



Version 3.0

The Emailing Plug-in for Filemaker 4.x / 5.x

Developed by Comm-Unity Networking Systems

Emailing with FileMaker Pro
has never been so easy
and flexible!



Comm-Unity Networking Systems

SMTPit Support Email:
smtpit@comm-unity.net

SMTPit Support Website:
<http://www.smtpit.com/>

Table of Contents

Introduction to SMTPit and FileMaker Pro Plug-ins	4
Example Databases Explained	5
External Function Reference and Contact Information	15



Introduction to SMTPit and FileMaker Pro Plug-ins

Introduction	4
Features	4
Installation and Configuration	5
Basics	6
Headers and Footers	7
Attachments	7
HTML Images	9
Email Headers	10
Authentication	11
After Send	11
Advanced	12
About	14
Registering	14
How to use FileMaker Pro Plug-ins	15

Introduction

Sending Email from within FileMaker Pro puts you in the driver's seat for complete communication. SMTPit allows you to send Email with several custom made features that just are not available in other products.

With SMTPit, mail merges are easy with our header and footer implementations. Imagine having a custom introduction to each of your customers aimed directly at them, calling them by name. From mailing lists to newsletters, SMTPit makes it simple, giving you the ability to have custom headers and footers for each Email.

SMTPit also works well with WebCompanion solutions. Imagine sending personalized thank you notes to every web visitor that fills out one of your web forms. You could even tailor those thank you notes with product details based on the web visitor's interests.

SMTPit also has the ability to send HTML only emails, or multipart/alternative Text & HTML emails for those email clients that do not understand HTML content. Plus, you can include inline HTML images to provide graphic logos or other images in your HTML emails.

Since SMTPit is so flexible, there are countless solutions. Imagination is the only limitation.

Features

SMTPit enables you to send Email directly from FileMaker Pro using To, From, Subject, CC, BCC, Email Headers, Attachments, and more. We have a few added features such as a headers and footers that are added to the body of email messages. We also have added functions to send HTML messages very easily. For your convenience, we have created a Configuration Dialog that allows you to set default settings of each feature.

Installation and Configuration

To install the plug-in, first make sure FileMaker Pro is closed. Next, unzip the SMTPit zip file on Windows, or unstuff the SMTPit stuffit file on Macintosh, and then double-click the SMTPit_Installer application. This will automatically place the SMTPit 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 SMTPit, 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 SMTPit 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 SMTPit plug-in.

Basics

Once the Configuration Dialog is open, click the **Basics** tab (See Figure 1) where you can set the following values. **SMTP Host** is where you enter the domain name or IP address of your mail server. **From** is where you put your email address. **Subject** is where you can put a default Subject for emails where you do not specify a Subject. **Priority** is where you set a default Priority.

SMTPit 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 from email address in your SMTPit related script, SMTPit will use the Default **From** address.

SMTPit waits for certain amount of time for your SMTP host to respond. By default, this is set to 30 seconds. However, you can adjust the setting to fit your needs by entering a different value in the **Connection Timeout**.

To be compliant with the SMTP protocol internet standard, SMTPit wraps all long lines down to a specified length before transferring your email. SMTPit adds the appropriate Email Headers for modern email clients to "unwrap" those lines back to their original length, so you will probably never know the difference. However, if you do need to change how many characters SMTPit leaves on a line, you can adjust the **SMTP Line Break Column** setting. The default setting is 80.

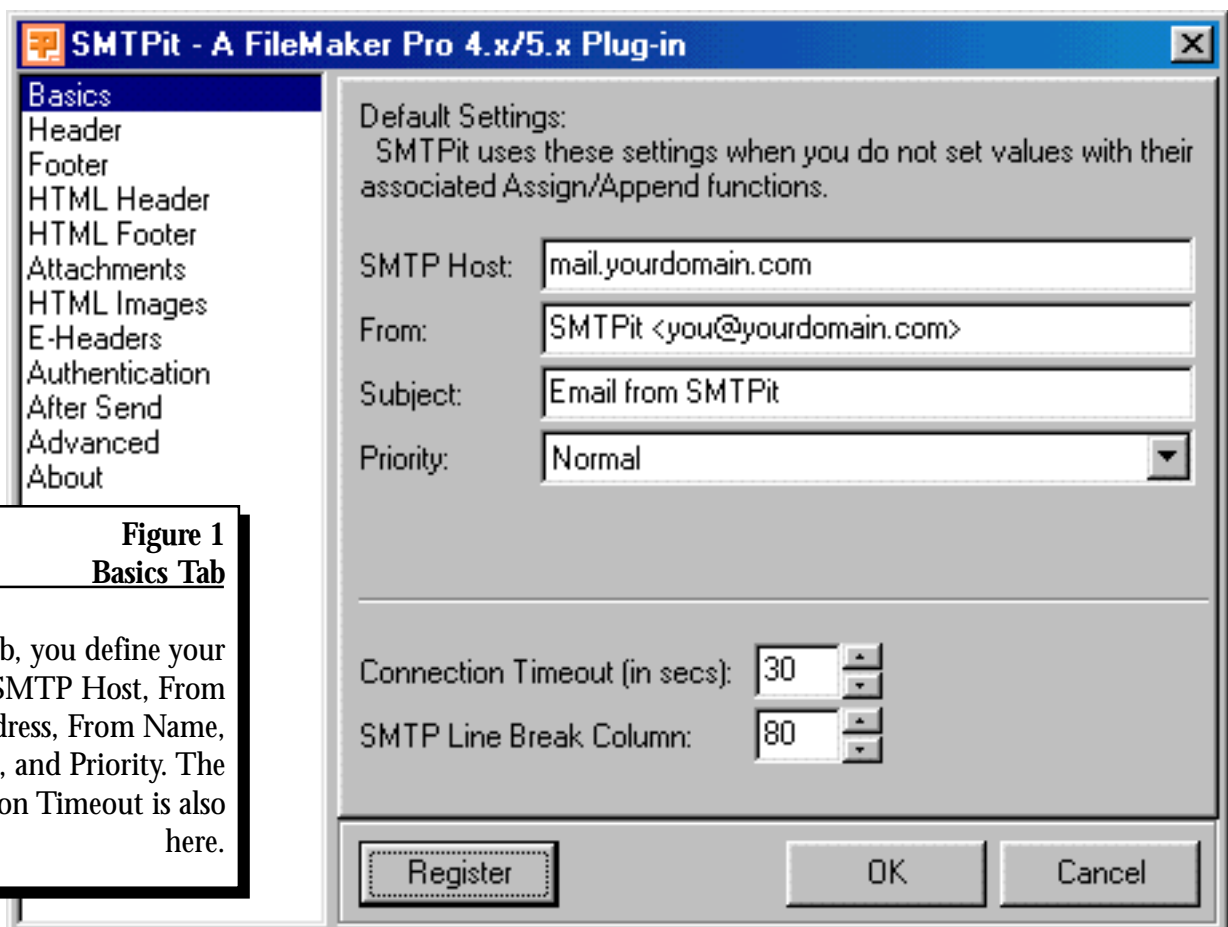


Figure 1
Basics Tab

On this tab, you define your Default SMTP Host, From Address, From Name, Subject, and Priority. The Connection Timeout is also here.

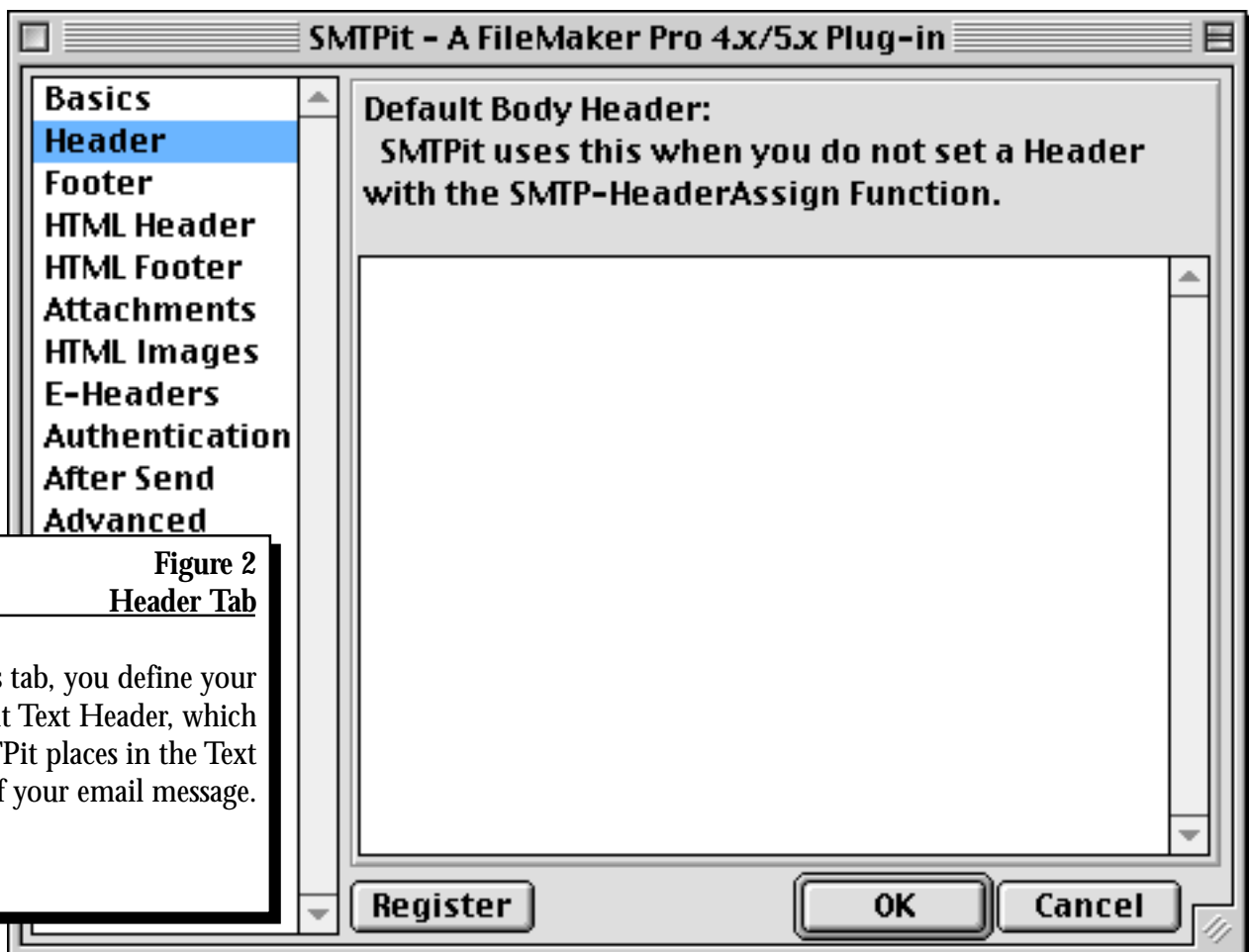
Headers and Footers

SMTPit has the unique ability to add headers and footers to the Plain Text Body and HTML Body of email messages. To enter a default header or footer simply click on one of the "Header" or "Footer" tabs (See Figure 2) in the Configuration Dialog. Then simply click in the edit field to enter a default header or footer. Like most SMTPit functions, you can replace the default values in your current scripts with values from fields in your database. You can also use the related append functions to add to the default headers and footers.

Because of this function, you can easily insert personal information into your email messages. You could have a default header and footer, and use the **SMTP-HeaderAppend** function to add a personal message or name to each email message. Another solution is to have a header or footer field in your database that is a calculation. With this scenario you could dynamically insert product information from your database into each email. Keep in mind that SMTPit also has the **SMTP-BodyAppend** and **SMTP-HTMLBodyAppend** functions that can be integrated into such a solution.

Attachments

SMTPit has the ability to send attachments either using the default attachments set in the



Configuration Dialog (See Figure 3) or via six functions: **SMTP-AttachAssign**, **SMTP-AttachAppend**, **SMTP-DlgAttachAssign**, **SMTP-DlgAttachAppend**, **SMTP-FileNameAcquire**, and **SMTP-PathToDBAcquire**. To set a default attachment, one that is always sent, click on the **Add** button on the **Attachments** tab. After you click the **Add** button, a standard open file dialog will pop up which will allow you to navigate to your desired attachment. Once you have found the file you want to attach, simply click the open button. To add additional default attachments, simply repeat the process above.

As mentioned above SMTPit has five functions related to attachments. The **SMTP-AttachAssign** and the **SMTP-AttachAppend** functions require a complete path to a valid file. To make sure you have the correct path, use the **SMTP-FileNameAcquire** function, which will present you with a standard open file dialog to locate the desired file. (Note: **SMTP-FileNameAcquire** only returns a valid path. It does not use the selected file as an attachment.) You can also use the new **SMTP-PathToDBAcquire** function to return a complete path to the folder on your hard drive that contains your database. This is useful if you have attachments in the same folder as your database. The **SMTP-DlgAttachAssign** and the **SMTP-DlgAttachAppend** present you with a standard open file dialog to choose an attachment just like the Configuration Dialog mentioned in the last paragraph. For more information on these functions, refer to the **SMTPit External Functions Reference** section of this documentation or refer to the SMTPit_Functions database that came with the SMTPit archive.

