Mailit 6

FileMaker Plug-in Manual

Dacons

This tutorial assumes that you have elementary knowledge of FileMaker Pro and you know how to use plug-ins and external functions of this database platform.

Please send feedback to info@dacons.net
Website: http://www.dacons.net

This version of the manual was updated on May 17, 2015.

CONTENTS

INTRODUCTION
Mailit 6 Features

What is Mailit 6

Mailit is a powerful FileMaker Pro plug-in that lets you send, receive and manage email with your new or existing FileMaker 10-14 databases.

This email plug-in sets new standards in functionality, stability and ease of use. For you this means full email functionality for FileMaker. No external email application is needed.

Smart programming and a highly user friendly interface as well as detailed open source example files and documentation allow you to customize your FileMaker Pro solution with ease.

Feature List

Mailit was created with the FileMaker developer and user in mind. Its ease of use and reliability are appreciated by thousands of satisfied users worldwide.

The combination of Mailit and FileMaker provides features no other email application offers, it is incredible fast and opens completely new possibilities in email communication. Mailit comes with a great set of example files and easy-to-follow instructions.

Mailit is compatible to the FileMaker Server SSS (Server Side Scripting) and IWP (Instant Web publishing).

Possible Mailit solutions:

1. Manage and categorize customer, client and project emails the powerful FileMaker way. Add email functionality to your existing or new database projects.
2. Create email automation for customer response or support.
3. Send highly customized plain text or HTML mass-mailings including inline images. Mailit allows you to compose your emails from different fields and data sources.

4 Check email information such as sender, subject and size before you download the actual message. Implement SPAM rules to delete certain messages directly on the server and keep your email traffic to an optimum.

5 Connect your FileMaker online solution or your database-driven web site to Mailit to send highly customized and personalized newsletters and notifications.

6 Create email discussion list solutions with the database and administration power of FileMaker

7 Manage and parse information received from online forms with FileMaker.

Find out more about customer solutions at:
http://www.dacons.net/fmplugins/mailit6/customerstories/

Quick Start

This chapter will get you started with Mailit 6 in minutes. All you need is the package you downloaded form the Internet and a supported copy of FileMaker 10-14.

System Requirements for Windows:
FileMaker 10-14 Pro/Advanced/Runtime on the following Windows platforms: Windows XP (SP3)/ Vista/7/8/2003/2008/2012.

System Requirements for Macintosh:
FileMaker 10-14 Pro/Advanced/Runtime on the following Mac platforms: Mac OS X 10.5 or later.

IMAP/POP3/SMTP Account
Note that Mailit needs an IMAP/POP3/SMTP email account.

It will NOT work with web based basic email accounts like Hotmail, AOL, Yahoo, etc.

IMAP protocol implementation provides functionality necessary for retrieving and uploading messages over IMAP, which allows synchronizing folder content, which nevertheless is not a full set of IMAP native functionality.

Installation

1 Make sure that any version of FileMaker is closed.

2 Extract all files of the Mailit package to any location on your machine.

3 Locate the appropriate Mailit plug-in according to your operating system in the extracted package.

4 Copy the plug-in "Mailit.fmplugin" (Mac) or "Mailit.fmx" (Windows) to your FileMaker **Extensions** folder. This folder is located in the same folder as the FileMaker Pro or FileMaker Developer application.

5 Open FileMaker and one of the example files to explore the power of Mailit. To get started with Mailit we recommend you have the complete account data of at least on email account handy.

Example Files

Using the `Personal Mail` solution is a good way to start using Mailit.

The `Simple Sending` and `Simple Receiving` example files show you the basics when it comes to scripting. They have limited functionality. For example, the `Simple Sending` files set only send one email at a time. The "Simple" files are designed to understand the core functionality of the Mailit plug-in.

Reading this manual will save you time and will enable you to create your own powerful email solutions.

Feel free to modify any of the example files to your needs or use them as a basis for your own solution.

Path formats

Mailit 6 requires paths of files and folder to be specified in a certain format. To make life easier, Mailit 6 uses the same path format as FileMaker script steps that operate with files. However, unlike FileMaker the plug-in does not support relative paths.

On Windows, paths to be used with Mailit 6 functions must be formatted according to one of the following patterns:

```
filewin:/driveletter:/directoryName/fileName.extension
filewin://computerName/shareName/fileName.extension
```

On Mac, please use the following pattern to specify a path:

```
filemac:/volumeName/directoryName/fileName.extension
```

When specifying a file path in FileMaker (File Location dialog) you can define alternative files. Paths of alternative files are separated by carriage return. Please note that Mailit 6 does not support alternative path this way. It allows you to specify exactly one path at a time only to avoid ambiguities.

Some Mailit 6 functions allow you to specify a folder path instead of a file path.
Please ensure that folder paths always end with a final slash ("/"). The following patterns refer to folder paths on Windows:

```
filewin:/driveletter:/directoryName/
filewin://computerName/shareName/folderName/
```

On Mac a folder paths has to be formatted as follows:

```
filemac:/volumeName/directoryName/
```

Mailit 6 uses the FileMaker path format not only to specify the location of files and folder. The FileMaker path format is also used when paths are *returned* by the plug-in (i.e. using the function `Emai_GetUserFolder`). FileMaker version 8 and later support script variables that can be used to handle retrieved paths for further processing with FileMaker script steps such as *Insert File* .

CHAPTER 1
Receiving Email

In this chapter a simple script to receive email Mailit and FileMaker is outlined.

As a reference we suggest you open the `Simple Receiving` (IMAP or POP3 edition) file that came with the package you downloaded. This will enable you to follow along.

The `Simple Receiving` example file will enable you to receive email for one account at a time only. As the name suggests this file is kept very simple to show basic functionality.

For a more complete email solution using Mailit please have a look at the `Personal Mail` example file which is part of the Mailit package as well.

This chapter shows how to receive email in three simple steps:

- Fields required to receive email
- Functions required to receive email
- Scripts required to receive email

Fields to Receive Email

The `Simple Receiving POP3` file has the following `"global"` fields. The fields are part of three tables, accessible via File > Manager > Database

1. System (S)
2. Messages (M)
3. Attachments (A)

Field Name	Table	Type	Description
Result	S	Text	Retrieves status results returned by the plug-in
POP Server	S	Text	POP server address for the email account
POP Password	S	Text	Password for your POP account
POP User Name	S	Text	POP user name for the email account
Preferred POP Authentication	S	Text	Preferred authentication mode of the POP server
SSL	S	Text	Toggles secure connection "On" and "Off"
POP Timeout	S	Number	Timeout interval in seconds
POP Port	S	Number	The port number for the POP server; usually "110"
Total Messages	S	Number	Number of new messages waiting on the POP server
Current Message	S	Number	Contains the current message number
Attachments path	S	Text	Global path to attachments folder. In case if not empty, solution will save all attachments to the folder specified
Leave Messages on server	S	Text	Toggles "On" or "Off" to leaves messages on server
Delete Messages from server	S	Text	Toggles "On" or "Off" to delete messages from server after a certain number of days
Delete Messages After	S	Number	The number of days after messages are deleted from server
Total Attachments	S	Number	Contains total number of attachments for current message
Current Attachment	S	Number	Contains the attachment number currently being processed
Path	S	Text	Stores the path for saved attachmentmessages
From	S	Text	Contains the email address of the email account
To	S	Text	Contains the recipient's email address
Cc	S	Text	Contains the Cc email address
Subject	S	Text	Contains the email subject of the email

Field Name	Table	Type	Description
Body	S	Text	Contains the body of the message either as plain text, as HTML code or as rich text (formatted text)
Message Object	S	Container	Original message object, necessary to display a message using Web Viewer
Format	S	Text	Displays the format of the email message
Header	S	Text	Contains the header of the message
Charset	S	Text	Contains the character set used for that message
Priority	S	Text	Contains the priority of the message
Date	S	Date	Contains the date of the message
Time	S	Time	Contains the time of the message
Message ID	S	Text	ID of the current message
Current Message	S	Text	Current message being processed
Attachment ID	S	Text	Contains unique ID for the attachment
Name (Attachment)	S	Text	Contains the name of the attachment file
Object	S	Container	Contains the actual attachment
Icon	S	Container	Contains icon of file type for the attachment
External Path	S	Text	Contains external file path, in case if attachment was stored externally

Functions to Receive Email

The following Mailit functions are needed for a receiving script. A full description of all functions and their parameters can be found in the chapter "External Functions".

