMaker™ from the Scripts menu. Create a new script named something like "Grab Versions". After pressing the Clear All button to clear out all the default Script Steps, insert a Set Field Script Step by double-clicking on Set Field in the list on the left. Use the Specify Field... button to specify the "Remote Version" field.

Next, press the Specify... button to bring up the calculation window. Change the View popup menu to External Functions and you should see the Auto Update External Functions in the list right below. Find and double-click the FMSAUC-FindPlugin function. This should insert that function into the calculation. Double-click the word parameter (to highlight it) and type in "FMPSADM.FMX" (with the quotes). You should now have this in the calculation box:

```
External ("FMSAUC-FindPlugin", "FMPSADM.FMX")
```

What this is going to do is ask the Auto Update plug-in to go find the version information for the plug-in named "FMPSADM.FMX". This is just the Windows version of this plug-in, so if you are doing this from Mac OS 9, then you will want to use "Server Administration" here instead:

```
External ("FMSAUC-FindPlugin", "Server Administration")
```

If you are doing this from Mac OS X, you will need this:

```
External ("FMSAUC-FindPlugin", "Server Administration X")
```

Note that with the Mac Plug-ins, you do not include the ".bin" extension on the end, even though that is the actual name of the file on the Server. In this case, the ".bin" extension is implied. (Yes, it is kind of confusing, but that is how it works.)

Click OK to close the Specify Calculation dialog, and then insert another new Set Field Script Step. This time, use the Specify Field... button to specify the "Local Version" field. Press the Specify... button to bring up the Specify Calculation dialog and then change the View drop down to External Functions just like before. Now, if you already have a copy of the Server Administration plug-in on this machine, then you can find it in the External Functions list. If you do, double-click the FMA-Version function to insert it into the calculation, and change the parameter to "". If you do not have the Server Administration plug-in installed, then just type it in. This is what it should look like:

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

This will simply set the "Local Version" field with the version of the Server Administration plug-in that you currently have on your machine. If you do not currently have any version on your machine, then the result of this will be a blank field. (Remember from the [Checking for Plug-ins](#) section above that FileMaker Pro ignores any External Functions for plug-ins that do not exist and will return nothing.)

Click OK to close the Specify Calculation dialog, and then click OK again to close the Script Definition dialog.

Next, create another new script called something like "Download Plugin". In this script, insert a Set Field Script Step (after clearing out all the default Script Steps), and use the Specify Field... button to specify the "Download Result" field. Bring up the calculation dialog (the Specify... button) and find the FMSAUC-UpdatePlugin function. This function looks exactly like the FMSAUC-FindPlugin, so make it look like this:

```
External ("FMSAUC-UpdatePlugin", "FMPSADM.FMX")
```

On Mac OS 9, it will be this:

```
External ("FMSAUC-UpdatePlugin", "Server Administration")
```

On Mac OS X:

```
External ("FMSAUC-UpdatePlugin", "Server Administration X")
```

This function will download and install the plug-in. Click OK twice, and Done once. You should now be back at the database. Before we can use this plug-in, we need to put it on the server, so you might need to change the Sharing to Multi-User unless you have your server set to Allow Single User files (See Figure 8). The reason we need to put this database on the server is because that is how the Auto Update plug-in knows where to look for the plug-ins. So, set it to Multi-User if you need to, close the database, and copy it to your server. Open it up on the server, come back to your FileMaker Pro client machine, and open the database up through the Hosts button or Open Remote menu item.

Now, try out your scripts. First, run the "Grab Versions" script and you should see those two "Version" field get filled in with information. The "Remote Version" field will contain what was in that one-line text file on the server. You can change that text file on the server, run the "Grab Versions" script again, and see that field change in your database. If you are getting a -1 in the field, then the Auto Update plug-in is having trouble finding the plug-in. Make sure you checked the Allow FileMaker Pro Guests to download updates automatically checkbox on the server, and that you restarted the Server application. Also, make sure you have the right file name in the script.

The "Local Version" field will contain the version information from the plug-in that is currently installed on your local machine. If it is blank, then you probably do not have the plug-in on your local machine, otherwise, it should report the version.

Next, try out the "Download Plugin" script. If it puts a 0 in the "Download Result" field, then it was successful. If it puts a -1 in the field, then it had some problem. Check that the filename is correct, and that it appears on the server. Once it reports a 0 in your "Download Result" field, that you know that it successfully downloaded and installed the new plug-in from the server.

Automating the Process

You have seen how easy it is to put the plug-in and its companion text file on the server, and how to use the Auto Update plug-in External Functions to retrieve the version information and the plug-in from the server. Obviously you will want to automate this, so all you need to do is figure out some way to compare the "Remote Version" and "Local Version" fields to see if the database should download the newer version or not.

With FileMaker's plug-ins you can just use the TextToNum calculation function in FileMaker to convert the version information into a number and compare the two. A script like this would work:

```
If [TextToNum(Remote Version) > TextToNum(Local Version)]
    Set Field [Download Result, External("FMSAUC-UpdatePlugin", "FMPSADM.FMX")]
End If
```

Unfortunately, a script like that is not going to work for all plug-ins because there is no standard method of defining a plug-in's version. For instance, all of our plug-ins return something like this from the Version functions by default:

FTPit v.1.1.7

Using TextToNum on that would return ".117", which would not work right in all situations with all different version numbers. For example, if we came out with "FTPit v.1.1.10", TextToNum would convert that to ".111", which is not greater than ".117" and it would never download that newer version.

Some plug-in developers have started to create special Version functions that return a version number that can be used with TextToNum to make this process easier. Check the documentation for the plug-in to see if they have a special Version function, or a special parameter for their Version function to return a more useful version number.

If the plug-in's version number is in an odd form, then you will need to come up with some combination of FileMaker's text calculation functions to determine if your solution should download the new plug-in. For instance, for the above FTPit version number, you could do this:

```
Set Field [Local Version, External("doFTP-Version", "")]
Set Field [Local Version, Middle(Result, Position(Result, ".", 1, 1)+1,
    Position(Result, ".", 1, 2)-Position(Result, ".", 1, 1)-1) & "." &
    Right("00" & Middle(Result, Position(Result, ".", 1, 2)+1, Position(Result, ".",
        1, 3)-Position(Result, ".", 1, 2)-1), 2) &
    Right("00" & Middle(Result, Position(Result, ".", 1, 3)+1, Length(Result)-
        Position(Result, ".", 1, 2)-1), 2)]
```

The "Local Version" field would then contain:

1.0107

Which you can use with the TextToNum function. On the server side, since you can put anything you want to in that one-line text file, all you would want to put in is "1.0110", and then when you use the TextToNum function on both the "Local Version" and "Remote Version" fields, you will know that you can download the newer version.

Conclusion

Once you understand all the interaction between FileMaker Pro and plug-ins as described above, you will be on your way to using plug-ins in no time. If you need more help understanding Scripts, Script Steps, and Calculations, consult the FileMaker Pro Help located in the Help menu. If you need more help with the Auto Update Plug-in feature, consult the FMS 5.5 Admin Guide.pdf file and the Auto Update Guide.pdf file in the Electronic Documentation folder on the FileMaker Server CD.

Credits

Content and screenshots by Jake Traynham (jake@comm-unity.net)
Direction and custom graphics by Jesse Traynham (jesse@comm-unity.net)

Contact Us

Have any comments or suggestions? Feel free to contact us in one of the following ways:

Email: info@comm-unity.net

Website: <http://www.cnsplug-ins.com/>

Address: Comm-Unity Networking Systems
8652 Camp Bowie West
Fort Worth, Texas 76116

Phone: 817-560-4226