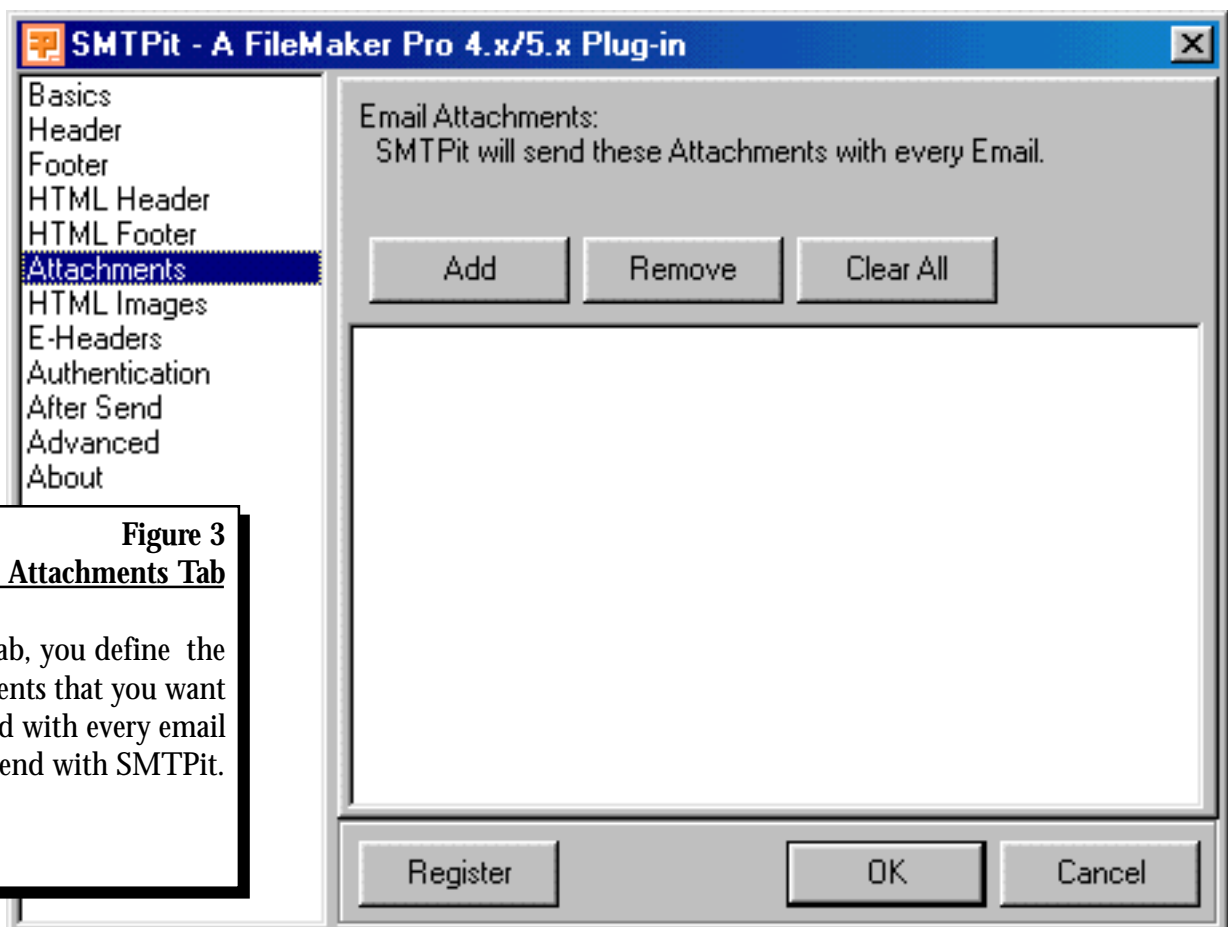


Figure 3
Attachments Tab

On this tab, you define the Attachments that you want to send with every email you send with SMTPit.

HTML Images

SMTPit 2.5 introduced the ability to send HTML emails with very little difference from sending Plain Text emails. With this new ability, comes the ability to send Inline HTML Images for display in the HTML emails. You can either assign default HTML Images in the Configuration Dialog (See Figure 4), or add HTML Images via four functions: **SMTP-HTMLImageAssign**, **SMTP-HTMLImageAppend**, **SMTP-DlgHTMLImageAssign**, and **SMTP-DlgHTMLImageAppend**. To set a default HTML Image, one that is always sent, click on the **Add** button on the **Attachments** tab. After you click the **Add** button, a standard open file dialog will pop up which will allow you to navigate to your desired HTML Image. Once you have found the image you want, simply click the open button. To add additional default HTML Images, simply repeat the process above.

As mentioned above, SMTPit 2.5 added four new functions related to HTML Images. The **SMTP-HTMLImageAssign** and the **SMTP-HTMLImageAppend** functions require a complete path to a valid file. Like the related Attachment functions, you can make sure you have the correct path by using the **SMTP-FileNameAcquire** function, which will present you with a standard open file dialog to locate the desired HTML Image. (Note: **SMTP-FileNameAcquire** only returns a valid path. It does not use the selected file as an HTML Image.) You can also use the new **SMTP-PathToDBAcquire** function to return the complete path to the folder on your hard drive that contains your database. This is useful if you have HTML Images in the same folder as your database.

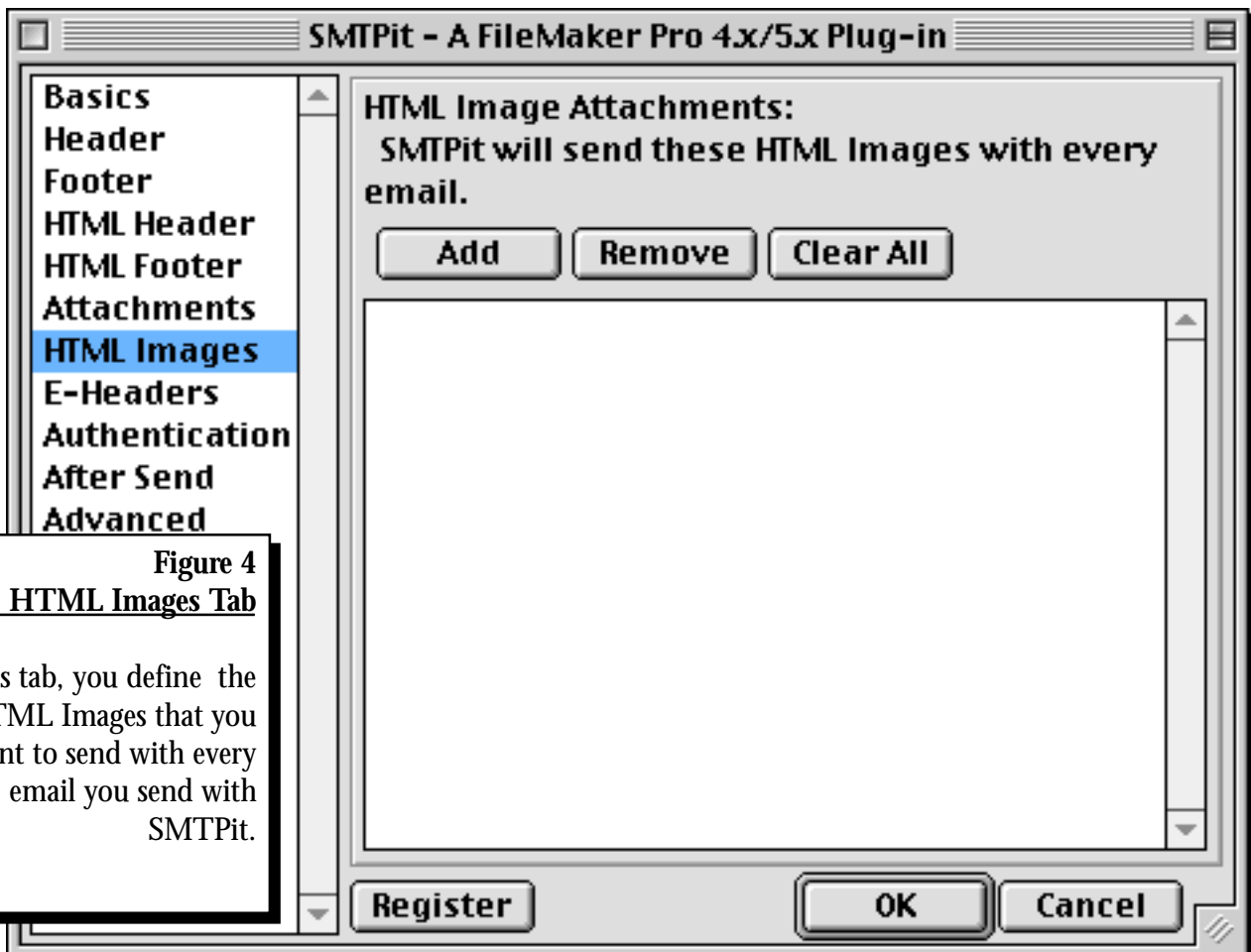


Figure 4
HTML Images Tab

On this tab, you define the HTML Images that you want to send with every email you send with SMTPit.

The **SMTP-DlgHTMLImageAssign** and the **SMTP-DlgHTMLImageAppend** functions present you with a standard open file dialog to choose an HTML Image just like the Configuration Dialog mentioned in the last paragraph. For more information on these functions, and on using inline HTML Images, refer to the **HTML Image Example Explained** and the **SMTPit External Functions Reference** sections later in this documentation, or refer to the SMTPit Functions database that came with the SMTPit archive.

Email Headers

SMTPit also has the ability to insert extra Email Headers into your email messages. For example, you can identify your company by creating an Email Header with the name, "X-Company", and the value being the name of your company. When the recipient gets your email message they will see "X-Company: Your Company Name" at the top of the email. It is important that you do not have any spaces in the Name of the header due to Email Header specifications, however, you can have spaces in the value. You can also use Email Headers to track different email promotions or use in conjunction with POP3it to run a script based on an Email Header.

To add a Default Email Header simple click on the **Add** button as shown below (See Figure 5), then type in your desired header name in the name field with the desired value in the value field. You can also add email headers dynamically by using the **SMTP-EmailHeaderAppend** function, which will append new headers. The default Email Headers you define in the Configuration Dialog will be

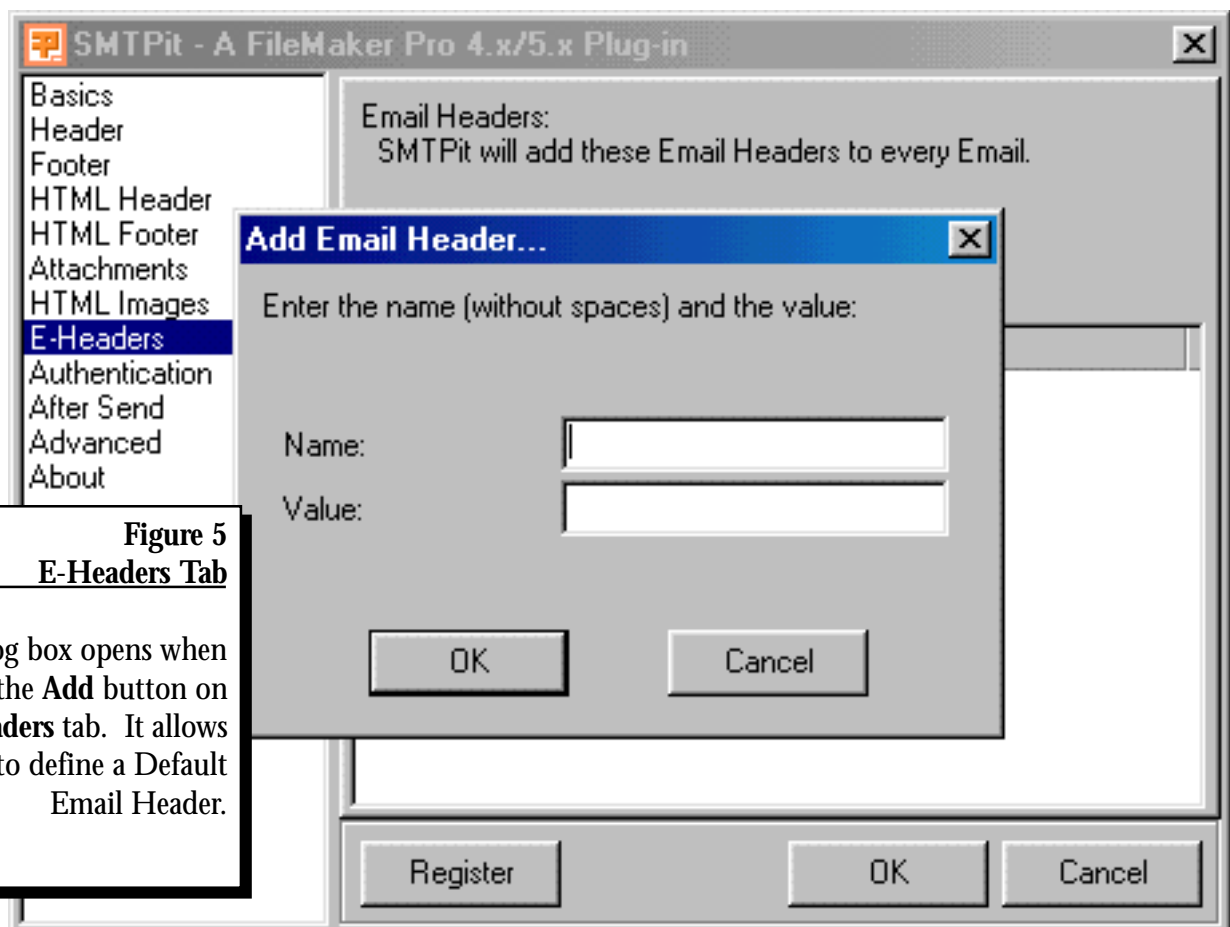


Figure 5
E-Headers Tab

This dialog box opens when you press the **Add** button on the **E-Headers** tab. It allows you to define a Default Email Header.

appended to the header of the email before any headers added by the **SMTP-EmailHeaderAppend** function.

Authentication

SMTPit 3.0 adds support for mail servers that require you to log in before you can send email. SMTPit supports the following Authentication Types (in increasing level of security): None (off), Plain, Login, CRAM-MD5. To set SMTPit to use the new authentication support, set the **Type** to something other than None, and fill in the **Username** and **Password** fields with the settings for your email account (See Figure 6).

After Send

The After Send settings (See Figure 7, Next Page) will determine what action SMTPit takes after sending each email. If you would like SMTPit to pop up a dialog window letting you know that your email message was sent, check the Popup "Email Sent Successfully" Dialog checkbox. This option is often helpful when debugging your scripts. You can also tell SMTPit to pop up this dialog on a per email basis with the new optional parameters to the **SMTP-Send** function. For more information on how to do this, see the **SMTPit External Function Reference** later in this document, or the SMTPit Functions database that came in the SMTPit archive.