Function	Description
Script: Get Mail	
Emai_SetupStatusDialog	Specifies the settings of a email status dialog that can be shown during message transmission
Emai_ShowStatusDialog	Shows or hides the status dialog
Emai_Connect	Establishes a connection to a POP server to receive messages or to a SMTP server to send messages
Emai_PopGetMessageCount	Shows or hides the status dialog
Emai_SetupProgressMax	Specifies the number of messages to define a progress bar maximum for the status dialog
Emai_PopGetMessageSize	Returns the size in bytes of a specified message on server
Emai_PopRetrieveMessage	Uses current POP connection to download a message from server
Sub-Script: Retrieve Message Content	
Emai_PopGetMessageField	Returns a specified header field of the current message for example: To, From, Cc, Subject
Emai_PopGetMessageBody	Returns the first "text" body of the current message. If no "text" body is discovered plug-in converts the first "html" part into text and returns it as result of this function
Emai_PopGetMessageHTML	Returns the HTML code of the current message.
Emai_PopGetMessageHeader	Returns the header of the current message
Emai_PopGetMessageCharset	Returns the character set of the current message
Emai_PopGetMessagePriority	Returns the priority of the current message as text value: "Very High", "High", "Normal", "Low" or "Very Low"
Emai_PopGetRecvDate	Returns the POP server receiving date of current message. The date is returned as text in the following format: DD/MM/YYYY
Emai_PopGetRecvTime	Returns the POP server receiving time of current message
Sub Script: Retrieve Message Attachments	
Emai_PopGetAttachmentCount	Returns the number of attachments for the current message
Emai_PopGetAttachmentName	Returns original file name and extension of an attachment
Emai_PopGetAttachmentSize	Returns size in Bytes of the specified attachment
Emai_PopGetAttachmentIcon	Returns icon of the specified attachment as image
Emai_PopGetAttachment	Returns specified attachment as binary object
Back to Script: Get Mail	
Emai_Disconnect	Disconnects from server
Emai_ShowStatusDialog	Use with parameter "hide" to hide status dialog after finishing download

Scripts to Receive Emails

The following shows the `'Get Mail'` script in detail:
Comments are **blue**
Sub-scripts are **green**

```
# Goto the list view and prevent undesired refreshes
Go to Layout [ "List" (Messages) ]
Freeze Window
# Setup and customize status dialog.
Set Field [ System::Result; Emai_SetupStatusDialog ("Downloading Messages";
  "allowCancelOn"; "Cancel"; "combinedText"; "Server: " & System::POP Server;
  "reset") ]
Set Field [ System::Result; Emai_ShowStatusDialog ("show") ]
# Connect to POP server
Set Field [ System::Result; Emai_Connect ("pop"; System::POP Server;
  System::POP User Name; System::POP Password; System::Preferred POP Authentication;
  If (System::SSL = "on"; "sslOn"; "sslOff"); System::POP Timeout;
  System::POP Port; "") ]
If [ Left(System::Result; 3) = "000" ]
  # Retrieve number of new messages on server}
  Set Field [ System::Total Messages; Emai_PopGetMessageCount ]
  # Pass number of new messages to status dialog to configure progress bar}
  Set Field [ System::Result; Emai_SetupProgressMax (System::Total Messages) ]
  # Retrieve data of every new message}
  Set Field [ System::Current Message; 1 ]
  Loop
    # Check if there are new messages to download
    Exit Loop If [ System::Current Message > System::Total Messages ]
    # Check message against max. size settings
    Go to Layout [ "List" (Messages) ]
    # Retrieve message from POP server
    Set Field [ System::Result; Emai_PopRetrieveMessage
      (System::Current Message)]
    Exit Loop If [ Left(System::Result; 3) ≠ "000" ]
    Go to Layout [ "List" (Messages) ]
    New Record/Request
    Set Field [ Messages::Current Message ID; Messages::Message ID ]
    # Parse message content
    Perform Script ["Retrieve Message Content"]
    # Refresh message list
    Go to Layout [ "List" (Messages) ]
    Go to Record/Request/Page [ First ]
    Refresh Window
    # Increasing message counter
    Set Field [ System::Current Message; System::Current Message + 1 ]
    # Update list content
    Sort Records [Specified Sort Order:Messages::Date;descending Messages::Time;
      descending ] [ Restore; No dialog ]
    Refresh Window
    Freeze Window
  End Loop
Else
  Show Custom Dialog [ Title: "Email Download"; Message: "Connection to the
    POP server could not be established! Check the account data and your
```

```
    internet/intranet connection.¶¶Plug-in result: " & System::Result;
    Buttons:"OK"]
End If
# Disconnect from POP server and hide status dialog
Set Field [ System::Result; Emai_Disconnect ("pop"; Case
  (System::Leave Messages on server = "on" and
  System::Delete Messages from server = "off"; "-1";
  System::Leave Messages on server = "on" and
  System::Delete Messages from server = "on"; System::Delete Messages After;
  System::Leave Messages on server = "off"; "0")) ]
Set Field [ System::Result; Emai_ShowStatusDialog ("hide") ]
```

'Receive Message Content' script (sub-script of 'Get Mail' script):

```
#Retrieve message object
Set Field [ Messages::From; Emai_PopGetMessageSource ]
#Retrieve message contacts
Set Field [ Messages::From; Emai_PopGetMessageField ("from") ]
Set Field [ Messages::To; Emai_PopGetMessageField ("to") ]
Set Field [ Messages::Cc; Emai_PopGetMessageField ("cc") ]
#Retrieve all other message content items
Set Field [ Messages::Subject; Emai_PopGetMessageField ("Subject") ]
Set Field [ Messages::Body; Emai_PopGetMessageBody ]
#Check if message contains HTML code
Set Field [ System::Result; Emai_PopGetMessageHTML ]
If [ Messages::Body HTML ≠ "" ]
  Set Field [ Messages::Format; "Rich" ]
Else
  Set Field [ Messages::Format; "Text" ]
End If
Set Field [ Messages::Header; Emai_PopGetMessageHeader ]
Set Field [ Messages::Charset; Emai_PopGetMessageCharset ]
Set Field [ Messages::Priority; Emai_PopGetMessagePriority ]
Set Field [ Messages::Date; Date (Middle (Emai_PopGetRecvDate; 4; 2);
  Middle (Emai_PopGetRecvDate; 1; 2); Middle (Emai_PopGetRecvDate; 7; 4)) ]
Set Field [ Messages::Time; Time (Left (Emai_PopGetRecvTime; 2);
  Middle (Emai_PopGetRecvTime; 4; 2); Middle (Emai_PopGetRecvTime; 7; 2)) ]
#Retrieve attachments
Perform Script [ "Retrieve Message Attachments" ]
```

'Retrieve Attachments' script (sub-script of 'Receive Message Content' script)

```
#Only attachments of type "attachment" are processed. Attachments of type "inline"
  are not stored as file in the database.
Set Field [ System::Total Attachments; Emai_PopGetAttachmentCount ("all") ]
Set Field [ System::Current Attachment; 1 ]
Loop
  Exit Loop If [ System::Current Attachment > System::Total Attachments ]
  Go to Layout [ "Attachments" (Attachments) ]
  New Record/Request
  #Retrieve attachment information
  Set Field [ Attachments::Message ID; Messages::Current Message ID ]
  Set Field [ Attachments::Name; Emai_PopGetAttachmentName
    (System::Current Attachment; "all") ]
  Set Field [ Attachments::Icon; Emai_PopGetAttachmentIcon
    (System::Current Attachment; "all"; 16) ]
  If [ IsEmpty (System::Attachments path) ]
    #Parse attachment content object
    Set Field [ Attachments::Object; Emai_PopGetAttachment
```

```
    (System::Current Attachment; "all") ]
  Else
    Set Field [ Attachments::External Path; Emai_PopExportAttachment
      (System::Current Attachment; System::Attachments path;
      "attachment") ]
    If [IsEmpty (Attachments::External Path) ]
      #The file could not be saved using external path, therefore we will
        save it in the database
      Set Field [ Attachments::Object; Emai_PopGetAttachment
        (System::Current Attachment; "all") ]
    End If
  End If
  Set Field [ System::Current Attachment; System::Current Attachment + 1 ]
End Loop
```

The `'Simple Receiving POP3'` example file contains more scripts. Please refer to the file for more details. Under FileMaker Developer/Advanced you can create a report for this file using Tools>Database Design Report.

Scripts to Receive Email in a Thread

Mailit 6 introduces an additional feature to receive messages in a thread, which is running simultaneously to the FileMaker application main process. This allows the user to make the FileMaker application accessible for other operations.