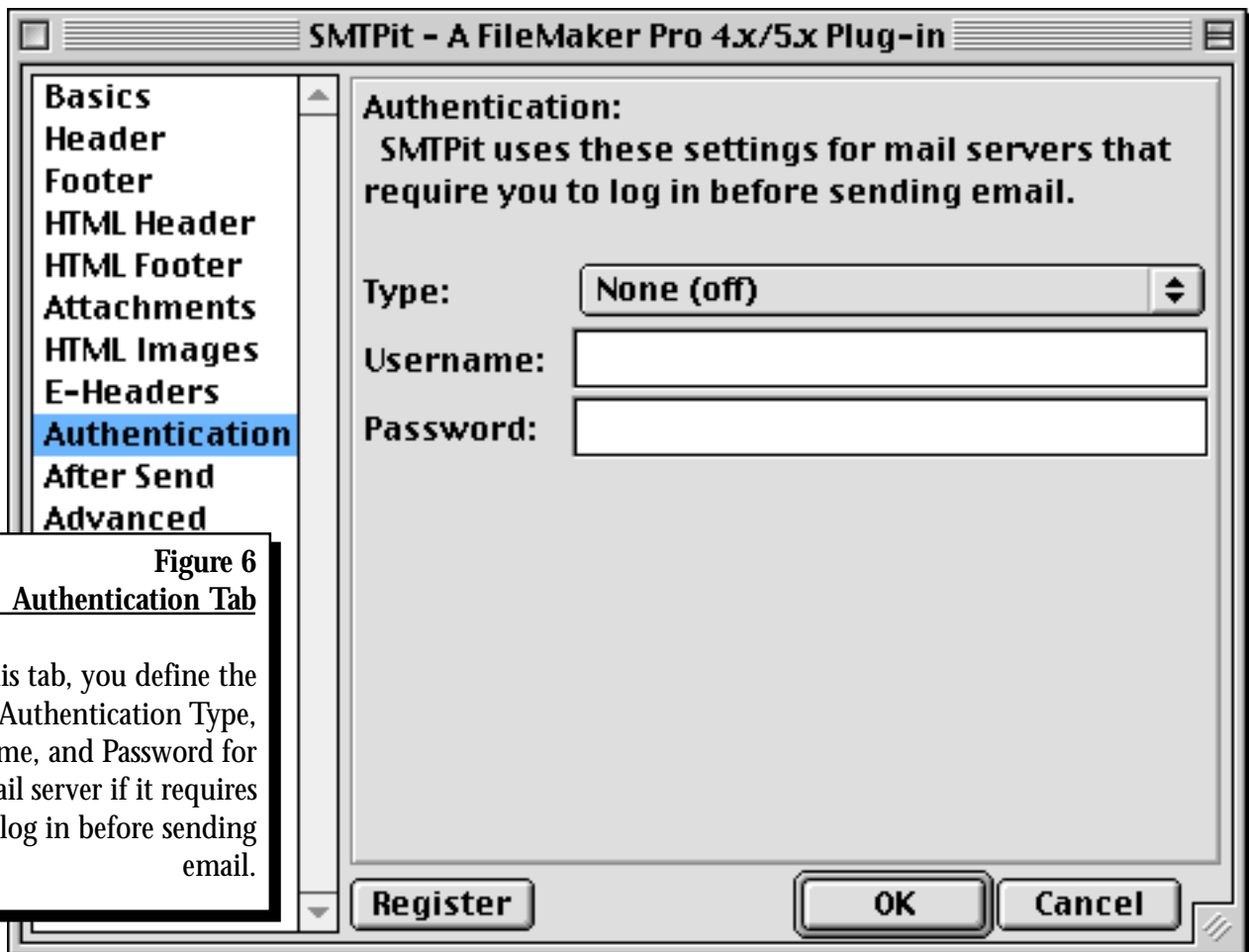


Figure 6
Authentication Tab

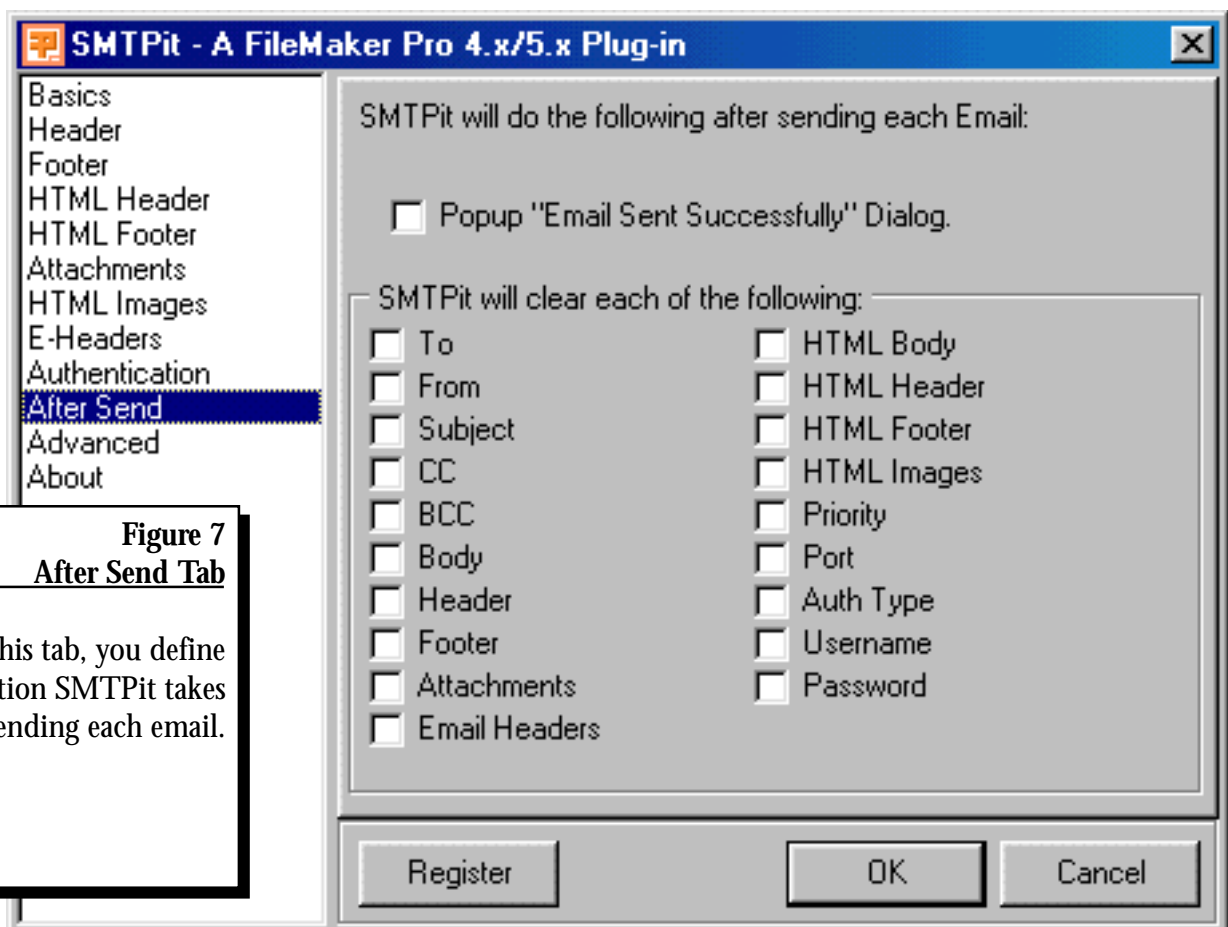
On this tab, you define the Authentication Type, Username, and Password for your mail server if it requires you to log in before sending email.

You also have the option to clear the following fields: **To, From, Subject, CC, BCC, Body, Header, Footer, Attachments, Email Headers, HTML Body, HTML Header, HTML Footer, HTML Images, Priority, Port, AuthType, Username, and Password.** Fields that are checked will clear after each email message is sent. Fields that are not checked retain their set values after each email message is sent.

To better explain this, here is an example scenario: You have a database of contacts and you want to send a form letter to each of those contacts. The body of each of these emails is exactly the same, but you want the subject to say "Dear Frank", where "Frank" is the name of the contact. To do this, put "Dear " in the **Subject** field on the **Basics** tab, and on the **After Send** tab, check off **To** and **Subject**, and leave everything else unchecked. Next, make a script for your database that first assigns the body of the message and then goes into a loop to send all the messages. Inside this loop, you only have to assign the "To" field, and use **SMTP-SubjectAppend** to add on the contact's name. Everything else, including the body, will be retained after each email is sent for use in the next email.

Advanced

New with SMTPit 3.0 is the **Advanced** tab. On this tab are advanced features that you probably will never need, and probably should not touch. However, if you know what you are doing, or if we instruct you to do so, you can change these settings.



The internal character set inside FileMaker Pro is not the more widely used character set used on the internet. Therefore SMTPit must convert the FileMaker Pro character set before it can send an email so that the characters appear correctly in the receiving email client. For the most part, the only characters that are affected are the "High ASCII" characters which are usually accented characters and some mathematical symbols. If you do not want SMTPit to do this character conversion, you can switch it off by checking the **Disable Character Conversions** checkbox (See Figure 8).

If you do want SMTPit to do its character conversions, but it appears to be doing the wrong character conversion for your localized system, you can change the **Convert To** setting to match your needs. For instance, if you have a Japanese version of FileMaker Pro and your Japanese characters are not being sent correctly, make sure the **Convert To** setting shows "ISO-2022-JP (Japanese)". When you first run SMTPit, it should auto-detect which character set your computer is set to, and set the **Convert To** setting accordingly. However, if it does not auto-detect, you can change the setting.

If your mail server requires you to connect on a different TCP/IP port than the standard port 25, you can change the **Port** setting to fit your needs.

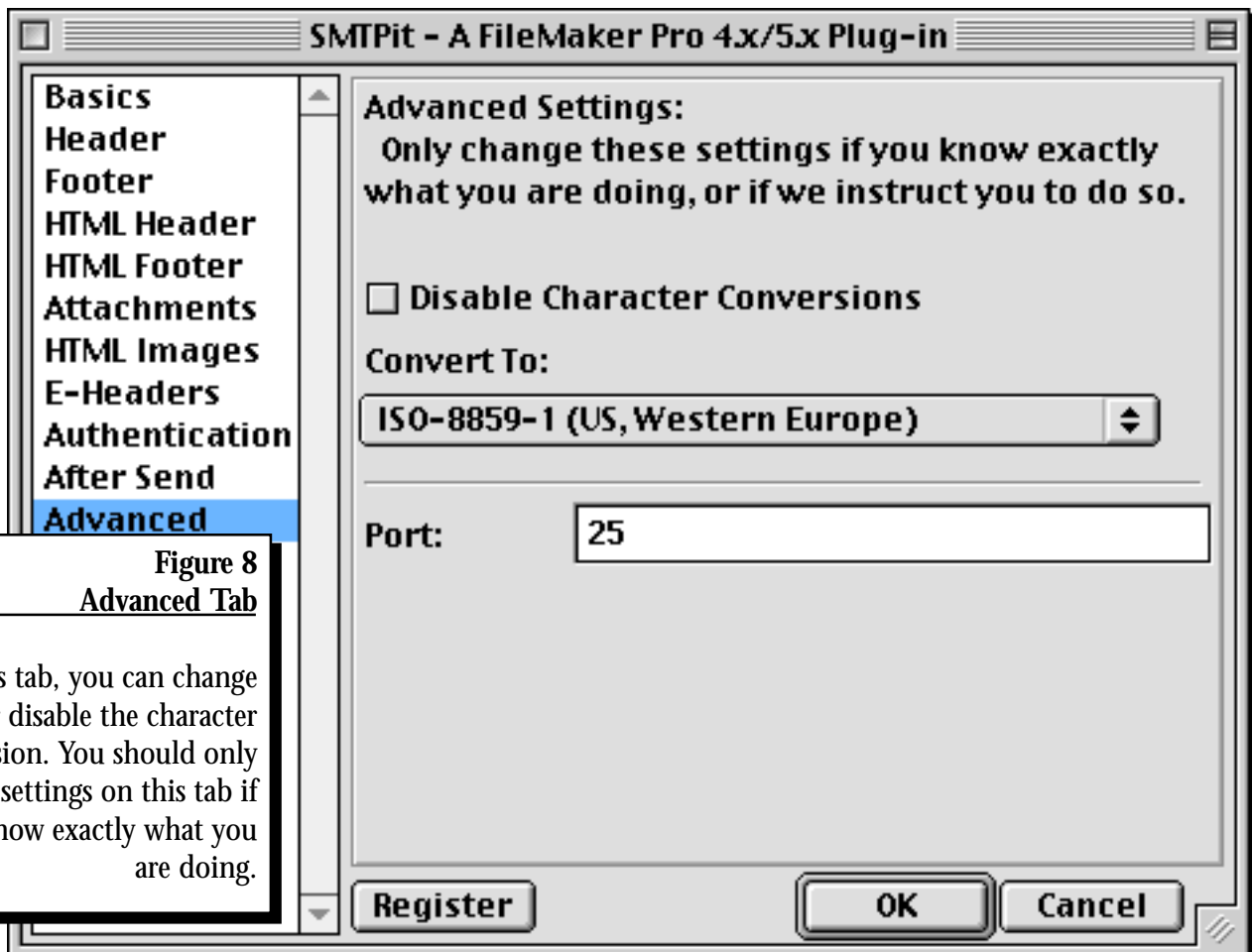


Figure 8
Advanced Tab

On this tab, you can change or disable the character conversion. You should only change settings on this tab if you know exactly what you are doing.

About

The About Tab simply reports which version of SMTPit that you are using. You can also use the **SMTP-Version** function to bring up the Configuration Dialog by passing the function the string "CONFIGURE" or the string "ABOUT". To make sure that you have the most recent version of SMTPit, please visit our website (<http://www.smtpit.com/>).

Registering

You can register your copy of SMTPit from the Configuration Dialog once you have purchased it from our secure website. After you purchase SMTPit, we will send you a registration number to register your copy by using the **Register** button in the Configuration Dialog (See Figure 9). We have also included a **SMTP-Register** function so that developers can easily register SMTPit with their bound solutions once they have purchased a Developer's License. For more information on purchasing SMTPit and other exciting plug-ins, visit our website (<http://www.cnsplug-ins.com/>) and choose the **Purchase** link from the tool bar.