The following shows the `'Get Mail'` script of the `'Simple Receiving POP3 Threaded'` example file, which illustrates message sending in a thread:

Comments are **blue**

```
# Prevent multiple download sessions
If [ $$download_in_process = "yes" ]
  Exit Script [ ]
End If
Set Variable [ $layout; Value:Get(LayoutName) ]
# Goto the list view and prevent undesired refreshes Freeze Window
Go to Layout [ "Account" (System) ]
#Create Thread
Set Variable [ $threadID; Value:Emai_CreateThread ]
# Connect to POP server
Set Variable [ $connectionID; Value:Emai_Connect( "pop"; System::POP Server;
  System::POP User Name; System::POP Password; System::Preferred POP Authentication;
  If (System::SSL = "on"; "sslOn"; "sslOff"); System::POP Timeout; System::POP Port;
  System::Download History; $threadID )]
Set Field [ System::Result; Emai_GetLastResult( $threadID; "Emai_Connect" ) ]
If [ $connectionID ≠ "" ]
  # Download all messages
  Set Variable [ $$download_in_process; Value:"yes" ]
  Set Variable [ $result;
    Value:Emai_PopRetrieveAllMessages( $connectionID ) ]
  If [ Left($result; 3) ≠ "000" ]
    Show Custom Dialog [ Title: "Error"; Message: "Error downloading messages:¶ "
      & $result; Buttons: "OK" ]
    Perform Script [ "Halt script"; Parameter: $layout ]
  End If
```

```
  # Disconnect from POP server
  Set Variable [ $result; Value:Emai_Disconnect ("pop"; Case (
    System::Leave Messages on server = "on" and
    System::Delete Messages from server = "off"; "-1";
    System::Leave Messages on server = "on" and
    System::Delete Messages from server = "on";
    System::Delete Messages After;
    System::Leave Messages on server = "off"; "0" ); $connectionID )]
Else
  Show Custom Dialog [ Title: "Email Download";
    Message: "Connection to the POP server could not be established!
    Check the account data and your internet/intranet connection.¶¶
    Plug-in result: " & System::Result; Buttons: "OK" ]
End If
Set Field [ System::Result; Emai_PerformScript ($threadID; Get (FileName);
  "Process Downloaded Messages"; $connectionID & "¶" & $threadID) ]
Go to Layout [ original layout ]
```

`'Process Downloaded Messages'` script, scheduled to execute when the receiving thread is complete (scheduled by `'Get Mail'` script):

```
Set Variable [ $$download_in_process; Value:"" ]
Set Variable [ $layout; Value:Get ( LayoutName ) ]
#Pull thread information from script parameter
Set Variable [ $connectionID;
  Value:Substitute (MiddleValues (Get (ScriptParameter); 1; 1);"¶";"") ]
Set Variable [ $threadID;
  Value:Substitute (MiddleValues (Get (ScriptParameter); 2; 1);"¶";"") ]
Go to Layout [ "Account" (System) ]
Set Field [ System::Download History; Emai_PopGetHistory( $connectionID ) ]
# Check result
Set Variable [ $result; Value:Emai_GetLastResult ($threadID;
  "Emai_PopRetrieveAllMessages") ]
If [ Left($result; 3) = "000" ]
  # Retrieve number of new messages
  Go to Layout [ "List" (Messages) ]
  Set Variable [ $total;
    Value:Emai_PopGetMessageCount(""; $connectionID) ]
  Set Variable [ $$progress_total; Value:$total ]
  # Goto the list view and prevent undesired refreshes
  If [ $total > 0 ]
  Set Variable [ $current; Value:1 ]
    Loop
      # Check if there are new messages to download
      Exit Loop If [ $current > $total ]
      # Check message against max. size settings
      Go to Layout [ "List" (Messages) ]
      # Retrieve message from POP server
      New Record/Request
      Set Field [ Messages::Message Object;
        Emai_PopGetMessageSource( $current; $connectionID )]
      Set Variable [ $result;
        Value:Emai_ProcessMessageSource( Messages::Message Object )]
      Exit Loop If [ Left($result; 3) ≠ "000" ]
      Go to Layout [ "List" (Messages) ]
      Set Field [ Messages::Current Message ID; Messages::Message ID ]
      # Parse message content
      Perform Script [ "Retrieve Message Content" ]
      # Refresh message list
```

```
      Go to Layout [ "List" (Messages) ]
      Go to Record/Request/Page [ First ]
      Refresh Window
      # Increasing message counter
      Set Variable [ $current; Value:$current + 1 ]
      Set Variable [ $$progress_done;
        Value:If ($current < $total; $current + 1;
        $total) ]
      # Update list content
      Sort Records [ Specified Sort Order: Messages::Date;
        descending Messages::Time; descending ][ Restore; No dialog ]
      Refresh Window
      Freeze Window
    End Loop
  End If
Else
  Show Custom Dialog [ Title: "Error"; Message: "Error downloading messages:¶ "
    & $result; Buttons: "OK" ]
End If
Set Variable [ $result; Value:Emai_DestroyThread( $threadID )]
Set Variable [ $$progress_total; Value:0 ]
Set Variable [ $$progress_done; Value:0 ]
```

CHAPTER 2
Sending Email

This chapter outlines a simple *sending* script to send email with FileMaker using Mailit 6. As a reference we suggest you open the `Simple Sending` file that came with the package you downloaded. This will enable you to follow along.
For a more complete email solution using Mailit please have a look at the `Personal Mail` example file which part of the Mailit package as well.
This chapter shows how to send email in three simple steps:

- Fields required to send email
- Functions required to send email
- Scripts required to send email

Fields to Send Email

The `Simple Sending` file has the following "global" fields. The fields are part of three tables, accessible via File > Manage > Database

1. System (S)
2. Messages (M)
3. Attachments (A):

Field Name	Table	Type	Description
Result	S	Text	Retrieves status results returned by the plug-in
Email Address	S	Text	Contains sender's email address
Sender Name	S	Text	Name of the sender
SMTP Server	S	Text	Contains the server address of email account

Field Name	Table	Type	Description
Smtp Port	S	Number	Contains the SMTP port (normally: 25)
Preferred Authentication Type	S	Text	Contains the preferred SMTP authentication
SSL	S	Text	Toggles secure connection "On" and "Off"
Smtp User name	S	Text	Contains the SMTP user name of the email account
Smtp Password	S	Text	Contains the SMTP password of the email account
Smtp Timeout	S	Number	Timeout interval in seconds for SMTP
Message ID	M	Text	Unique ID for message
Current Message ID	M	Text	ID of the current message being processed
To	M	Text	Contains the recipient's email address
Cc	M	Text	Carbon Copy secondary email addressee
Bcc		Text	Blind carbon copy email addressee
Priority	M	Text	Contains the priority of the message
Subject	M	Text	Contains the email subject of the email
Format	M	Text	Format of the email (Plain, Rich, Inline or HTML)
Message	M	Text	Email message (Plain, Rich, Inline or HTML)
Sent	M	Date	Marks message after being successfully sent
Message ID	M	Text	Unique ID for the attachment
Object	M	Container	Contains the attachment file
Name	M	Text	Contains the name of the attachment
Icon	M	Container	Contains icon of file type for the attachment

Functions to Send Email

The following Mailit plug-in functions are needed for a sending script. Please notice that example provided doesn't use a threaded mode. Full description of each function and their parameter can be found in the Chapter "External Functions".

Function	Description
Script: Send	
Emai_SmtpResetMessage	Resets all internal message variables of the plug-in. Use this function in a sending loop before sending the next message
Emai_SetupStatusDialog	Shows or hides the status dialog
Emai_SetupProgressMax	Specifies the number of messages to define a progress bar maximum for the status dialog
Emai_ShowStatusDialog	Specifies the settings of the email status dialog that can be shown during message transmission
Emai_SmtpAddMessageField	Sets a header field for outgoing message; covers all header fields including from, sender, reply-to, to, cc, in-reply-to, subject, etc
Emai_SmtpSetMessagePriority	Set the priority of the outgoing message
Emai_SmtpAddBody	Adds a message body part for the next outgoing email
Emai_SmtpAddAttachment	Adds an attachment to the current message
Emai_Connect	Establishes a connection to the SMTP server to send messages or to a POP server to receive messages
Emai_SmtpSendMessage	Sends a message using the current SMTP connection. This function can send a pre-composed message (message source object) or it composes a message from the current message variables that have been set
Emai_Disconnect	Closes the connection to SMTP server
Emai_ShowStatusDialog	Hides the status dialog.

Scripts to Send Email

The following shows the `'Send'` script in detail:
Comments are **blue**

```
#Reset record sending flag
Set Field [ Messages::Sent; "" ]
#Reset plug-in message variables
Set Field [ System::Result; Emai_SmtpResetMessage ("resetAttachments"; "resetBodies"; "
    resetHeader") ]
#Setup status dialog
Set Field [ System::Result; Emai_SetupStatusDialog ("Sending Message"; "allowCancelOn";
     "Cancel"; "autoText"; ""; "reset") ]
Set Field [ System::Result; Emai_SetupProgressMax (1) ]
Set Field [ System::Result; Emai_ShowStatusDialog ("show") ]
#Set "From" name and address
Set Field [ System::Result; Emai_SmtpAddMessageField ("From" ;"\"" & System::Sender
    Name & "\"" & "<" & System::Email Address & ">") ]
#Set "To" name and address(es)
Set Field [ System::Result; Emai_SmtpAddMessageField ("To" ; Messages::To) ]
#Set "Cc" name and address(es)
Set Field [ System::Result; Emai_SmtpAddMessageField ("Cc" ; Messages::Cc) ]
#Set "Bcc" name and address(es)
Set Field [ System::Result; Emai_SmtpAddMessageField ("Bcc" ; Messages::Bcc) ]
#Set priority
Set Field [ System::Result; Emai_SmtpSetMessagePriority (Messages::Priority) ]
#Set subject
Set Field [ System::Result; Emai_SmtpAddMessageField ("Subject"; Messages::Subject) ]
#Set message body
If [ Messages::Format = "Text" ]
  Set Field [ System::Result; Emai_SmtpAddBody ("text"; Messages::Message; "")]
Else If [ Messages::Format = "HTML Code" ]
  Set Field [ System::Result; Emai_SmtpAddBody ("html"; Messages::Message; "")]
Else If [ Messages::Format = "Rich" ]
  Set Field [ System::Result; Emai_SmtpAddBody ("rich"; Messages::Message; "")]
End If
#Set attachments
Set Field [ Messages::Current Message ID; Messages::Message ID ]
Go to Layout [ "Attachments" (Attachments) ]
Set Error Capture [ On ]
Enter Find Mode [ ]
Set Field [ Attachments::Message ID; Messages::Current Message ID ]
Perform Find [ ]
If [ Get(FoundCount) > 0 ]
  Loop
  Set Field [ System::Result; Emai_SmtpAddAttachment (Attachments::Object; "attachment"
      ; Attachments::Name; "once"; "") ]
  Go to Record/Request/Page [ Next; Exit after last ]
  End Loop
End If
Go to Layout [ original layout ]
#Compose and store message object
Go to Layout [ "List" (Messages) ]
#Connect to SMTP server
Set Field [ System::Result; Emai_Connect ("smtp"; System::Smtp Server; System::Smtp
    User Name; System::Smtp Password; System::Preferred Authentication Type; If (System
```

```
    ::SSL = "on"; "sslOn"; "sslOff"); System::Smtp Timeout; System::Smtp Port; "") ]
If [ Left (System::Result; 3) = "000" ]
  #Send message to SMTP server
  Set Field [ System::Result; Emai_SmtpSendMessage (""; "CancelOnRcptErr") ]
  #Change message attributes if sending succeeded
  If [ Left(System::Result; 3) = "000" ]
    Set Field [ Messages::Sent; "Sent" ]
  Else
    If [ Left(System::Result; 4)  "-005" // User clicked Cancel button ]
      #Show error message if sending failed due to other reason than user clicked
          Cancel button
      Show Custom Dialog [ Title: "Send Message"; Message: "The message '" & Messages::
          Subject & "' could not be sent. Check all message values.  ¶¶Plugin result: "
           & System::Result; Buttons: "OK" ]
    End If
  End If
  #Disconnect from SMTP
  Set Field [ System::Result; Emai_Disconnect("smtp"; "") ]
Else
  #Show error message if connection failed
  Show Custom Dialog [ Title: "Send Message"; Message: "Connection to SMTP server
      failed. Message could not be sent.¶¶Plug-in result: " & System::Result; Buttons:
      "OK" ]
End If
#Hide status dialog
Set Field [ System::Result; Emai_ShowStatusDialog ("hide") ]
```

The `'Simple Sending'` example file contains more scripts. Details can be found in the example file. Under FileMaker Developer/Advanced you can create a report for this file using Tools > Database Design Report.

Scripts to Send Email in a Thread

Mailit 6 introduces an additional feature to send messages in a thread, which is running simultaneously to the FileMaker application main process. This allows the user to make the FileMaker application accessible for other operations.

The following shows the `'Send'` script of the `'Simple Sending Threaded'` example file, which illustrates message sending in a thread:

Comments are **blue**

```
#Reset record sending flag
Set Field [ Messages::Sent; "" ]
#Reset plug-in message variables
Set Field [ System::Result; Emai_SmtpResetMessage ("keepNone") ]
#Create Thread
Set Variable [ $threadID; Emai_CreateThread ]
#Setup status dialog
Set Field [ System::Result; Emai_SetupStatusDialog ("Sending Message"; "allowCancelOn";
  "Cancel"; "autoText"; ""; "reset"; $threadID) ]
Set Field [ System::Result; Emai_SetupProgressMax ("1"; $threadID) ]
Set Field [ System::Result; Emai_ShowStatusDialog ("show"; $threadID) ]
#Create Message
Set Variable [ $messageID; Emai_SmtpCreateMessage ]
```

```
#Set "From" name and address
Set Field [ System::Result; Emai_SmtpAddMessageField ("From" ;"\"" &
  System::Sender Name & "\"" & "<" & System::Email Address & ">";
  $messageID) ]
#Set "To" name and address(es)
Set Field [ System::Result; Emai_SmtpAddMessageField ("To"; Messages::To;
  $messageID) ]
#Set "Cc" name and address(es)
Set Field [ System::Result; Emai_SmtpAddMessageField ("Cc"; Messages::Cc;
  $messageID) ]
#Set "Bcc" name and address(es)
Set Field [ System::Result; Emai_SmtpAddMessageField ("Bcc"; Messages::Bcc;
  $messageID) ]
#Set priority
Set Field [ System::Result; Emai_SmtpSetMessagePriority (Messages::Priority;
  $messageID) ]
#Set subject
Set Field [ System::Result; Emai_SmtpAddMessageField ("Subject"; Messages::Subject;
  $messageID) ]
#Set message body
If [ Messages::Format = "Text" ]
  Set Field [ System::Result; Emai_SmtpAddBody ("text"; Messages::Message; "";
    $messageID) ]
Else If [ Messages::Format = "HTML Code" ]
  Set Field [ System::Result; Emai_SmtpAddBody ("html"; Messages::Message; "";
    $messageID) ]
Else If [ Messages::Format = "Rich" ]
  Set Field [ System::Result; Emai_SmtpAddBody ("rich"; Messages::Message; "";
    $messageID) ]
End If
#Set attachments
Set Field [ Messages::Current Message ID; Messages::Message ID ]
Go to Layout [ "Attachments" (Attachments) ]
Set Error Capture [ On ]
Enter Find Mode [ ]
Set Field [ Attachments::Message ID; Messages::Current Message ID ]
Perform Find [ ]
If [ Get(FoundCount) > 0 ]
  Loop
  Set Field [ System::Result; Emai_SmtpAddAttachment (Attachments::Object;
    "attachment"; Attachments::Name; "once"; ""; $messageID) ]
  Go to Record/Request/Page [ Next; Exit after last ]
  End Loop
End If
Go to Layout [ original layout ]
#Compose and store message object
Go to Layout [ "List" (Messages) ]
#Connect to SMTP server
Set Variable [ $connectionID; Emai_Connect ("smtp"; System::Smtp Server;
  System::Smtp User Name; System::Smtp Password; System::Preferred Authentication Type;
  If (System::SSL = "on"; "sslOn"; "sslOff"); System::Smtp Timeout;
  System::Smtp Port; ""; $threadID) ]
Set Field [ System::Result; Emai_GetLastResult ($threadID;
  "Emai_Connect") ]
#Send message to SMTP server
Set Field [ System::Result; Emai_SmtpSendMessage (""; "CancelOnRcptErr";
  $connectionID; $messageID) ]
#Disconnection from SMTP
Set Field [ System::Result; Emai_Disconnect ("smtp"; "";
```

```
  $connectionID) ]
#Destroy the Message object
Set Field [ System::Result; Emai_SmtpDestroyMessage ($messageID) ]
#Hide status dialog
Set Field [ System::Result; Emai_ShowStatusDialog ("hide"; $threadID;
  "queue")]
#Schedule a script, to be executed when sending is complete
Set Field [ System::Result; Emai_PerformScript ($threadID; Get (FileName);
  "Sending Complete"; $threadID & "¶" & Messages::Message ID)]
```

`'Sending Complete'` script, scheduled to execute when the sending thread is complete (scheduled by `'Send'` script):

```
#Pull thread information from script parameter
Set Variable [ $threadID; Substitute
  (MiddleValues (Get (ScriptParameter); 1; 1);"¶";"") ]
Set Variable [ $messageID; Substitute
  (MiddleValues (Get (ScriptParameter); 2; 1);"¶";"") ]
#Find the target message
Go to Layout [ "List" (Messages) ]
Enter Find Mode [ ]
Set Field [ Messages::Message ID; "==" & $messageID ]
Perform Find [ ]
If [ Get (FoundCount) > 0 and Get (LastError) $\neq{}$ 401 ]
  #Verify sending status
  Set Field [ System::Result; Emai_GetLastResult ($threadID;
    "Emai_SmtpSendMessage") ]
  If [ Left (System::Result; 3) = "000" ]
    Set Field [ Messages::Sent; "Sent" ]
  Else
    If [ Left(System::Result; 4) ≠ "-005" ]
      #Show error message if sending failed due to other reason than user
        clicked Cancel button
      Show Custom Dialog [ Title: "Send Message"; Message: "The message '"
        & Messages::Subject & "' could not be sent. Check all message
        values ¶¶Plugin result: " & System::Result; Buttons: "OK" ]
    End If
  End If
End If
Show All Records
#Destroy Thread
Set Field [ System::Result; Emai_DestroyThread ($threadID) ]
```

CHAPTER 3
External Functions

This chapter gives you a general introduction on how to use Mailit 6 plug-in functions with FileMaker.

It explains where you find the plug-in functions in FileMaker and how to trigger them. The next chapter provides in-depth information about all Mailit 6 functions and their parameters.

Accessing external functions

Since Mailit 6 is a FileMaker plug-in, so-called external functions are used to trigger the plug-in from a FileMaker database solution. They are called external functions because these functions are provided by a plug-in and thus, they are not part of the actual FileMaker application. In order to use the external functions provided by a plug-in it must always be installed and enabled in the FileMaker application preferences.