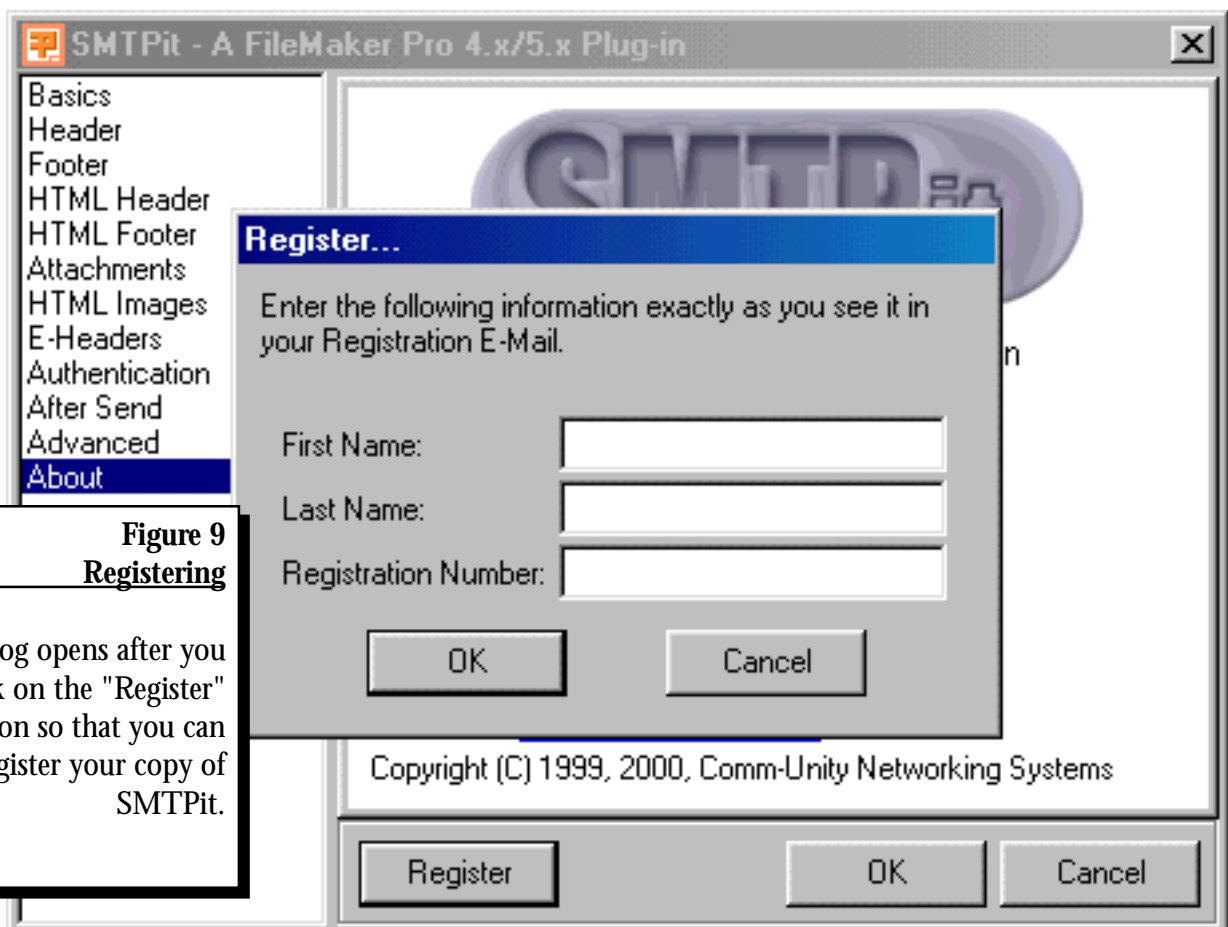


Figure 9
Registering

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

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.

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 a list of all the External Functions of all the plug-ins 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 SMTPit installed (and enabled) you will see the SMTPit External Functions listed as well. Choosing to **View by "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 choose External Functions by double clicking on them in this list. This will ensure that you have the correct spelling of the function.

FileMaker Pro's External Function looks like the following:

External ("PluginNamePrefix-FunctionName", parameter)

For example take our **SMTP-Version** function, which returns the version of SMTPit that you are

using. When you double-click the External Function in the External Function list, it inserts this into your calculation:

```
External ("SMTP-Version", parameter)
```

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

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

The parameter of the External Function is where you put the information for that function. For example, if you are setting your From Email Address using the **SMTP-FromAssign** function, you could 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 could use it 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. This is because FileMaker Pro will interpret anything 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 called "Contact Name" and a field called "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 "Jake Traynham", and your "Contact Email" field contained "jake@comm-unity.net", then FileMaker Pro would turn the above into:

```
External ("SMTP-ToAssign", "Jake Traynham" & " <" & "jake@comm-unity.net" & ">")
```

And further into this:

```
External ("SMTP-ToAssign", "Jake Traynham <jake@comm-unity.net>")
```

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

Once you understand the 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 Help Files for FileMaker Pro located in the **Help** menu.



Example Databases Explained

Quick Start Example Explained	19
Simple Mailer Example Explained	21
Send - Text Script	22
Add Attachment Script	24
Mass Mailer Example Explained	26
Go Script	26
HTML Image Example Explained	31

Quick Start Example Explained

The Quick Start Example database is a very simple database that should help you understand the very basics of sending email with SMTPit. This database contains the following six fields:

Field Name	Field Type
SMTP Host	Text
To Email Address	Text
From Email Address	Text
Subject	Text
Body	Text
SMTPit Result	Text

The first five fields should be self-explanatory; you should be able to fill them out with your information. The sixth field, **SMTPit Result**, will hold the results of the SMTPit External Functions. You do not need to put anything in this field because when you press the **Send** button, the **SMTPit Result** field will be filled in.

Let us look at the scripts in this database. Go to the **Script** menu and select **ScriptMaker...** Select the **Send (with fields)** script, and press the **Edit** button. The **Script Definition for "Send (with fields)"** dialog should open. In the big text area on the right, select the **Set Field** Script Step. Near the bottom of the dialog, some buttons should have appeared when you selected the **Set Field** Script Step. Press the **Specify...** button. Now the **Specify Calculation** dialog is open. You should see several `External()` functions in the large text area. These should look familiar if you read the **How To Use FileMaker Pro Plug-ins** section earlier in this documentation. You should see the following:

A	<code>External("SMTP-ClearAll", "") & "¶" &</code>
B	<code>External("SMTP-HostAssign", SMTP Host) & "¶" &</code>
C	<code>External("SMTP-ToAssign", To Email Address) & "¶" &</code>
D	<code>External("SMTP-FromAssign", From Email Address) & "¶" &</code>
E	<code>External("SMTP-SubjectAssign", Subject) & "¶" &</code>
F	<code>External("SMTP-BodyAssign", Body) & "¶" &</code>
G	<code>External("SMTP-Send", "")</code>

Line **A** tells SMTPit to clear any previous settings it may have. It is a good idea to always start with this function so that you can ensure you are starting with a clean slate. Line **B** tells SMTPit the mail server host to connect to using the value you specified in the **SMTP Host** field. Line **C** tells SMTPit to email address of the person you are sending email to, using the value you specified in the **To Email Address** field. Line **D** tells SMTPit the **From Email Address**. Line **E** tells SMTPit the subject of the email you are creating. Line **F** tells SMTPit the content of the email. And finally, Line **G** tells SMTPit to send the email you just created.

That is all there is to it. If you fill in those five fields, **SMTP Host**, **To Email Address**, **From Email Address**, **Subject**, and **Body**, and then press the **Send** button, then SMTPit will send an email based on the values you put in those fields. If SMTPit has an error with something you entered into the fields, or if it has a problem connecting to your mail server or sending your email, you can look at the **SMTPit Result** field. This field will contain the results of each of the functions in the **Set Field** Script Step in the **Send (with fields)** script.

You can also look at the **Send (with hard-coded values)** script to see how you can enter in literal values into the SMTPit External Functions. You can mix and match literal values with fields as well. Now that you know how simple it is to send email with SMTPit, you can add a few fields and a script or two to your own databases, and have email sending capabilities in no time!

Simple Mailer Example Explained

This section of the documentation will explain the **Send - Text** script of the Simple Mailer Example database. This script is a little more advanced than the script in the Quick Start Example database. It should give you a few more ideas on how to use some of the more advanced features of SMTPit, while not getting overly advanced.

There are fourteen fields in this database that are used by the **Send - Text** script:

Field Name	Field Type
SMTP Host	Global (Text)
Authentication Type	Global (Text)
Username	Global (Text)
Password	Global (Text)
From Email Address	Global (Text)
To Email Addresses	Text
CC Email Addresses	Text
Subject	Text
Priority	Text (Auto Enter:"Normal")
Attachments	Text
Text Body	Text
Text Signature	Global (Text)
Setup Result	Text
Send Result	Text

Note that not all fields are text like in the Quick Start Example database. The **SMTP Host**, **Authentication Type**, **Username**, **Password**, **From Email Address**, and **Text Signature** fields are global fields which will allow you to have the same value for each record in your database. An alternative way to do this would be to use the default settings in the Configuration Dialog as discussed earlier in this documentation. The **Priority** text field is a little different in that it has an Auto Entered value of "Normal". This will set the priority to it's default value, "Normal", when you create a new record.

Send - Text Script

Now that you know the fields in the database, let us look at the "Send - Text" script. (It has been slightly modified for this documentation to exclude a few steps that do not pertain to sending email with SMTPit.)

```
1 Comment ["Check to make sure there are recipients."]
2 If ["IsEmpty(To Email Addresses) and IsEmpty(CC Email Addresses)"]
3     Show Message ["This email is missing recipients. Please enter..."]
4     Exit Script
5 End If
6 Comment ["Set up the email."]
7 Set Field ["Setup Result", "External("SMTP-ClearAll", "", & "¶"..."]
8 If ["PatternCount(Setup Result, "ERROR:")"]
9     Show Message ["There was an error setting up your email. Plea..."]
10 Else
11     Comment ["Send the message."]
12     Set Field ["Send Result", "External("SMTP-Send", "")"]
13     If ["PatternCount(Send Result, "Success")"]
14         Show Message["Your email was sent!"]
15     Else
16         Show Message["There was an error sending your email. Please..."]
17     End If
18 End If
```

Line 1 contains a simple **Comment** Script Step. **Comment** Script Steps are useful for making notes about what a particular part of the script is doing so that if you ever need to modify your script, you can easily find the part you need to modify. You can also use **Comment** Script Steps to remind yourself why you did something a certain way or what parts you should not touch so that you do not break something.

Line 2 contains an **If** Script Step. An **If** Script Step can be used to check a value and then take a course of action based on the result. In this instance, we are checking to see if the **To Email Addresses** and **CC Email Addresses** fields are empty or not. If they are empty, then in Line 3 we inform the user with a **Show Message** Script Step that they need to put an email address in the **To Email Addresses** Field, and then in Line 4, we exit the script. An **Exit Script** Script Step tells FileMaker Pro to stop executing the script and return control to the user. FileMaker Pro will not execute any more Script Steps after an **Exit Script** Script Step.

If there is at least one email address in the **To Email Addresses** or **CC Email Addresses** fields, then Line 3, 4, and 5 will be skipped and the next Line will be Line 6, which is another **Comment** Script Step.

Next is Line 7 which is a **Set Field** Script Step. Remember in our Quick Start Example database, all of our interaction with SMTPit happened in the calculation field of a **Set Field** Script Step. Well, this is where the "Setup" SMTPit functions are for this script. If you look at the calculation field of this **Set Field** Script Step, you will see the following:

A	External("SMTP-ClearAll", "") & "¶" &
B	External("SMTP-HostAssign", SMTP Host) & "¶" &
C	External("SMTP-AuthTypeAssign", Authentication Type) & "¶" &
D	External("SMTP-UsernameAssign", Username) & "¶" &
E	External("SMTP-PasswordAssign", Password) & "¶" &
F	External("SMTP-FromAssign", From Email Address) & "¶" &
G	External("SMTP-ToAssign", To Email Addresses) & "¶" &
H	External("SMTP-CCAssign", CC Email Addresses) & "¶" &
I	External("SMTP-SubjectAssign", Subject) & "¶" &
J	External("SMTP-PriorityAssign", Priority) & "¶" &
K	External("SMTP-AttachAssign", Attachments) & "¶" &
L	External("SMTP-BodyAssign", Text Body) & "¶" &
M	External("SMTP-FooterAssign", Text Signature)

Lines **A** and **B** should be familiar to you from the Quick Start Example database. Lines **C**, **D**, and **E** set the **Authentication Type**, **Username**, and **Password** for your email account on the mail server. If your mail server requires you to log in before sending email, then you can set values in these fields in the **Preferences** layout. If you set the **Authentication Type** to None and leave the **Username** and **Password** fields blank, then these External Functions will not do much, but they will not return errors for giving them blank values.

Lines **F** and **G** should also look familiar from the Quick Start Example. Line **H** has another new function, **SMTP-CCAssign**, which works exactly like **SMTP-ToAssign**, except that it sets the CC or Carbon Copy Email Addresses instead of the To Email addresses. Any email addresses you have in the **CC Email Addresses** field will be sent a copy of the email that you are sending.

Line **I** looks the same as the Quick Start Example, but Lines **J** and **K** are new. Line **J** sets the priority of the email. Most modern email clients can recognize the priority of email and will highlight your email as being more or less important than other emails sent to them. Line **K** sets the File Attachments you want to send with your email. You can use the **Add** button next to the **Attachments** field on the layout to add extra attachments, or you can type them in yourself. Just remember that you need the Full Path and FileName of the attachment on your hard drive so that SMTPit can find it. If you do not need to send any files with your email, you can leave the **Attachments** field blank, and SMTPit will just ignore your **SMTP-AttachAssign** function.

Line **L** looks like the Quick Start Example, but Line **M** shows another new function. This time we are using the **SMTP-FooterAssign** function to attach an Email Signature. SMTPit will put anything you place in the "Footer" onto the end of your email. So, after you fill in all of your email content in the **Body** field, SMTPit will stick your Email Signature onto the end for you before it sends the email.

Next we go back to our **Send - Text** script and look at Line **8**. Line **8** uses another **If Script Step** and the **PatternCount** Text Function to search the **Setup Result** field for the word "ERROR:". The **Setup Result** field was filled in by all of your External Functions in the previous **Set Field** Script Step. So, after FileMaker Pro calls all of the External Functions for us, we can then look into the field and see if there were any errors. If there are errors, then we go to Line **9**. If there are no errors, then we go to Line **11**.

Line 9 uses another **Show Message** Script Step to inform the user that there was an error and that they need to look at the **Setup Result** field to determine why there was an error. FileMaker Pro would then jump to Line 19 after the **End If** Script Step, if there was a Line 19. Since there is not, FileMaker Pro stops executing the script and returns control to the user. The user could then inspect the **Setup Result** field for the error.

If there were no errors, then we go to Line 11 for our next Script Step. This line contains another **Comment** Script Step telling us that we are going to now send the message.

Line 12 uses another **Set Field** Script Step to call the **SMTP-Send** External Function. In this documentation, we have simplified the calculation for this **Set Field** Script Step to just contain "External("SMTP-Send", ""). This should look familiar from the Quick Start Example database, and like in the Quick Start Example database, this tells SMTPit to send the email. The actual script in the **Send - Text** script contains the optional **SMTP-Send** parameter, Transcript, and if you are interested in what that does, please see the **SMTP-Send** documentation in the **SMTPit External Functions Reference** later in this document or in the SMTPit Functions database.

Line 13 contains another **If** Script Step. This time, we are checking to see if the **SMTP-Send** function returned successfully. If SMTPit successfully sent the email, then the **Send Result** field will contain the value "Email Sent Successfully." Since it is possible that future versions of SMTPit might return a different success result, or might contain more information, it is a good idea to use the **PatternCount** Text Function to search for the word "Success". This is much safer than using an **If** Script Step Like this:

```
If ["Send Result = "Email Sent Successfully."]  
...  
End If
```

If you compare a value to a literal string like "Email Sent Successfully.", then you are setting yourself up for disaster if that literal string ever changes. This is why we encourage you to use the **PatternCount** Text Function to test the result of any plug-in External Function, not just ours.