To access external functions provided by installed plug-ins the FileMaker calculation editor is used. FileMaker shows this dialog whenever a calculation has to be defined (i.e. for calculated fields, validation calculations etc). In most cases you will trigger Mailit 6 functions from a script. The easiest way to perform an external function call from a script is to add the script step *Set Field* to your script.

Next, you specify the target field, which will contain the result of the operation. The result of Mailit 6 operations is a simple result code which tells you if a function could be executed successfully or if errors occurred. Therefore, you should setup a global text field in your solution and call it *Result*. Specify the field *Result* as target field of the calculation.

Click the *Specify* button (the second Specify button in FileMaker 10-14) to open the calculation editor. In the top right corner of the dialog you find a drop-down menu called *View*. Change to the section *External functions*. All Mailit 6 functions are listed in the section *Mailit*. If you have additional plug-ins installed you might have to scroll down to the *Mailit* section of the list. If the Mailit 6 functions do not appear in the list, the plug-in is not installed correctly. Refer to the Installation section of this manual on instructions to install Mailit 6.

Double-click on the Mailit 6 function in the list which you would like to use for this script step operation. As a result the function is copied to the large text box of the calculation editor.

Functions updated in Mailit 6 are marked with asterisk (*).
Functions introduced in Mailit 6 are marked with double asterisk (**).

Emai_GetLastResultCode

Description: Returns the result code of the last action performed by the plug-in.

Server: Supported

Parameters:

{threadUID}	Use this parameter to retrieve a function result that was queued in a thread.
{functionName}	In case if a `{threadUID}` parameter was specified, this parameter should be used to specify which function result code should be requested from the queue. As soon as the `Emai_GetLastResultCode` returns a result from a thread, this result is been deleted from the queue. This way, in order to retrieve multiple results subsequent calls should be used.

Emai_Version

Description: Returns version information about the Mailit version installed.

Server: Supported

Parameters:

{option}	`"product"` , `"version"` , `"platform"` or `"copyright"`
Empty	All values are returned separates by space `" "` .

Emai_Connect

Description: Establishes a connection to an SMTP server to send messages or to an IMAP / POP3 server to receive messages.
For threaded SMTP sessions a unique thread identifier should be specified using the `"threadUID"` parameter. In this case the function shell return a unique connection identifier (`"connectionUID"`), instead of the result code. In order to retrieve the function call result use the `"Emai_GetLastResultCode"` function.

Server: Supported

Parameters:

connectionType	`"pop"` , `"imap"` , or `"smtp"`
server	Server address
{userName}	POP, IMAP, or SMTP user name
{password}	POP, IMAP or SMTP password
{authType}	For POP and IMAP connections this can be: `"login"` , `"apop"` , `"plain"` , `"password"` and `"ntlm"` For SMTP connections this can be: `"login"` , `"plain"` , `"ntlm"` and `"cram-md5"` If no preferred value is indicated the plug-in tries all.
{useSSL}	`"sslOn"` or `"sslOff"` (default); establishes a secure connection to a POP or SMTP server; if secure connection is requested and the server does not support it the connection will not be established.
{timeout}	Default is `"60"` (value in seconds)
{port}	Default for POP is `"110"` , for IMAP is `"143"` , and for SMTP it is `"25"`
{history}	For POP3 connections, this parameter should contain a message history object. In case if this parameter is empty, local history file will be used. For IMAP connections, this parameter can be: `"imapUIDL"` (default), `"imapCustomFlags"` and `"imapStandardFlags"` This option defines the method used to identify the messages that should be downloaded. `"imapCustomFlags"` and `"imapStandardFlags"` are provided for backward compatibility to Mailit 4. The default `"imapUIDL"` flags requires a list of available message UIDs to be supplied to the `Emai_ImapGetMessageCount` function, as a parameter.
{threadUID}	This parameter may optionally contain a thread ID, created by the `Emai_CreateThread` plug-in function, used for multi-threaded sending.

Emai_Disconnect

Description: Closes the connection to an IMAP, POP or SMTP server.

Prerequisites: A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

connectionType	`"imap"`, `"pop"` or `"smtp"`
{deleteMessages}	For POP and IMAP (in Mailit 4 compatibility mode) connections only; specifies the number of days messages are left on server; `"-1"` leaves messages on server forever; `"0"` (default) deletes messages immediately after successful download; other number leaves messages on server for specified number of days and deletes them if they have been downloaded successfully AND are older than the specified number of days.
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multithreaded sending.

Emai_CreateThread

Description: Creates a new thread, and returns a unique thread ID.

Threads can be used to make the FileMaker application accessible for other actions, while the plug-in will keep performing mass-mailings. The FileMaker application should not be closed, as far as this will also close the plug-in including all the threads running.

In order to obtain a function execution result, when using threaded sending, make sure to specify the thread UID when calling the `Emai_GetLastResultCode` function.

Server: Not supported. On Server side use single-threaded functions.

Emai_DestroyThread

Description: Destroys a particular thread, including all the related information, such as result queue.

Prerequisites: A thread should be created using the `Emai_CreateThread` before using this function.

Server: Not supported. On Server side use single-threaded functions.

Parameters:

threadUID — Unique thread ID, generated by `Emai_CreateThread` function.

{queue} — This function call may get queued. Use `"queue"` parameter value for this purpose. In this case the function will get executed only when all previous functions of this thread are complete. Otherwise, the plug-in will destroy the thread immediately, including all the related information, such as result queue.

Emai_GetThreadQueueItemsNumber

Description: Returns the number of items in the thread queue.

Prerequisites: A thread should be created using the `Emai_CreateThread` before using this function.

Server: Not supported. On Server side use single-threaded functions.

Parameters:

threadUID — Unique thread ID, generated by `Emai_CreateThread` function.

Emai_ProcessMessageSource

Description: Processes the message source object of a message that has been retrieved earlier and makes it the current message from which information can be retrieved.

Prerequisites: The source of a message must have been retrieved using `Emai_PopGetMessageSource` before using this function.

Server: Supported

Parameters:

messageObject — Message source object to be processed. To process an external message source object the function `Emai_ImportFile` should be used as parameter.

{charset} — Default character set to be used with this message object. The following values are supported: `UTF-8`, `ISO8859-1` through `ISO8859-16`, `KOI-8R`, `KOI-8U`, `Windows-1250` through `Windows-1258`, `ISO-2022-JP`, `ISO-2022-JP-1` through `ISO-2022-JP-3`, `CSISO-2022-JP`, `EUC-JP`, `ISO-2022-CN`, `ISO-2022-CN-EXT`, `GB2312`, `HZ-GB-2312`, `ISO-2022-KR`, `KSC5601-1987`, `EUC-KR`, `GBK` and `Big5`.

Emai_GetFileName

Description: Returns original file name and extension of the specified external file.

Server: Supported

Parameters: filePath Path of the external file from which the file name is to be extracted.

Emai_GetFileSize

Description: Returns the size in Bytes of a file that is located on disk.

Server: Supported

Parameters: filePath Path of the file including file name and extension.

Emai_GetFileIcon

Description: Returns the icon of the specified file as image.

Server: Supported

Parameters: filePath Path pf the file including file name.

{iconSize} Icon size in pixels, `"16"` (default) or `"32"`

Emai_GetFileIcon

Description: Returns the icon of the specified file as image.

Server: Supported

Parameters: filePath Path pf the file including file name.

{iconSize} Icon size in pixels, `"16"` (default) or `"32"`

Emai_ShowFolderDialog

Description: Shows Choose Folder dialog window and returns selected path. This function can't be used when running the plug-in on FileMaker Server.

Server: Not supported. On Server side will be ignored.

Parameters: {dialogTitle} Defines a custom title of the dialog; default title is empty.

{defaultFolderPath} Default folder which will be shown in dialog

Emai_ShowFileDialog

Description: Shows an Open/Save File dialog window and returns selected file path. This function can't be used when running the plug-in on FileMaker Server.

Server: Not supported. On Server side will be ignored.

Parameters:

dialogMode `"save"` or `"open"`

{dialogTitle} Defines a custom title of the dialog; default title is empty

{defaultFolderPath} Default folder which will be shown in dialog

{defaultFileName} Default file will be selected (`dialogMode="open"`) or default new file name will be suggested (`dialogMode="save"`)

{filter} Sets file filters for dialog; syntax for filter as follows:

```
[Adobe Photoshop; CompuServe GIF] *.psd; *.gif
[Microsoft Word; Microsoft Excel] *.doc; *.xls
[Movie Files] *.mov; *.avi
```

Multiple filters are separated by carriage return. First filter will be default on in the dialog's combobox. Filters without descriptions are supported as well; in this case the description string for each filter will equal the actual filter:

```
*.fp5; *.fp7; *.fmp12
*.fp3
```

Default will be no filter (any file can be selected)

Emai_ImportFile

Description: Imports specified file as object into FileMaker (i.e. to send it as attachment later).

Server: Supported

Parameters:

filePath Path of the file including file name and extension.

Note The plug-in can not return the icon and the file name *together* with a binary object to FileMaker 7-10 (due to a confirmed FileMaker SDK bug). Thus, the items of information *object, file name* and *icon* are returned separately. This affects attachments that are retrieved and files that are imported to FileMaker (i.e. for later sending as attachment). It also affects attachments that are added to a message from FileMaker since they might have only the default file name "Untitled.dat" when stored in FileMaker 10-14. The plug-in provides separate functions to retrieve file names and icons of external files and attachments into separate fields.

Emai_ExportFile

Description: Exports file object from container field to disk (i.e. an attachment that has been retrieved from a message and stored in FileMaker)

Prerequisites: To import external files into FileMaker use `Emai_ImportFile` (incombination with `Emai_GetFileName` etc). To import attachments from a retrieved message into FileMaker use `Emai_PopGetAttachment` (in combination with `Emai_PopGetAttachmentName` etc). Use `Emai_GetUserFolder` to retrieve system folder paths such as Desktop or Temp. Alternatively, use `Emai_ShowFileDialog` to let the user specify the target path.

Server: Supported

Parameters:

fileObject — File object to be exported

filePath — Path (including file name and extension) to which the file will be saved If path does not include file name the default file name from object will be used

{saveMode} — `"strict"` (default), `"rename"` or `"overwrite"`

{protectFile} — Used to write-protect the exported (temp) file if it should only be used for viewing purposes by the user. `"modify"` (default) does not write-protect the file. `"read-only"` protects it from modification.

`saveMode="strict"` this function fails if there is an existing file. If `saveMode="rename"` plug-in adds a counter to the file name if there is a conflict with an existing file. If `saveMode="overwrite"` plug-in tries to overwrite existing file with the same name (if any).

Note — Files that are exported to the temporary folder will be erased by the plug-in when it is unloaded. Thus, it is strongly recommended to set `protecFile="read-only"` when exporting files to the temp folder so that users cannot modify them.

Emai_GetUserFolder

Description: Returns the path of a certain local user folder.

Server: Supported

Parameters:

folderType — Can be `"desktop"`, `"temp"`, `"documents"` or `"userroot"`.

The value `"temp"` returns a special temp folder inside the system's temp folder that it maintained by the plug-in. The plug-in automatically deletes the contents of the folder when it unloads.

Emai_ConvertPath

Description: Mailit 5 operates with paths in the FileMaker path format. This means, paths begin with a `filewin` or a `filemac` prefix depending on the current operating system. You may want to convert retrieved paths into a different format to either display them to user in a native system format or to use them for script steps (i.e. *Insert Picture* and *Insert QuickTime*) that require different path formats. The function lets you convert path into any supported format.

Server: Supported

Parameters:

path	This parameter specifies the source path which is to be converted. The source path must be in `filewin`, `filemac`, `moviewin`, `moviemac`, `imagewin`, `imagemac` format or in a native Windows or Mac path format (depending on the current system).
targetFormat	This parameter is optional. It specifies the format of the output path. The value `"file"` (default) returns the path in `filewin` format (Windows) or `filemac` format (Mac OS). The value `"image"` returns the path in the FileMaker `imagewin` format (Windows) or the `imagemac` format (Mac OS) which is required for the *Insert Picture* script step. The value `"movie"` returns the path in the FileMaker `moviewin` format (Windows) or in the `moviemac` format (Mac OS), which is required for the *Insert QuickTime* script step. The value `"system"` returns the path in the native system format depending on the current operating system (Windows or Mac OS).

Emai_SetupTimer

Description: Schedules a script timer that can be used to send and receive messages in a specified interval.
This function can't be used when running the plug-in on FileMaker Server.

Server: Not supported

Parameters:

filename	Name of the file containing the script to be triggered
scriptName	Name of the script to be triggered
{scriptParameter}	Script parameter to be used for the script triggered
{intervalMinuntes}	Interval in minutes; value `"0"` (default) disables timer

Emai_PerformScript

Description: Add a script execution in to the thread queue.
This function can't be used when running the plug-in on FileMaker Server.

Server: Not supported

Parameters:

threadUID	Thread unique ID, where the script will get queued
fileName	Name of the file containing the script to be triggered
scriptName	Name of the script to be triggered
{scriptParameter}	Script parameter to be used for the script triggered

Emai_ResetMessageHistory

Description: Resets internal message history for specific POP account or all POP accounts and downloads all messages again that are still on server next time the plug-in checks for new messages.

As far as IMAP account download history is saved on the IMAP server, in order to reset the IMAP download history this function should be triggered while connected to the IMAP server, with empty both parameters.

Prerequisites: Working in Mailit 4 compatible mode, in order to reset an IMAP account download history, an IMAP connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

{account}	POP account (login) to be reset
{server}	Address of POP server If both parameters are empty entire internal message history is reset

Emai_SetupStatusDialog

Description: Specifies the settings of the email status dialog that can be shown during message transmission.
This function can't be used when running the plug-in on FileMaker Server.

Prerequisites: Working in Mailit 4 compatible mode, in order to reset an IMAP account download history, an IMAP connection has to be established using `Emai_Connect` before using this function.

Server: Not supported. On Server side will be ignored.

Parameters:

{dialogTitle}	Title of the status dialog; default is empty
{allowCancel}	`"allowCancelOn"` or (default) `"AllowCancelOff"`
{cancelButtonText}	`"cancelButtonText"` or (default) `"AllowCancelOff"`
{allowCancel}	(default) `"Cancel"` or custom value
{textMode}	`"autoText"` (default), `"combinedText"` (custom text is shown on top of auto text) or `"customText"` (only custom text)
{customText}	Custom text to be shown
{resetProgress}	If set to `"reset"` the progress value will be set to zero (used to reset progress from previous dialog sessions) The dialog setup can be changed before or while the dialog is shown. To change only certain set- tings of the dialog while it is shown (and leave other settings unmodified), pass a non-text value (i.e. `"0"`) to the plug-in for those parameters which are not to be updated.
{threadUID}	Use this parameter to attach a dialog to a specific thread. When sending or receiving messages in a thread status dialog also runs in a threaded context. In order to affect a dialog of a specific thread provide thread ID to status dialog functions.
{fileName}	Name of the file containing the script to be triggered, in case if the user will cancel the operation
{scriptName}	Name of the script to be triggered, in case if the user will cancel the operation
{scriptParameter}	Script parameter to be used for the script triggered, in case if the user will cancel the operation

Emai_SetupProgressMax

Description: Specifies the number of messages to define a progress bar maximum for the status dialog. This function can't be used when running the plug-in on FileMaker Server.

Prerequisites: When using the status dialog during message receiving invoke the function `Emai_PopGetMessageCount` to get the number of new messages which will then be used for the `totalMessages` parameter of this function. When using the status dialog during message sending pass the number of messages to be sent to the `totalMessage` parameter of this function.

Server: Not supported. On Server side will be ignored.

Parameters:

{totalMessages} Number of messages the progress bar will be based on.

{threadUID} Use this parameter to affect a dialog in a specific thread.
When sending or receiving messages in a thread status dialog also runs in a threaded context. In order to affect a dialog of a specific thread provide thread ID to status dialog functions.

Emai_ShowStatusDialog

Description: Shows or hides the status dialog.
This function can't be used when running the plug-in on FileMaker Server.

Prerequisites: Before showing the status dialog it is recommended to specify all dialog preferences using `Emai_SetupStatusDialog`.

Server: Not supported. On Server side will be ignored.

Parameters:

option `"show"` or `"hide"`

{threadUID} Use this parameter to affect a dialog in a specific thread.
When sending or receiving messages in a thread status dialog also runs in a threaded context. In order to affect a dialog of a specific thread provide thread ID to status dialog functions.

{queue} This function call may get queued. Use `"queue"` parameter value for this purpose. In this case the function will get executed only when all previous functions of this thread are complete. Otherwise, the plug-in will update the dialog immediately.

Emai_FlashAppIcon

Description: Flashes the application icon in task bar (Windows) or animates the application icon in Doc (Mac OS X). Use this function to indicate that new messages have been retrieved when downloading them with the script timer.
This function can't be used when running the plug-in on FileMaker Server.

Server: Not supported. On Server side will be ignored.

Parameters: {times} Number of times icon will be flashed; `"1"` (default)

Emai_SetupLog

Description: Specifies the log settings of the plug-in. By default the log file will be saved to the Documents folder. Specify log preferences before using log-relevant plug-in functions.

Server: Supported

Parameters:

{setup} `"logOff"`, `"logAppend"` or `"logOverwrite"` (default)

{maxSize} Maximum size of log file in KB (default: no limit)

{logLocation} Specify log file location. By default log file is stored in Documents folder.

Emai_SetupTranscriptSize

Description: Use this function to specify the maximum number of text lines that plug-in will keep for each type of connection. By default the plug-in will keep the last 3000 lines.

The last session transcript can be retrieved using the `Emai_GetLastSessionTranscript` function.

Server: Supported

Parameters: lineNumber Maximum number of text lines to keep

Emai_GetLastSessionTranscript

Description: Use this function to retrieve the last session transcript, according to the connection type specified.

Server: Supported

Parameters:

connectionType `"pop"`, `"imap"`, or `"smtp"`.

{connectionUID} Specify thread unique identifier for SMTP connection

Emai_OpenBrowserHtml

Description: Opens default web browser with specified HTML code or HTML object. Depending of the field type, which is passed to the function, it will treat the content either as `"HTML code"` (in case if `"html"` parameter is a text-type field), or as message object (in case if `"html"` parameter is a container-type field).