If Line 13 found the word "Success" in the Send Result Field, then Line 14 is executed. Line 14 uses another **Show Message** Script Step to inform the user that the email was sent successfully. If Line 13 did not find the word "Success", then Line 16 is executed, which uses a **Show Message** Script Step to inform the user that their email was not successfully sent and that they should look at the **Send Result** field to determine why. Note that the documentation here differs from the actual **Send - Text** script in the database. The documentation shows a much simpler set of Script Steps instead of the layout switching and other Script Steps that go on in the actual database that don't really have anything to do with SMTPit.

Add Attachment Script

This script simply calls the **SMTP-FileNameAcquire** function, which brings up a standard open file dialog that allows you to locate a file on your hard drive that you want to attach. This function does not actually attach the file to the email, but instead returns a full path and filename to the file and

places it in the **Attachments** field. When one of the **Send - XXX** scripts is called, the contents of the **Attachments** field is sent to the plug-in so that it can attach the files.

Well, now you have seen a slightly more advanced script for sending email with SMTPit. We hope that you are starting to see that SMTPit is easy to use and flexible enough to do just about anything you want with your email sending databases.

Mass Mailer Example Explained

The Mass Mailer Example database shows how to exploit several of the unique features of SMTPit. The most important feature that makes this Mass Mailer Example database work so well is SMTPit's ability to retain settings from one email to the next. Generally, the content of mass mail-outs is the exact same for every email. So, to be more efficient, SMTPit allows you set the Body of the email once at the very beginning, and then use that same Body for every email you send. You can do the same with any of the other SMTPit settings. The Mass Mailer Example database also keeps the same From Email Address and Subject for all emails.

This example database also makes use of the new **SMTP-Connect** and **SMTP-Disconnect** functions in SMTPit 3.0. In previous versions of SMTPit, the **SMTP-Send** button did all the work. It would connect you to the mail server, send a single email, and then disconnect. The **SMTP-Send** function in SMTPit 3.0 works the exact same way, if you use it by itself, however, this is not very efficient in a mass mail-out solution. So, instead of connecting and disconnecting for every single email, you can now use the **SMTP-Connect** and **SMTP-Disconnect** functions before and after your mass mail-out loop. If you call **SMTP-Connect**, then **SMTP-Send** will see that it's already connected, and just simply send the email. This is much more efficient. Please note, though, that if you do use **SMTP-Connect**, you must use **SMTP-Disconnect** when you are done. Otherwise, SMTPit will never disconnect you from your mail server.

Go Script

Let us look at the main script that starts and runs the whole mass mail-out process:

```

1 Enter Browse Mode []
2 Go to Layout ["Main"]
3 Replace [No Dialog, "Send Result", ""]
4 Set Field ["Start Time", "Status(CurrentTime)"]
5 Perform Script [Sub-scripts, " \_ Connect"]
6 If ["PatternCount(Setup Result, "ERROR:") > 0"]
7     Show Message ["There was an error setting up the email. Pleas..."]
8 Else
9     Go to Layout [Refresh window, "List"]
10    Go to Record/Request/Page [First]
11    Loop
12        Perform Script [Sub-scripts, " \_ Send One"]
13        Flush Cache to Disk
14        Go to Record/Request/Page [Exit after last, Next]
15        Refresh Window []
16    End Loop
17    Go to Layout [Refresh window, "Main"]
18 End If
19 Perform Script [Sub-scripts, " \_ Disconnect"]
20 Set Field ["End Time", "Status(CurrentTime)"]
21 Perform Script [Sub-scripts, " \_ Calculate Duration"]

```

Lines 1 and 2 just make sure you are in the right mode and on the right layout. Line 3 uses the **Replace Script Step** to run through every record and clear out the **Send Result** field. If you need to stop your mass mail-out for one reason or another, then you will also want to be able to find where you left off in the database. The easiest way to do this is to look and see which is the first record that does not have something in the **Send Result** field. However, if you have run this script before, then the **Send Result** fields of all of the records would have the results from the last send, making it impossible to find where it left off. This is why all the **Send Result** fields are cleared out with a **Replace Script Step** before we start a new mail-out.

Line 4 sets the **Start Time** field with the current time so that when we are done, we can see how long it took to send all of the emails.

Line 5 calls our **Connect** script to set up the email and connect to the mail server. This script contains the following calculation:

```

A External("SMTP-ClearAll", "") & "¶" &
B External("SMTP-HostAssign", SMTP Host) & "¶" &
C External("SMTP-AuthTypeAssign", Authentication Type) & "¶" &
D External("SMTP-UsernameAssign", Username) & "¶" &
E External("SMTP-PasswordAssign", Password) & "¶" &
F External("SMTP-FromAssign", From Email Address) & "¶" &
G External("SMTP-SubjectAssign", Subject) & "¶" &
H External("SMTP-BodyAssign", Text Body) & "¶" &
I External("SMTP-HTMLBodyAssign", HTML Body) & "¶" &
J External("SMTP-Connect", "")

```

Most of this should look familiar from the Quick Start Example and Simple Mailer Example databases. There are two differences, however, that should be pointed out. First, there is no **SMTP-ToAssign** in this calculation. The reason for this is that this is the setup script that happens before

our main mail-out loop. The **To Email Addresses** will be different for each record, so we want to use the **To Email Address** inside our main mail-out loop, not before it. The second difference is the new **SMTP-Connect** function on Line J. This connects you to your mail server so that you can start sending all of your emails.

Now back to the **Go** script and Line 6. This line uses the **If Script Step** to see if anything in the **Connect** script had problems. If it can find the word "ERROR:" in the **Setup Result** field, then something did not work right, and it tells the user there was a problem with a **Show Message** Script Step in line 7.

If, however, the **Connect** script was successful, then we go to Line 9, which takes us to the List View layout so we can watch as SMTPit sends each of the emails. Line 10 will take us to the top of this list with the **Go to Record/Request/Page** Script Step with the "First" option. Using this Script Step does not effect the Sort Order, so if you have several contacts and you want to send them in a certain order, like alphabetically by **Contact Name**, then you can sort your records first, and then run the **Go** script.

Lines 11 through 16 contain our main mail-out loop. This is the section of the script that gets repeated over and over for every single one of the records in the database. Line 12 calls the **Send One** script which sends an email with the Contact information in the current record. We will look at this script in a second.

Line 13 uses a lesser known Script Step called **Flush Cache to Disk**. To be more efficient and responsive to the user, FileMaker Pro keeps all database changes in your computer's RAM and only writes those database changes to disk when the program is idle, like when you are reading a field in the database. This is normal operation and great for when you are doing your normal daily database operations, however it is not so great for a mass mail-out solution.

The reason it is not so great is that FileMaker Pro never has any idle time to write those changes to disk when you are running through 1000 or so records sending emails to each one. If FileMaker Pro never has any idle time to write those changes to disk, then the changes start stacking up in RAM. This can be a bad thing for a number of reasons. Your computer or FileMaker Pro could crash for some unknown reason and lose all of those database changes since they were never written to disk. If FileMaker Pro's memory gets full of these database changes that have not been written to disk, and you try to make another change, then FileMaker Pro will finally give in and clear out it's memory by writing all the changes to disk. This can make it look like FileMaker Pro has stopped responding and can take a long time if you have allocated a lot of RAM to FileMaker Pro.

So, to remedy this situation, you need some way to tell FileMaker Pro to write all of the database changes to disk when you want. This is the purpose of the **Flush Cache to Disk** Script Step. This tells FileMaker Pro to write any and all database changes to your hard drive, right now.

After you have sent the email for the current record, and flushed those database changes to disk, then you need to go to the next record. Line 14 uses the **Go to Record/Request/Page** Script Step, this time with the "Next; Exit after Last" option to do this. The extra "Exit after Last" option allows you to break out of the main mail-out loop when you have come to the end of your list of contacts.

Line 15 simply refreshes or redraws the screen so that you can watch it progress through your records. Without this Script Step, you would just see a blank screen after the first few records have sent, and the only way you could tell it was working is it would flicker slightly every once in a while.

Line 17 takes you back to the **Main** Layout after it has completed sending all of the emails. Line 19 calls the **Disconnect** script which simply uses the **SMTP-Disconnect** function to disconnect from your mail server. Line 20 sets the **End Time** field with the current time, and Line 21 calls the **Calculate Duration** script which fills in the number of hours, minutes, and seconds it took to send your complete mail-out.

Now let us take a look back at the **Send One** script called from Line 12:

```
1 If ["not IsEmpty(Contact Email Address)"]
2   If ["Contact Name <> """]
3     Set Field ["Send Result", "External("SMTP-ToAssign", """" &...""]
4   Else
5     Set Field ["Send Result", "External("SMTP-ToAssign", "<" & ..."]
6   End If
7   Set Field ["Send Result", "Send Result & "¶" & External("SMTP..."]
8   Set Field ["Send Result", "External("SMTP-Send", "") & "¶" & ..."]
9 End If
```

Line 1 looks to see if this record has an email address. If it does not, then this script will exit. If the record does have an email address, then it goes to Line 2.

Line 2 determines if the current record has a **Contact Name**. If there is a **Contact Name**, then Line 3 is run which uses this calculation:

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

This calculation simply takes the **Contact Name** and **Contact Email Address** and makes it look like "Jake Traynham" <jake@comm-unity.net>.

If there is no **Contact Name**, then all we have is the **Contact Email Address**. Line 5 then uses this calculation:

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

Which simply turns the **Contact Email Address** into <jake@comm-unity.net>.

Line 7 contains the rest of the setup for this single record. It contains the following calculation:

A	Send Result & "¶" &
B	External("SMTP-EmailHeaderAppend", "X-RecID=" & NumToText(Contact ID)) & "¶" &
C	External("SMTP-HeaderAssign", Text Header) & "¶" &
D	External("SMTP-HTMLHeaderAssign", HTML Header) & "¶" &
E	External("SMTP-FooterAssign", "To Remove: send email to "" & From Email Address & "" with the subject "REMOVE " & Contact Email Address & "".) & "¶" &
F	External("SMTP-HTMLFooterAssign", "<p>Remove Me")

This calculation is using several External Functions to enter in all sorts of information about the current record. It first uses the unique **SMTP-EmailHeaderAppend** function in Line B to add an email header to the header portion of the email. (The header portion of an email is the portion that contains the To, From, Subject, and other similar lines.) We use the **SMTP-EmailHeaderAppend** function to add a header that looks like "X-RecID: 47". In every mass mail-out you do, there will most likely be email addresses that were invalid or no longer in use. For those bad email addresses, you will receive a bounced or returned email saying the email could not be delivered. The majority of these bounced emails will at least have the header portion of the original email. So, by putting the "X-RecID" email header in the email, you can quickly take a look at the bounced message, find the "X-RecID" email header, and know exactly which record to look at in your database to correct or delete. You could also use an email receiving plug-in, like our popular POP3it plug-in, to receive these bounced messages and remove or flag bad email addresses automatically.

Lines C and D use the **SMTP-HeaderAssign** and **SMTP-HTMLHeaderAssign** functions to insert our special "Dear xxx," greetings at the top of the Body of our email. Take note that the header *portion* of an email is not the same as the header *section* of the body of the email. The header *portion* of an email contains all the To, From, Subject and other similar lines, while the header *section* of the body of the email is an optional section that goes before the Body of your email much like the footer section goes after the Body of your email.

Lines E and F use the **SMTP-FooterAssign** and **SMTP-HTMLFooterAssign** functions to insert special remove instructions for the recipient of the message to follow if they no longer wish to receive emails from you. The versions here in this documentation are much shorter than the version in the actual Mass Mailer Example database for simplicity. These two lines show you how you can build part of the message "on the fly" with information in your record.

That concludes the Mass Mailer Example database. Hopefully you now have a better understanding of how to use SMTPit in a mass mail-out solution. Feel free to use this database as an example to build your own, or use this as a starting point for your own database.

HTML Image Example Explained

The HTML Image Example database is our final example database for SMTPit. This database shows you how to use the HTML Image feature introduced in SMTPit 2.5. This ability allows you to include inline graphics for use in your HTML emails. Normally, if you want to have images in your HTML emails, you have to place those images on a publicly available website. Then when the recipient of your email views your HTML email, it can download the images from that website and display them to the user. Unfortunately, there are many drawbacks to this. The recipient of your email could be viewing your email "offline" or while they are not connected to the internet. If they are not connected to the internet, then they cannot download the image to view it. It could also be quite slow for the recipient's email client to contact the website and download the image.

Having inline HTML images fixes this problem by including the actual image in the email, sort of like a file attachment. You can then set up your HTML img tags to use the image embedded in the email instead of downloading it from a website. Be warned, though, that you should not go overboard with this and include a lot of graphics or include large graphics. Just like it can take a long time to download an image from a website, it can also take a long time to download an email that has several graphics or large graphics in it. You should try and keep the images small and stick to no more than 3 or 4, otherwise your recipients may not like it, especially if they have a slow connection to the internet.

With that in mind, let us look at the HTML Image Example database. You should find that the database contains the familiar fields from the other databases including **To**, **From**, **Subject**, **Body**, and **Result**. A new field is here, though, called **HTML Images**. In this field, you enter the Full Paths and FileNames of the HTML Images that you want to include in your email. Separate multiple images with a return. You can also use the **Add** button to use a standard open file dialog to select images to add to your email.

Also take a look at the HTML code in the **HTML Body** field. You should see a couple HTML img tags in the body. They look something like this:

```

```

The "src" parameter of the img tag is the key to showing the correct images at the correct locations. The parameter must always contain "cid:part#" where "#" is the number of the attachment in the order that you assign them. When you use the **SMTP-HTMLImageAssign** and **SMTP-**

HTMLImageAppend functions, you need to remember the order in which you add the images. Let us say you have two graphic files, named "logo.gif" and "product.jpg", and you add them in that order, with "logo.gif" being the first, and "product.jpg" being the second. Then when you need to insert the "logo.gif" file in your email, you would specify the src of the img tag as "cid:part1", and when you need to insert the "product.jpg" file in your email, you would specify the src of the img tag as "cid:part2".

There are three extra things to note about HTML Images. First, if you are always going to be sending a specific graphic with every single one of your emails, then you can specify that graphic in the Default HTML Images in the Configuration Dialog. Any graphics you have specified there will always send with every email. They will also be placed first in the list of images. So, if you have two images in the Default HTML Images in the Configuration Dialog, and then you add two images with the **SMTP-HTMLImageAssign** function, the two images you assign with the function will be the third and fourth images, which would correspond to "cid:part3" and "cid:part4".