`"Message object"` is an object, which contains HTML code and inline images.

This function can't be used when running the plug-in on FileMaker Server.

Prerequisites: To retrieve the HTML code of an incoming HTML message, invoke the function `Emai_PopGetMessageHTML`.

Server: Not supported

Parameters:

html — HTML code or HTML object to be displayed.

{charset} — The plug-in will add the specified character set to the exported HTML file. In case if HTML code already contains a character set, the plug-in will overwrite it. In case If parameter is empty, or set to `"AUTO"` value, the plug-in will use the character set, found in the message source code. Alternatively the following values can be used: `UTF-8`, `ISO8859-1` through `ISO8859-16`, `KOI-8R`, `KOI-8U`, `Windows-1250` through `Windows-1258`, `ISO-2022-JP`, `ISO-2022-JP-1` through `ISO-2022-JP-3`, `CSISO-2022-JP`, `EUC-JP`, `ISO-2022-CN`, `ISO-2022-CN-EXT`, `GB2312`, `HZ-GB-2312`, `ISO-2022-KR`, `KSC5601-1987`, `EUC-KR`, `GBK` and `Big5`.

Emai_ExportHtml

Description: Returns local URL to the exported HTML file to be used for Web Viewer display.

Depending of the field type, which is passed to the function, it will treat the content either as "HTML code" (in case if "html" parameter is a text-type field), or as message object (in case if "html" parameter is a container-type field).

Message object is an object, which contains HTML code and inline images, to be used with Web Viewer display.

Prerequisites: To retrieve the HTML code of an incoming HTML message, invoke the function `Emai_PopGetMessageHTML`.

Server: Supported

Parameters:

html	HTML code or HTML object to be exported.
{destinationPath}	Destination folder, where HTML object should be saved to. By default the object will be saved at the system default Temporary folder.
{charset}	The plug-in will add the specified character set to the exported HTML file. In case if HTML code already contains a character set, the plug-in will overwrite it. In case If parameter is empty, or set to "AUTO" value, the plug-in will use the character set, found in the message source code. Alternatively the following values can be used: `UTF-8`, `ISO8859-1` through `ISO8859-16`, `KOI-8R`, `KOI-8U`, `Windows-1250` through `Windows-1258`, `ISO-2022-JP`, `ISO-2022-JP-1` through `ISO-2022-JP-3`, `CSISO-2022-JP`, `EUC-JP`, `ISO-2022-CN`, `ISO-2022-CN-EXT`, `GB2312`, `HZ-GB-2312`, `ISO-2022-KR`, `KSC5601-1987`, `EUC-KR`, `GBK` and `Big5`.

Emai_FmTextToHtml

Description: Convert FileMaker rich text format to HTML.

The text which is prefixed as URL ("http://," "https://," "ftp://," "mailto:", and "callto:") will be converted to active links automatically.

Server: Supported

Parameters:

text — Text to be converted to HTML.

{parseUrls} — Parse URLs. Values: "Parse" (the plug-in will convert the corresponding text blocks to links), "Ignore". Default: "Ignore".

{charset} — Character set to be used for this conversion. In case if no character set is specified, default character set will be used. The following values are supported: `UTF-8`, `ISO8859-1` through `ISO8859-16`, `KOI-8R`, `KOI-8U`, `Windows-1250` through `Windows-1258`, `ISO-2022-JP`, `ISO-2022-JP-1` through `ISO-2022-JP-3`, `CSISO-2022-JP`, `EUC-JP`, `ISO-2022-CN`, `ISO-2022-CN-EXT`, `GB2312`, `HZ-GB-2312`, `ISO-2022-KR`, `KSC5601-1987`, `EUC-KR`, `GBK` and `Big5`.

Emai_HtmlToFmText

Description: Convert HTML to FileMaker rich text format.

Server: Supported

Parameters:

html — HTML to be converted to FileMaker rich text format.

{charset} — Character set to be used for this conversion. In case if no character set is specified, default character set will be used.
The following values are supported: `UTF-8`, `ISO8859-1` through `ISO8859-16`, `KOI-8R`, `KOI-8U`, `Windows-1250` through `Windows-1258`, `ISO-2022-JP`, `ISO-2022-JP-1` through `ISO-2022-JP-3`, `CSISO-2022-JP`, `EUC-JP`, `ISO-2022-CN`, `ISO-2022-CN-EXT`, `GB2312`, `HZ-GB-2312`, `ISO-2022-KR`, `KSC5601-1987`, `EUC-KR`, `GBK` and `Big5`.

Emai_SetDefaultCharset

Description: Used for decoding if charset if not specified or is invalid (for IMAP, POP3 and SMTP).

For SMTP default charset is used as "recommended". Plugin will try to interpret all text data using this charset, but if it fails it will try other supported charsets.

Plug-in will choose an appropriate default charset on startup, according to the current system settings. This function is a way to override the charset chosen by the plug-in.

Server: Supported

Parameters: {charset} Default character set to be used with this solution. The following values are supported: `UTF-8`, `ISO8859-1` through `ISO8859-16`, `KOI-8R`, `KOI-8U`, `Windows-1250` through `Windows-1258`, `ISO-2022-JP`, `ISO-2022-JP-1` through `ISO-2022-JP-3`, `CSISO-2022-JP`, `EUC-JP`, `ISO-2022-CN`, `ISO-2022-CN-EXT`, `GB2312`, `HZ-GB-2312`, `ISO-2022-KR`, `KSC5601-1987`, `EUC-KR`, `GBK` and `Big5`.

Emai_GetDefaultCharset

Description: The plug-in will set an appropriate default charset on startup, according to current system settings. The function will return the automatically chosen charset, if `"Emai_SetDefaultCharset"` was not used for the solution; or will return the charset which was set by the `"Emai_SetDefaultCharset"` function accordingly.

Emai_RegisterSession

Description: Registers the current plug-in session. The plug-in can also be registered using the preferences dialog (FileMaker Preferences > Plug-Ins > Dacons Mailit 6). If a session is registered using this function the registration takes priority for the current session over the registration information from the preferences.

Server: Supported

Parameters:

user	User name
key	Registration key
Note	User name and Registration key are provided by Dacons after a plug-in license is purchased.

Emai_RegisterServer

Description: Permanently registers the plug-in on FileMaker Server. This function is only valid when running on FileMaker Server.

Server: Supported

Parameters:

user	User name
key	Registration key
Note	User name and Registration key are provided by Dacons after a plug-in license is purchased.

Emai_PopGetHistory*

Description: Updates the download history object for POP3 connections.

Returns the updated download history object for the current session.

Prerequisites: This function should only be used with POP3 accounts.

This function should be triggered after `Emai_Disconnect` function call.

Make sure to specify a custom download history object, when invoking the `Emai_Connect` function.

Server: Supported

Parameters: {connectionUID} This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_PopGetMessageCount*

Description: Returns the number of messages for current POP connection waiting for download on server.
For single-threaded connections should be triggered prior to `Emai_PopRetrieveMessage` loop.
For multi-threaded connections should be triggered after `Emai_PopRetrieveAllMessages` call.

Prerequisites: A POP3 connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters: {connectionUID} This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_PopGetMessageInfo

Description: Uses current POP connection to download message header from server. All items of information from the header can be retrieved afterwards with appropriate functions. This does not include information about attachments. The size of the message can not be retrieved after calling this function (`Emai_PopGetMessageCount` has to be called to before retrieving message size).

Prerequisites: A POP3 connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_PopGetMessageCount` before using this function to know how many new messages are on the server (needed to be sure about the messageIndex parameter).

Server: Supported

Parameters: messageIndex Index of new message on server (beginning with 1)

Emai_PopRetrieveMessage

Description: Uses current POP connection to download message from server.

Prerequisites: A POP3 connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_PopGetMessageCount` before using this function to know how many new messages are on the server (needed to be sure about the messageIndex parameter).

Server: Supported

Parameters: messageIndex Index of new message on server (beginning with 1)

Emai_PopRetrieveAllMessages**

Description: Uses POP connection UID to download all new messages from server.

Prerequisites: A POP3 connection has to be established using `Emai_Connect` in a threaded context before using this function.
This function does not require `Emai_PopGetMessageCount` call before use.

Server: Not supported. On Server side use single-threaded functions.

Parameters: {connectionUID} Connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_PopGetMessageSource*

Description: Returns the entire source of the current message as message file object.

Prerequisites: A POP3 connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_PopGetMessageInfo` or `Emai_PopRetrieveMessage` before using this function.

Server: Supported

Parameters: {connectionUID} This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_PopGetContactsCount

Description: Returns the number of contacts for a specified field of the current message.

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Parameters: fieldName `"from"`, `"to"`, `"cc"`, `"reply-to"` or `"sender"`

Emai_PopGetContact

Description: Returns name or address of a specified message contact of the current message

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. Call `Emai_PopGetContactsCount` before using this function to know the total number of contacts (needed for the `contactIndex` parameter).