The second thing to note is that if you do have multiple images they do not need to be sequential in your email. In other words, an img tag with the src "cid:part3" could come before an img tag with the src "cid:part1" in your HTML code. The final thing to note is if you want to use the same graphic in multiple places, you don't need to attach it multiple times. For example, if you had a list of items in your email and you want a tiny version of your logo to be in front of each of those line items, then you can use the same "cid:part2" img src for each of them, if your mini logo graphic was the second image assigned.

Once you fill in all the fields with your information, and insert a few image FileNames, you can press the **Send** button to send the email. This button calls the **Send Message** script which contains the following calculation:

A	External("SMTP-ClearAll", "") & "¶" &
B	External("SMTP-ToAssign", To) & "¶" &
C	External("SMTP-FromAssign", From) & "¶" &
D	External("SMTP-SubjectAssign", Subject) & "¶" &
E	External("SMTP-HTMLBodyAssign", HTML Body) & "¶" &
F	External("SMTP-HTMLImageAssign", HTML Images) & "¶" &
G	External("SMTP-Send", "")

Lines **A** through **E** should look familiar to you from the examples explained earlier in this documentation. Line **F** is the new function that assigns the HTML Images you have defined in the **HTML Images** field to the email you are creating. Finally, Line **G** sends the email.

This concludes the explanations of the example databases. We hope that you now have a complete understanding of the possibilities of sending emails using SMTPit. Please keep in mind that the example databases we provide are just that, examples. They are by no means the only way to use SMTPit. We encourage you to play with the example databases, check out some of the other functions available in SMTPit, and to try out different things. If you have any questions about any of the example databases, and you cannot find answers in this documentation, then check our website dedicated to SMTPit at <http://www.smtpit.com/>. If you still cannot find answers to your questions, please feel free to email us at smtpit@comm-unity.net.



External Function Reference and Contact Information

SMTPit External Function Reference	35
SMTP-Version	35
SMTP-Register	35
SMTP-ShowStatus	35
SMTP-HideStatus	36
SMTP-MoveStatus	37
SMTP-ClearAll	37
SMTP-AssignAll	37
SMTP-HostAssign	38
SMTP-PortAssign	38
SMTP-AuthTypeAssign	38
SMTP-UsernameAssign	39
SMTP-PasswordAssign	39
SMTP-ToAssign	40
SMTP-ToAppend	41
SMTP-FromAssign	41
SMTP-SubjectAssign	42
SMTP-SubjectAppend	42
SMTP-CCAssign	42
SMTP-CCAppend	43
SMTP-BCCAssign	43
SMTP-BCCAppend	44
SMTP-BodyAssign	44
SMTP-BodyAppend	45
SMTP-FileBodyAssign	45

SMTP-FileBodyAppend	46
SMTP-HTMLBodyAssign	46
SMTP-HTMLBodyAppend	47
SMTP-HTMLFileBodyAssign	47
SMTP-HTMLFileBodyAppend	47
SMTP-Send	48
SMTP-HeaderAssign	49
SMTP-HeaderAppend	49
SMTP-HTMLHeaderAssign	49
SMTP-HTMLHeaderAppend	50
SMTP-FooterAssign	50
SMTP-FooterAppend	51
SMTP-HTMLFooterAssign	51
SMTP-HTMLFooterAppend	51
SMTP-PathToDBAcquire	52
SMTP-AttachAssign	52
SMTP-DlgAttachAssign	53
SMTP-AttachAppend	53
SMTP-DlgAttachAppend	54
SMTP-HTMLImageAssign	54
SMTP-DlgHTMLImageAssign	54
SMTP-HTMLImageAppend	55
SMTP-DlgHTMLImageAppend	55
SMTP-EmailHeaderAppend	56
SMTP-FileNameAcquire	56
SMTP-PriorityAssign	57
SMTP-Connect	57
SMTP-Disconnect	58

Obsolete Functions	58
--------------------------	----

Mass Mailer Example Explained	59
-------------------------------------	----

Credits	60
---------------	----

Contact Information	60
---------------------------	----

SMTPit 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.

For general information on how the External() functions work and how you can use them in your calculations, please see the **How to use FileMaker Pro Plug-ins** section earlier in this document.

Functions with an asterisk (*) are new with version 3.0. Functions with a pound sign (#) have changed with version 3.0

SMTP-Version

This function returns the current version of SMTPit 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("SMTP-Version", "")  
External("SMTP-Version", "CONFIGURE")  
External("SMTP-Version", "ABOUT")
```

SMTP-Register

This function allows you to register your copy of SMTPit 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("SMTP-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.)

SMTP-ShowStatus *

Use this function to show a status window that displays information about what the plug-in is currently doing. When you are sending an email, the status window will show you what part of the email it is currently sending as well as a progress bar indicating how much of that part it has completed. 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 **SMTP-HideStatus** in a "close" script that you define in **Edit->Preferences->Document** of your main database.

Examples:

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

See Also:

SMTP-HideStatus
SMTP-MoveStatus

SMTP-HideStatus *

Use this function to hide the status window that you had previously shown with the **SMTP-ShowStatus** function. You must call this function to hide the status window before closing the FileMaker Pro application (see **SMTP-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("SMTP-HideStatus", "")
```

See Also:

SMTP-ShowStatus
SMTP-MoveStatus

SMTP-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 **SMTP-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("SMTP-MoveStatus", "700,100")  
External("SMTP-MoveStatus", "-1,600")  
External("SMTP-MoveStatus", "-1,-1")
```

See Also:

SMTP-ShowStatus
SMTP-HideStatus

SMTP-ClearAll

The **SMTP-ClearAll** function clears all settings that have been set with the External Functions so that the next email can start with a clean slate. The default settings in the Configuration Dialog are not cleared and will be used in your next email unless you set new values with the External Functions. You can also set SMTPit to automatically clear or not clear specific fields on the **After Send** tab in the Configuration Dialog. The parameter should be the empty string ("").

Example:

```
External("SMTP-ClearAll", "")
```

See Also:

SMTP-AssignAll
All Assign and Append functions.

SMTP-AssignAll

Use this function to assign any or all of the following settings at once: Host, To, From, CC, BCC, Subject, Port, AuthType, Username, and/or Password. The **SMTP-AssignAll** function will only set those settings; for all other SMTPit values, you must use their Assign and Append functions (see below). Note: The settings can be in any order, but you must separate them with the paragraph mark (§) or with the pipe character (|). Also note: You do not have to use this function to set these settings, this is just one way to set them. You can use the other Assign and Append functions below, some of which have better support for their specific setting.

Examples:

```
External("SMTP-AssignAll", "Host=smtp.your.com¶To=theiremail@their.com¶  
From=youremail@your.com")
```

```
External("SMTP-AssignAll", "CC=otherpeople@their.com¶BCC=blindcopies@their.com¶  
Subject=your subject")
```

See Also:

SMTP-ClearAll

All Assign and Append functions.

SMTP-HostAssign

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

Example:

```
External("SMTP-HostAssign", "mail.yourdomain.com")
```

See Also:

SMTP-PortAssign

SMTP-AuthTypeAssign

SMTP-UsernameAssign

SMTP-PasswordAssign

SMTP-Send

SMTP-Connect

SMTP-PortAssign *

Use this function to specify an alternate TCP/IP port for SMTPit to connect to on your mail server. SMTPit defaults to the standard SMTP TCP/IP port: 25. You will most likely never need to use this function.

Example:

```
External("SMTP-PortAssign", "16384")
```

See Also:

SMTP-HostAssign

SMTP-AuthTypeAssign *

If your mail server requires authentication (requires you to log in) before it allows you to send email, then use the **SMTP-AuthTypeAssign** function to assign the type of authentication your mail server

requires. The types of authentication SMTPit supports are (in increasing levels of security): None (off), Plain, Login, and CRAM-MD5.

Examples:

```
External("SMTP-AuthTypeAssign", "NONE")
External("SMTP-AuthTypeAssign", "PLAIN")
External("SMTP-AuthTypeAssign", "LOGIN")
External("SMTP-AuthTypeAssign", "CRAM-MD5")
```

See Also:

SMTP-HostAssign
SMTP-UsernameAssign
SMTP-PasswordAssign
SMTP-Send
SMTP-Connect

SMTP-UsernameAssign *

If your mail server requires authentication (requires you to log in) before it allows you to send email, then use the **SMTP-UsernameAssign** function to assign the username of your email account. Note: You will also need to use the **SMTP-AuthTypeAssign** function to specify what type of authentication your mail server needs. Otherwise, SMTPit will ignore this setting.

Example:

```
External("SMTP-UsernameAssign", "frank")
```

See Also:

SMTP-HostAssign
SMTP-PasswordAssign
SMTP-Send
SMTP-Connect

SMTP-Password *

If your mail server requires authentication (requires you to log in) before it allows you to send email, then use the **SMTP-PasswordAssign** function to assign the password of your email account. Note: You will also need to use the **SMTP-AuthTypeAssign** function to specify what type of authentication your mail server needs. Otherwise, SMTPit will ignore this setting.

Example:

```
External("SMTP-PasswordAssign", "love")
```

See Also:

SMTP-HostAssign
SMTP-UsernameAssign
SMTP-Send
SMTP-Connect

SMTP-ToAssign

Use the **SMTP-ToAssign** function to assign the "To" email address(es). Once assigned you can add other email addresses by using the **SMTP-ToAppend** function. Note: If you use the **SMTP-ToAssign** function twice, the second will overwrite the first. You can assign multiple email addresses with this function. If you have multiple email addresses and you want to set them all at once, separate each email address with a paragraph mark (§) or, if in a field, with a return. SMTPit will validate the email addresses you give it to see if they "look" like email addresses. If SMTPit cannot find a valid email address, it will return an error.

SMTPit considers all of the following to be "valid" email addresses:

smtpit@comm-unity.net
<smtpit@comm-unity.net>
SMTPit Support <smtpit@comm-unity.net>
"SMTPit Support" <smptit@comm-unity.net>
"Traynham, Jake" <jake@comm-unity.net>
<smtpit@comm-unity.net> (SMTPit Support)

The basic rule is, if you have any "human readable" name in the email address (like "SMTPit Support"), then you need angled brackets (< >) around the actual email address. This is so mail servers and mail clients can correctly identify the email address. If you have a comma in the "human readable" name, then you need to enclose it in double quotes (") (like "Traynham, Jake").

Examples:

```
External("SMTP-ToAssign", "SMTPit Support <smtpit@comm-unity.net>")
External("SMTP-ToAssign", "smtpit@comm-unity.net§""Traynham, Jake" <jake@comm-unity.net>")
External("SMTP-ToAssign", MyEmailAddressesField)
External("SMTP-ToAssign", """" & Contact Name & """" <" & Contact Email & ">")
```

See Also:

SMTP-ToAppend
SMTP-FromAssign
SMTP-CCAssign
SMTP-CCAppend
SMTP-BCCAssign
SMTP-BCCAppend

SMTP-ToAppend

The **SMTP-ToAppend** function will add email addresses to the current list of "To" email addresses. If you have not set an email address with **SMTP-ToAssign**, **SMTP-ToAppend** will assign the current value to the "To" setting. You can use **SMTP-ToAppend** multiple times without overwriting previous settings, unlike the **SMTP-ToAssign** function. You can append multiple email addresses with this function. If you have multiple email addresses and you want to append them all at once, separate each email address with a paragraph mark (§) or, if in a field, with a return. SMTPit will validate the email addresses you give it to see if they "look" like email addresses. If SMTPit cannot find a valid email address, it will return an error. For examples of what SMTPit considers to be a "valid" email address, see the **SMTP-ToAssign** function.

Examples:

```
External("SMTP-ToAppend", "<smtpit@comm-unity.net>")
External("SMTP-ToAppend", "<smtpit@comm-unity.net> (SMTPit Support)§jake@comm-unity.net")
```

See Also:

- SMTP-ToAssign**
- SMTP-FromAssign**
- SMTP-CCAssign**
- SMTP-CCAppend**
- SMTP-BCCAssign**
- SMTP-BCCAppend**

SMTP-FromAssign

Use the **SMTP-FromAssign** function to assign the "From" email address. Note: If you use the **SMTP-FromAssign** function twice, the second one will overwrite the first. If you assign a from email address using this function, SMTPit will ignore the Default From field in the Configuration Dialog. SMTPit will validate the email address you give it to see if it "looks" like an email address. If SMTPit cannot find a valid email address, it will return an error. For examples of what SMTPit considers to be a "valid" email address, see the **SMTP-ToAssign** function.

Examples:

```
External("SMTP-FromAssign", "smtpit@comm-unity.net")
External("SMTP-FromAssign", "SMTPit Support <smtpit@comm-unity.net>")
External("SMTP-FromAssign", Company Name & "Support <" & Company Email & ">")
```

See Also:

- SMTP-ToAssign**
- SMTP-ToAppend**
- SMTP-CCAssign**
- SMTP-CCAppend**
- SMTP-BCCAssign**
- SMTP-BCCAppend**

SMTP-SubjectAssign

Use the **SMTP-SubjectAssign** function to assign the "Subject" of the email. Note: If you use the **SMTP-SubjectAssign** function twice, the second will overwrite the first. If you assign a "Subject" of the email with this function, SMTPit will ignore the Default Subject field in the Configuration Dialog.

Example:

```
External("SMTP-SubjectAssign", "In response to your inquiry")
```

See Also:

SMTP-SubjectAppend

SMTP-SubjectAppend

The **SMTP-SubjectAppend** function will add to the current "Subject" of the email. If you have previously used the **SMTP-SubjectAssign** function, this function will add on to that setting. However, if you have not previously used the **SMTP-SubjectAssign** function, but you have a value in the Default Subject field in the Configuration Dialog, this function will add to the default setting. You can use **SMTP-SubjectAppend** multiple times without overwriting previous assignments, unlike the **SMTP-SubjectAssign** function.

Example:

```
External("SMTP-SubjectAppend", " Order Number: " & OrderID)
```

See Also:

SMTP-SubjectAssign

SMTP-CCAssign

Use the **SMTP-CCAssign** function to assign the "CC" (Carbon Copy) email address(es). Note: If you use the **SMTP-CCAssign** function twice, the second will overwrite the first. You can assign multiple email addresses with this function. If you have multiple email addresses and you want to set them all at once, separate each email address with a paragraph mark (§) or, if in a field, with a return. SMTPit will validate the email addresses you give it to see if they "look" like email addresses. If SMTPit cannot find a valid email address, it will return an error. For examples of what SMTPit considers to be a "valid" email address, see the **SMTP-ToAssign** function.

Examples:

```
External("SMTP-CCAssign", ""Traynham, Jake" <jake@comm-unity.net>")  
External("SMTP-CCAssign", "smtpit@comm-unity.net§info@comm-unity.net")
```

See Also:

SMTP-ToAssign
SMTP-ToAppend
SMTP-FromAssign
SMTP-CCAppend
SMTP-BCCAssign
SMTP-BCCAppend

SMTP-CCAppend

The **SMTP-CCAppend** function will add email addresses to the current list of "CC" email addresses. If you have not set an email address with **SMTP-CCAssign**, **SMTP-CCAppend** will assign the current value to the "CC" setting. You can use **SMTP-CCAppend** multiple times without overwriting previous settings, unlike the **SMTP-CCAssign** function. You can append multiple email addresses with this function. If you have multiple email addresses and you want to append them all at once, separate each email address with a paragraph mark (§) or, if in a field, with a return. SMTPit will validate the email addresses you give it to see if they "look" like email addresses. If SMTPit cannot find a valid email address, it will return an error. For examples of what SMTPit considers to be a "valid" email address, see the **SMTP-ToAssign** function.

Examples:

```
External("SMTP-CCAppend", "smtpit@comm-unity.net")  
External("SMTP-CCAppend", "<jake@comm-unity.net>§SMTPit Support  
<smtpit@comm-unity.net>")
```

See Also:

SMTP-ToAssign
SMTP-ToAppend
SMTP-FromAssign
SMTP-CCAssign
SMTP-BCCAssign
SMTP-BCCAppend

SMTP-BCCAssign

Use the **SMTP-BCCAssign** function to assign the "BCC" (Blind Carbon Copy) email address(es). Note: If you use the **SMTP-BCCAssign** function twice, the second will overwrite the first. Any email addresses you have in the BCC list will have the email sent to them, but their email will not show up in the actual email. You can assign multiple email addresses with this function. If you have multiple email addresses and you want to set them all at once, separate each email address with a paragraph mark (§) or, if in a field, with a return. SMTPit will validate the email addresses you give it to see if they "look" like email addresses. If SMTPit cannot find a valid email address, it will return an error. For examples of what SMTPit considers to be a "valid" email address, see the **SMTP-ToAssign** function.

Examples:

```
External("SMTP-BCCAssign", ""Traynham, Jake" <jake@comm-unity.net>")  
External("SMTP-BCCAssign", "smtpit@comm-unity.net¶info@comm-unity.net")
```

See Also:

- SMTP-ToAssign**
- SMTP-ToAppend**
- SMTP-FromAssign**
- SMTP-CCAssign**
- SMTP-CCAppend**
- SMTP-BCCAppend**

SMTP-BCCAppend

The **SMTP-BCCAppend** function will add email addresses to the current list of "BCC" email addresses. If you have not set an email address with **SMTP-BCCAssign**, **SMTP-BCCAppend** will assign the current value to the "BCC" setting. You can use **SMTP-BCCAppend** multiple times without overwriting previous settings, unlike the **SMTP-BCCAssign** function. You can append multiple email addresses with this function. If you have multiple email addresses and you want to append them all at once, separate each email address with a paragraph mark (¶) or, if in a field, with a return. SMTPit will validate the email addresses you give it to see if they "look" like email addresses. If SMTPit cannot find a valid email address, it will return an error. For examples of what SMTPit considers to be a "valid" email address, see the **SMTP-ToAssign** function.

Examples:

```
External("SMTP-CCAppend", "smtpit@comm-unity.net")  
External("SMTP-CCAppend", "<jake@comm-unity.net>¶SMTPit Support  
<smtpit@comm-unity.net>")
```

See Also:

- SMTP-ToAssign**
- SMTP-ToAppend**
- SMTP-FromAssign**
- SMTP-CCAssign**
- SMTP-CCAppend**
- SMTP-BCCAssign**

SMTP-BodyAssign

Use the **SMTP-BodyAssign** function to assign the Plain Text Body of the email. The "Body" of the email is the content of the email that the receiver will read. Note: If you use the **SMTP-BodyAssign** function twice, the second will overwrite the first.

Examples:

```
External("SMTP-BodyAssign", MyEmailBodyField)
External("SMTP-BodyAssign", "Hi!")
```

See Also:

SMTP-BodyAppend
SMTP-FileBodyAssign
SMTP-FileBodyAppend
SMTP-HeaderAssign
SMTP-HeaderAppend
SMTP-FooterAssign
SMTP-FooterAppend

SMTP-BodyAppend

The **SMTP-BodyAppend** function will add to the Plain Text Body of the email assigned by **SMTP-BodyAssign**. If you have not set a body with **SMTP-BodyAssign**, **SMTP-BodyAppend** will assign the current value to the "Body" of the email. You can use **SMTP-BodyAppend** multiple times without overwriting previous settings, unlike the **SMTP-BodyAssign** function.

Examples:

```
External("SMTP-BodyAppend", portal_relation::Item Name & " " & portal_relation::Item
Cost)
```

See Also:

SMTP-BodyAssign
SMTP-FileBodyAssign
SMTP-FileBodyAppend
SMTP-HeaderAssign
SMTP-HeaderAppend
SMTP-FooterAssign
SMTP-FooterAppend

SMTP-FileBodyAssign

You can use the **SMTP-FileBodyAssign** function to load a text file into the Plain Text Body of the email. Note: If you use **SMTP-BodyAssign** or **SMTP-FileBodyAssign** a second time, it will overwrite the previous setting.

Examples:

```
External("SMTP-FileBodyAssign", "c:\My Documents\email body.txt")
External("SMTP-FileBodyAssign", "Macintosh HD:Documents:email body.txt")
```

See Also:

- SMTP-BodyAssign**
- SMTP-BodyAppend**
- SMTP-FileBodyAppend**
- SMTP-HeaderAssign**
- SMTP-HeaderAppend**
- SMTP-FooterAssign**
- SMTP-FooterAppend**

SMTP-FileBodyAppend

The **SMTP-FileBodyAppend** function will add the contents of a text file to the Plain Text Body of the email. You can use this function several times to add different text files to the Plain Text Body without overwriting any previous settings.

Examples:

```
External("SMTP-FileBodyAppend", "c:\My Documents\email signature.txt")
External("SMTP-FileBodyAppend", "Macintosh HD:Documents:email signature.txt")
```

See Also:

- SMTP-BodyAssign**
- SMTP-BodyAppend**
- SMTP-FileBodyAssign**
- SMTP-HeaderAssign**
- SMTP-HeaderAppend**
- SMTP-FooterAssign**
- SMTP-FooterAppend**

SMTP-HTMLBodyAssign

This function works exactly like the **SMTP-BodyAssign** function, except that it sets the HTML Body of the email instead of the Plain Text Body of the email. Note: If you use this function twice, the second will overwrite the first.

Examples:

```
External("SMTP-HTMLBodyAssign", MyHTMLEmailBodyField)
External("SMTP-HTMLBodyAssign", "<h1>Hi!</h1>")
```

See Also:

- SMTP-HTMLBodyAppend**
- SMTP-HTMLFileBodyAssign**
- SMTP-HTMLFileBodyAppend**
- SMTP-HTMLHeaderAssign**
- SMTP-HTMLHeaderAppend**
- SMTP-HTMLFooterAssign**
- SMTP-HTMLFooterAppend**

SMTP-HTMLBodyAppend

This function works exactly like the **SMTP-BodyAppend** function, except that it appends to the HTML Body of the email instead of the Plain Text Body of the email. You can use this function multiple times to append more HTML text.

Examples:

```
External("SMTP-HTMLBodyAppend", "<tr><td>portal_relation::Item Name & "</td><td>" & portal_relation::Item Cost & "</td></tr>")
```

See Also:

- SMTP-HTMLBodyAssign**
- SMTP-HTMLFileBodyAssign**
- SMTP-HTMLFileBodyAppend**
- SMTP-HTMLHeaderAssign**
- SMTP-HTMLHeaderAppend**
- SMTP-HTMLFooterAssign**
- SMTP-HTMLFooterAppend**

SMTP-HTMLFileBodyAssign

This function works exactly like the **SMTP-FileBodyAssign** function, except that it sets the HTML Body of the email instead of the Plain Text Body of the email. Note: If you use **SMTP-HTMLBodyAssign**, or **SMTP-HTMLFileBodyAssign** a second time, it will overwrite the previous setting.

Examples:

```
External("SMTP-HTMLFileBodyAssign", "c:\My Documents\email body.htm")  
External("SMTP-HTMLFileBodyAssign", "Macintosh HD:Documents:email body.htm")
```

See Also:

- SMTP-HTMLBodyAssign**
- SMTP-HTMLBodyAppend**
- SMTP-HTMLFileBodyAppend**
- SMTP-HTMLHeaderAssign**
- SMTP-HTMLHeaderAppend**
- SMTP-HTMLFooterAssign**
- SMTP-HTMLFooterAppend**

SMTP-HTMLFileBodyAppend

This function works exactly like the **SMTP-FileBodyAppend** function, except that it appends to the HTML Body of the email instead of the Plain Text Body of the email. You can use this function several times to add different HTML files to the HTML Body without overwriting any previous settings.

Examples:

```
External("SMTP-HTMLFileBodyAppend", "c:\My Documents\disclaimer.htm")
External("SMTP-HTMLFileBodyAppend", "Macintosh HD:Documents:disclaimer.htm")
```

See Also:

SMTP-HTMLBodyAssign
SMTP-HTMLBodyAppend
SMTP-HTMLFileBodyAssign
SMTP-HTMLHeaderAssign
SMTP-HTMLHeaderAppend
SMTP-HTMLFooterAssign
SMTP-HTMLFooterAppend

SMTP-Send

The **SMTP-Send** function first checks to make sure you have assigned enough fields to send an email. If everything checks out, it attempts to send the email to your mail server. If there is a problem sending the email to your mail server, it will return the error. If the email sent successfully, it will return the string "Email Sent Successfully." For normal operation, use the empty string as the parameter ("").

The **SMTP-Send** function can now take optional arguments. If you would like a dialog to pop up informing you that the email was sent, use "Dialog=Yes". This is the same as checking the option in the Configuration Dialog, but this allows you to do it on a per email basis. If you would like a transcript of the interaction between SMTPit and your mail server, use "Transcript=Yes". This may be useful when determining why your mail server is not allowing you to send email through it. If you want to specify more than one argument, separate them with a comma.

You can use this function by itself, or with the **SMTP-Connect** and **SMTP-Disconnect** functions. If you use this function by itself, this function will connect to your mail server, send the email, and then disconnect. If you call **SMTP-Connect** before calling this function, this function will simply send the email; it will not try to connect and disconnect. This is useful in mass-email solutions that send out several emails in one setting. Connecting once, sending several emails, and then disconnecting is much more efficient than connecting and disconnecting for each individual email. If you call **SMTP-Connect**, you must call **SMTP-Disconnect**, otherwise SMTPit will never disconnect you from the mail server.

If your mail server requires authentication before you can send email, you must set up the Authentication Type, Username, and Password before calling this function. Otherwise, SMTPit will not have the required information to properly authenticate with your mail server.

Examples:

```
External("SMTP-Send", "")
External("SMTP-Send", "Dialog=Yes, Transcript=Yes")
External("SMTP-Send", "Dialog=" & YesNoField)
```


See Also:

SMTP-HostAssign
SMTP-PortAssign
SMTP-AuthTypeAssign
SMTP-UsernameAssign
SMTP-PasswordAssign
SMTP-Connect
SMTP-Disconnect

SMTP-HeaderAssign

This is a unique function that will allow you to have a "Header" section in the Plain Text Body of your email message. Possible uses are email letterheads or personalized email messages for mass emailing. SMTPit also allows you to set a Default Header in the Configuration Dialog. If you assign a Plain Text Header using **SMTP-HeaderAssign**, SMTPit will ignore the Default Header.

Example:

```
External("SMTP-HeaderAssign", "Dear " & Contact Name & ",!!")
```

See Also:

SMTP-BodyAssign
SMTP-BodyAppend
SMTP-HeaderAppend
SMTP-FooterAssign
SMTP-FooterAppend

SMTP-HeaderAppend

The **SMTP-HeaderAppend** will append text to the current Plain Text Body Header. If you have not previously used the **SMTP-HeaderAssign** function, but you have a Default Header defined in the Configuration Dialog, this function will append to the Default Header.

Example:

```
External("SMTP-HeaderAppend", "-----!!")
```

See Also:

SMTP-BodyAssign
SMTP-BodyAppend
SMTP-HeaderAssign
SMTP-FooterAssign
SMTP-FooterAppend

SMTP-HTMLHeaderAssign

This function works exactly like the **SMTP-HeaderAssign** function, except that it sets the HTML Body Header instead of the Plain Text Body Header. You can also assign a Default HTML Header in

the Configuration Dialog. If you assign an HTML Header using **SMTP-HTMLHeaderAssign**, SMTPit will ignore the Default HTML Header.

Example:

```
External("SMTP-HTMLHeaderAssign", "<p>Dear " & Contact Name & ",</p>")
```

See Also:

- SMTP-HTMLBodyAssign**
- SMTP-HTMLBodyAppend**
- SMTP-HTMLHeaderAppend**
- SMTP-HTMLFooterAssign**
- SMTP-HTMLFooterAppend**

SMTP-HTMLHeaderAppend

This function works exactly like the **SMTP-HeaderAppend** function, except that it appends to the HTML Body Header instead of the Plain Text Body Header. If you have not previously used the **SMTP-HTMLHeaderAssign** function, but you have a Default HTML Header in the Configuration Dialog, this function will append to the Default HTML Header.

Example:

```
External("SMTP-HTMLHeaderAppend", "<p><hr></p>")
```

See Also:

- SMTP-HTMLBodyAssign**
- SMTP-HTMLBodyAppend**
- SMTP-HTMLHeaderAssign**
- SMTP-HTMLFooterAssign**
- SMTP-HTMLFooterAppend**

SMTP-FooterAssign

This is another unique function in SMTPit that will allow you to have a "Footer" section in the Plain Text Body of your email. Possible uses include an email signature, a public key, an email advertisement, quotes, or a passage telling how to unsubscribe from your list. You can set a Default Footer in the Configuration Dialog. If you assign a Plain Text Footer using **SMTP-FooterAssign**, SMTPit will ignore the default footer.

Example:

```
External("SMTP-FooterAssign", "-----¶Jake Traynham¶jake@comm-unity.net")
```

See Also:

SMTP-BodyAssign
SMTP-BodyAppend
SMTP-HeaderAssign
SMTP-HeaderAppend
SMTP-FooterAppend

SMTP-FooterAppend

Like the **SMTP-HeaderAppend**, the **SMTP-FooterAppend** appends text to the current Plain Text Body Footer. If you have not previously used the **SMTP-FooterAssign** function, but you have a Default Footer defined in the Configuration Dialog, this function will append to the Default Footer.

Example:

```
External("SMTP-FooterAppend", Company Disclaimer)
```

See Also:

SMTP-BodyAssign
SMTP-BodyAppend
SMTP-HeaderAssign
SMTP-HeaderAppend
SMTP-FooterAssign

SMTP-HTMLFooterAssign

This function works exactly like the **SMTP-FooterAssign** function, except that it sets the HTML Body Footer instead of the Plain Text Body Footer. You can also assign a Default HTML Footer in the Configuration Dialog. If you assign an HTML Footer using the **SMTP-HTMLFooterAssign**, SMTPit will ignore the Default HTML Footer.

Example:

```
External("SMTP-HTMLFooterAssign", "<p><hr><br>Jake Traynham<br>jake@community.net</p>")
```

See Also:

SMTP-HTMLBodyAssign
SMTP-HTMLBodyAppend
SMTP-HTMLHeaderAssign
SMTP-HTMLHeaderAppend
SMTP-HTMLFooterAppend

SMTP-HTMLFooterAppend

This function works exactly like the **SMTP-FooterAppend** function, except that it appends to the HTML Body Footer instead of the Plain Text Body Footer. If you have not previously used the

SMTP-HTMLFooterAssign function, but you have a Default HTML Footer in the Configuration Dialog, this function will append to the Default HTML Footer.

Example:

```
External("SMTP-HTMLFooterAppend", "<p>Visit my <a href=""http://  
www.mydomain.com/">website</a>!"</p>")
```

See Also:

- SMTP-HTMLBodyAssign**
- SMTP-HTMLBodyAppend**
- SMTP-HTMLHeaderAssign**
- SMTP-HTMLHeaderAppend**
- SMTP-HTMLFooterAssign**

SMTP-PathToDBAcquire

If you need to store your attachments in the same folder as your database, either for convenience or in your bound solution, and you are not sure if that location will stay the same, then you need a way to find the location of your database on the hard drive. This function will return to you the path to the folder that contains your database. You can then add the name of your file to the end of this path for use with the Attachment and HTML Image functions. You must provide the name of the database as the parameter.

Example:

```
External("SMTP-PathToDBAcquire", Status(CurrentFileName))
```

See Also:

- SMTP-AttachAssign**
- SMTP-AttachAppend**
- SMTP-HTMLImageAssign**
- SMTP-HTMLImageAppend**
- SMTP-FileNameAcquire**

SMTP-AttachAssign

The **SMTP-AttachAssign** function allows you to attach a file from your hard drive or network to the current email being sent. You must specify the full path and filename of the file. You can assign more than one file with this function by separating the file names with the paragraph mark (¶) or, if in a field, with a return. Note: If you use the **SMTP-AttachAssign** function twice, the second one will overwrite the first. If you have attachments assigned in the Configuration Dialog, they will always be sent with your email.

Examples:

```
External("SMTP-AttachAssign", "c:\My Documents\resume.doc")
External("SMTP-AttachAssign", "Macintosh HD:Documents:resume.doc")
External("SMTP-AttachAssign", "c:\pdfs\Manual.pdf¶c:\pdfs\How to Install.pdf")
External("SMTP-AttachAssign", "Macintosh HD:PDFs:Manual.pdf¶Macintosh
HD:PDFs:How to Install.pdf")
External("SMTP-AttachAssign", MyEmailAttachmentsField)
```

See Also:

SMTP-PathToDBAcquire
SMTP-DlgAttachAssign
SMTP-AttachAppend
SMTP-FileNameAcquire

SMTP-DlgAttachAssign

This function presents you with a standard open file dialog that lets you choose a file from your hard drive. It then attaches the chosen file to the current email. The parameter should be the empty string ("").

Example:

```
External("SMTP-DlgAttachAssign", "")
```

See Also:

SMTP-AttachAssign
SMTP-DlgAttachAppend

SMTP-AttachAppend

The **SMTP-AttachAppend** function works exactly like the **SMTP-AttachAssign** function except that it adds attachments to your current list of attachments instead of overwriting them. You can append more than one file with this function by separating the file names with the paragraph mark (¶) or, if in a field, with a return.

Examples:

```
External("SMTP-AttachAppend", "c:\My Documents\resume.doc")
External("SMTP-AttachAppend", "Macintosh HD:Documents:resume.doc")
External("SMTP-AttachAppend", "c:\pdfs\Manual.pdf¶c:\pdfs\How to Install.pdf")
External("SMTP-AttachAppend", "Macintosh HD:PDFs:Manual.pdf¶Macintosh
HD:PDFs:How to Install.pdf")
```

See Also:

SMTP-PathToDBAcquire
SMTP-AttachAssign
SMTP-DlgAttachAppend
SMTP-FileNameAcquire

SMTP-DlgAttachAppend

The **SMTP-DlgAttachAppend** function works exactly like the **SMTP-DlgAttachAssign** function except that it adds attachments to your current list of attachments. The parameter should be the empty string ("").

Example:

```
External("SMTP-DlgAttachAppend", "")
```

See Also:

SMTP-DlgAttachAssign
SMTP-AttachAppend

SMTP-HTMLImageAssign

The **SMTP-HTMLImageAssign** function is similar to the **SMTP-AttachAssign** function. This function allows you to attach a graphic image file from your hard drive or network to the current email being sent. SMTPit takes this graphic and makes it an inline graphic image for you to use in your HTML emails. You must specify the full path and filename of the image. You can assign more than one file with this function by separating the file names with the paragraph mark (§) or, if in a field, with a return. If you have HTML Images assigned in the Configuration Dialog, they will always be sent with your email, and they will be added to the email first. For more information on inline HTML images, see the **HTML Image Example Explained** section earlier in this documentation. If you do not assign an HTML Body, then an HTML email will not be sent and these HTML Images will be ignored.

Examples:

```
External("SMTP-HTMLImageAssign", "c:\Images\logo.gif")
External("SMTP-HTMLImageAssign", "Macintosh HD:Images:logo.gif")
External("SMTP-HTMLImageAssign", "c:\thumbnails\product1.jpg¶
c:\thumbnails\product2.jpg")
External("SMTP-HTMLImageAssign", "Macintosh HD:Thumbnails:product1.jpg¶
Macintosh HD:Thumbnails:product2.jpg")
External("SMTP-HTMLImageAssign", MyHTMLImagesField)
```

See Also:

SMTP-PathToDBAcquire
SMTP-DlgHTMLImageAssign
SMTP-HTMLImageAppend
SMTP-FileNameAcquire

SMTP-DlgHTMLImageAssign

The **SMTP-DlgHTMLImageAssign** function is similar to the **SMTP-DlgAttachAssign** function. This function presents you with a standard open file dialog that lets you choose a graphic image file

from your hard drive. It then uses the chosen graphic as an inline graphic image for you to use in your HTML emails. For more information on inline HTML images, see the **HTML Image Example Explained** section earlier in this documentation. The parameter should be the empty string ("").

Example:

```
External("SMTP-DlgHTMLImageAssign", "")
```

See Also:

SMTP-HTMLImageAssign

SMTP-DlgHTMLImageAppend

SMTP-HTMLImageAppend

The **SMTP-HTMLImageAppend** function is similar to the **SMTP-AttachAppend** function. This function will add more graphic images files to your current list of HTML Images. You can append more than one file with this function by separating the file names with the paragraph mark (¶) or, if in a field, with a return. For more information on inline HTML images, see the **HTML Image Example Explained** section earlier in this documentation.

Examples:

```
External("SMTP-HTMLImageAssign", "c:\Images\logo.gif")
```

```
External("SMTP-HTMLImageAssign", "Macintosh HD:Images:logo.gif")
```

```
External("SMTP-HTMLImageAssign", "c:\thumbnails\product1.jpg¶  
c:\thumbnails\product2.jpg")
```

```
External("SMTP-HTMLImageAssign", "Macintosh HD:Thumbnails:product1.jpg¶  
Macintosh HD:Thumbnails:product2.jpg")
```

See Also:

SMTP-PathToDBAcquire

SMTP-HTMLImageAssign

SMTP-DlgHTMLImageAppend

SMTP-FileNameAcquire

SMTP-DlgHTMLImageAppend

The **SMTP-DlgHTMLImageAppend** function is similar to the **SMTP-DlgAttachAppend** function. This function presents you with a standard open file dialog that lets you choose a graphic image file on your hard drive. It then adds it to your current list of HTML Images. For more information on inline HTML images, see the **HTML Image Example Explained** section earlier in this documentation. The parameter should be the empty string ("").

Example:

```
External("SMTP-DlgHTMLImageAppend", "")
```

See Also:

SMTP-DlgHTMLImageAssign
SMTP-HTMLImageAppend

SMTP-EmailHeaderAppend

The **SMTP-EmailHeaderAppend** function adds extra email headers to your email messages. Note that there should be no spaces in the name section of the header, however the value can have spaces. This function looks like the following:

```
External("SMTP-EmailHeaderAppend", "name=email header value")
```

The first part is the name and the second part is the value corresponding to the name part. Use an equal sign to separate them. Note: You can also set Default Email Headers in the Configuration Dialog. If you have Email Headers assigned in the Configuration Dialog, they will be appended to the Header of the email before any Email Headers you set with this function. You cannot set any of the following with this function: To, From, Subject, Date, Message-ID, CC, or BCC.

You can assign multiple email headers with this function. If you have multiple email headers and you want to set them all at once, separate each email header with a paragraph mark (§) or, if in a field, with a return.

Examples:

```
External("SMTP-EmailHeaderAppend", "X-Company=My Company")
External("SMTP-EmailHeaderAppend", "X-RecID=" & RecordID)
External("SMTP-EmailHeaderAppend", "X-Company=" & Company Name & "§X-URL="
& Company URL)
External("SMTP-EmailHeaderAppend", MyEmailHeadersField)
```

SMTP-FileNameAcquire

This function allows you to use a standard open file dialog to choose a file, and then returns the full path and filename of the selected file. Note: This function only returns a path to a file to you. It does nothing with the actual file. It does not use the selected file as an attachment or HTML image. The parameter should be the empty string ("").

Example:

```
External("SMTP-FileNameAcquire", "")
```

See Also:

SMTP-PathToDBAcquire
SMTP-AttachAssign
SMTP-AttachAppend
SMTP-HTMLImageAssign
SMTP-HTMLImageAppend

SMTP-PriorityAssign

This function allows you to set the "Priority" of an email. Most email clients understand the Priority setting on an email and will highlight the email as more or less important than other emails. This function will take the numbers 1 through 5 with 1 being the Highest Priority, and 5 being the Lowest. This function will also take one of the following five words: "Highest", "High", "Normal", "Low", or "Lowest". The Priority defaults to "Normal".

Examples:

```
External("SMTP-PriorityAssign", "1")  
External("SMTP-PriorityAssign", "High")  
External("SMTP-PriorityAssign", Email Priority)
```

SMTP-Connect *

SMTP-Connect is a new function for version 3.0. This function allows you to manually connect to your mail server instead of letting the **SMTP-Send** function connect for you. When you use the **SMTP-Send** function by itself, it will connect to your mail server, send your email, and then disconnect. If you call **SMTP-Connect** before calling **SMTP-Send**, then **SMTP-Send** will only send your email; it will not try to connect or disconnect. This is useful in mass-email solutions that need to send out several emails at a time. Connecting once, sending several emails, and then disconnecting is much more efficient than connecting and disconnecting for each individual email. If you call **SMTP-Connect**, you must call **SMTP-Disconnect**, otherwise SMTPit will never disconnect you from the mail server. In a mass-email solution, you would want to call **SMTP-Connect** before your looping script, and call **SMTP-Disconnect** after your looping script. For normal operation, use the empty string as the parameter ("").

The **SMTP-Connect** function can take optional arguments. If you would like a transcript of the interaction between SMTPit and your mail server, use "Transcript=Yes". This may be useful when determining why your mail server is not allowing you to connect to it. If you want to specify more than one argument, separate them with a comma.

If your mail server requires authentication before you can send email, you must set up the Authentication Type, Username, and Password before calling this function. Otherwise, SMTPit will not have the required information to properly authenticate with your mail server.

Examples:

```
External("SMTP-Connect", "")  
External("SMTP-Connect", "Transcript=Yes")
```

See Also:

SMTP-HostAssign
SMTP-PortAssign
SMTP-AuthTypeAssign
SMTP-UsernameAssign
SMTP-PasswordAssign
SMTP-Send
SMTP-Disconnect

SMTP-Disconnect *

If you use **SMTP-Connect** to connect to your mail server manually (instead of letting **SMTP-Send** do it), then you must call **SMTP-Disconnect** to disconnect from your mail server. Otherwise, SMTPit will never disconnect you from the mail server. The parameter should be the empty string ("").

Example:

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

See Also:

SMTP-Send
SMTP-Connect

Obsolete Functions

SMTP-SendWait

Removed in version 2.0. The **SMTP-SendWait** function confused more of our users than helped, and the function did not do much more than the **SMTP-Send** function, so we decided to remove it. The **SMTP-Send** function now waits until the email has successfully been sent to your mail server before returning to FileMaker Pro. The function name still exists for those who are using **SMTP-SendWait** in their existing scripts, but it simply calls the **SMTP-Send** function.

SMTP-FromNameAssign

Removed in version 3.0. The **SMTP-FromNameAssign** is another function that confused more of our users than helped. It has been around since the first version of Windows SMTPit and was in that version because of the underlying SMTP components that were used at that time. Since then it has just not made that much since. Our users were confused on which one of the functions **SMTP-FromNameAssign** or **SMTP-FromName** got the actual email address, and were confused by the lack of any **SMTP-ToNameAssign** or **SMTP-CCNameAssign** functions.

So, if you were using a previous version of SMTPit and were using the **SMTP-FromNameAssign** function, you will need to modify your scripts. Unfortunately, there is no easy way to make this

function backward compatible, so if you use it, you will get an error response. To change your scripts, simply use the **SMTP-FromAssign** function as described above.

Understanding Error Responses

Every function in SMTPit 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 From Address using **SMTP-FromAssign**, and you attempt to send the email using **SMTP-Send**, the Send function will return the following Error Response:

ERROR: Send: Missing From

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("SMTP-Connect", "")"]
If ["LeftWords(Result, 1) = "ERROR"]
    <inform the user>
Else
    <send an email>
End If
```

Credits

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

Contact Information

Email:	SMTPit@comm-unity.net
SMTPit Website:	http://www.smtpit.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