Server: Supported

Parameters:

fieldName	`"from"`, `"to"`, `"cc"`, `"reply-to"` or `"sender"`
fieldType	`"name"` or `"address"`
contactIndex	Index of contact in this field (starting with 1)

Emai_PopGetMessageHeader

Description: Returns the header of the current message.

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_PopGetMessageBody

Description: Returns a plain `"text"` body of the current message, merging all `"text"` parts of the message. If no `"text"` body is discovered plug-in converts the first `"html"` part into text and returns it as result of this function.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. Invoking `Emai_PopGetMessageInfo` alone prior to calling this function will not work.

Server: Supported

Emai_PopGetMessageHTML

Description: Returns the HTML code of the current message. Therefore, the first HTML part (in multi-part messages) is used. If the message contains several HTML parts all further parts are returned as HTML file attachments.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. Invoking `Emai_PopGetMessageInfo` alone prior to calling this function will not work.

Server: Supported

Emai_PopGetMessageCharset

Description: Returns the character set of the current message.

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_PopGetMessagePriority

Description: Returns the priority of the current message as text value: `"Very High"`, `"High"`, `"Normal"`, `"Low"` or `"Very Low"`.

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_PopGetMessageField

Description: Returns a specified header field of the current message.

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Parameters: fieldName Name of the header field, i.e. `"to"`, `"from"`, `"subject"`, etc.

Emai_PopGetRecvDate

Description: Returns the POP server receiving date of current message. The date is returned as text in the following format: DD/MM/YYYY

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_PopGetRecvTime

Description: Returns the POP server receiving time of current message. The time is returned as text in the following format: HH:MM:SS

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_PopGetMessageSize

Description: Returns size in Bytes of a specified message on server.

Prerequisites: Retrieve the list of messages by invoking `Emai_PopGetMessageCount`

Server: Supported

Parameters: messageIndex Index of new message on server (beginning with 1)

Emai_PopDeleteMessage

Description: Deletes a specific message from POP server.

Prerequisites: Invoke `Emai_PopGetMessageCount` to retrieve the number of new messages from server. Use `Emai_PopGetMessageInfo` to retrieve the header of a specific message. Delete certain messages using this function if you would not like to download them.

Server: Supported

Parameters: messageIndex Index of new message on server (beginning with 1)

Emai_PopGetAttachmentCount

Description: Returns the number of attachments of the current message.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header.

Server: Supported

Parameters: {attachmentType} Type of attachments to be counted; `"inline"`, `"attachment"` or `"all"` (default).

Emai_PopGetAttachment

Description: Returns specified attachment as binary object.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
{attachmentType}	Type of attachment; `"inline"`, `"attachment"` or `"all"` (default)
Note	The plug-in can not return the icon and the file name *together* with a binary object to FileMaker 10-14. Thus, the items of information *object, file name* and *icon* are returned separately. This affects attachments that are retrieved and files that are imported to FileMaker (i.e. for later sending as attachment). It also affects attachments that are added to a message from FileMaker since they might have only the default file name 'Untitled.dat' when stored in FileMaker 10-14. The plug-in provides separate functions to retrieve file names and icons of external files and attachments into separate fields.

Emai_PopExportAttachment

Description: Saves specified attachment to disk and returns the file path. Use this function to export incoming attachments directly to disk without storing them in FileMaker.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
destinationPath	Can be exact file path including file name or path to a folder (in second case the original file name of the attachment will be used)
{attachmentType}	Type of attachment; `"inline"`, `"all"` or `"attachment"` (default)
{saveMode}	`"strict"`, `"rename"` (default) or `"overwrite"`

Emai_PopGetAttachmentName

Description: Returns original file name and extension of an attachment.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
{attachmentType}	Type of attachment; `"inline"`, `"attachment"` or `"all"` (default)

Emai_PopGetAttachmentSize

Description: Returns size in Bytes of the specified attachment.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
{attachmentType}	Type of attachment; `"inline"`, `"attachment"` or `"all"` (default)

Emai_PopGetAttachmentIcon

Description: Returns icon of the specified attachment as image.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
{attachmentType}	Type of attachment; `"inline"`, `"attachment"` or `"all"` (default)
{iconSize}	Icon size in pixels, `"16"` (default) or `"32"`

Emai_PopGetAttachmentCID

Description: Returns attachment Content ID. This function is only relevant for `"inline"` attachments, for any other en empty string will be returned. `"attachmentType"` parameter is used for consistency with other functions, and convenience in scripting.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
{attachmentType}	Type of attachment; `"inline"`, `"attachment"` or `"all"` (default)

Emai_ImapListFolders*

Description: Returns the list of folders, available at IMAP server.

Prerequisites: An IMAP connection has to be established using `Emai_Connect`, before using this function.

Server: Supported

Parameters:

{connectionUID} This parameter should contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapSelectFolder*

Description: Selects a folder, to be used with the other IMAP messages retrieval functions. For example: `"Drafts"`.
In case if no folder is selected, the plug-in will use the `"Inbox"` folder, by default.

Prerequisites: An IMAP connection has to be established using `Emai_Connect`, before using this function.

Server: Supported

Parameters:

`folder` IMAP Folder name to be selected.

{connectionUID} This parameter should contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapGetMessageCount*

Description: Returns the number of messages for current IMAP connection waiting for download on server.
For single-threaded connections should be triggered prior to `Emai_ImapRetrieveMessage` loop.
For multi-threaded connections should be triggered after `Emai_ImapRetrieveAllMessages` call.

Prerequisites: An IMAP connection has to be established using `Emai_Connect`, before using this function.

Server: Supported

Parameters:

{messageUIDL} Use this parameter to supply the list of message unique identifiers, available in database, for a particular folder. The plug-in will use this list to find the messages that should be downloaded.
The list should contain message UIDs separated by carriage return character. Use the `Emai_ImapGetMessageUID` function to obtain a UID for each downloaded message.
This parameter is necessary when running in non-threaded mode. Otherwise use empty parameter.

{connectionUID} This parameter should contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapGetMessageInfo

Description: Uses current IMAP connection to download message header from server. All items of information from the header can be retrieved afterwards with appropriate functions. This does not include information about attachments. The size of the message can not be retrieved after calling this function (`Emai_ImapGetMessageCount` has to be called to before retrieving message size).

Prerequisites: An IMAP connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_ImapGetMessageCount` before using this function to know how many new messages are on the server (needed to be sure about the `messageIndex` parameter).

Server: Supported

Parameters:

messageIndex Index of new message on server (beginning with 1)

Emai_ImapRetrieveMessage

Description: Uses current IMAP connection to download message from server.

Prerequisites: An IMAP connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_ImapGetMessageCount` before using this function to know how many new messages are on the server (needed to be sure about the `messageIndex` parameter).

Server: Supported

Parameters: messageID Index of new message (beginning with 1) or message UID on server

Emai_ImapRetrieveAllMessages**

Description: Uses IMAP connection UID to download all new messages from server.

Prerequisites: An IMAP connection has to be established using `Emai_Connect` in a threaded context before using this function.
This function does not require `Emai_ImapGetMessageCount` call before use.

Server: Not supported. On Server side use single-threaded functions.

Parameters: {connectionUID} Connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapGetMessageUID*

Description: Returns current message IMAP unique ID.

Prerequisites: An IMAP connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_ImapGetMessageInfo` or `Emai_ImapRetrieveMessage` before using this function.

Server: Supported

Parameters: {connectionUID} This parameter should contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

{messageIndex} Index of new message on server (beginning with 1), used for multi-threaded receiving.

Emai_ImapGetMessagesByFlags

Description: Returns the list of IMAP message unique IDs, that correspond to the list of flags specified, separated by carriage return.

Prerequisites: This function requires an active IMAP connection established. A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

messageFlags `"seen"`, `"unseen"`, `"recent"`, `"draft"`, `"deleted"`, `"flagged"` or `"answered"`.

Emai_ImapGetDeleteUIDList*

Description: Returns the list of unique message IDs, that are not available on the IMAP server any longer. This list should be used to delete those messages from the database, as they are no longer relevant.

Prerequisites: An IMAP connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

messageUIDL — Use this parameter to supply the list of message unique identifiers, available in database, for a particular account. The plug-in will use this list to find the messages that should be deleted.
The list should contain message UIDs separated by carriage return character. Use the `Emai_ImapGetMessageUID` function to obtain a UID for each downloaded message.

{connectionUID} — This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapGetMessageSource*

Description: Returns the entire source of the current message as message file object.

Prerequisites: An IMAP connection has to be established using `Emai_Connect` before using this function. Invoke `Emai_ImapGetMessageInfo` or `Emai_ImapRetrieveMessage` before using this function.

Server: Supported

Parameters:

{connectionUID} — This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

messageID — Index of new message (beginning with 1) or message UID on server

Emai_ImapGetContactsCount

Description: Returns the number of contacts for a specified field of the current message.

Prerequisites: Make a message the current message by invoking `Emai_PopGetMessageInfo`, `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Parameters: fieldName `"from"`, `"to"`, `"cc"`, `"reply-to"` or `"sender"`.

Emai_ImapGetContact

Description: Returns name or address of a specified message contact of the current message.

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`. Call `Emai_ImapGetContactsCount` before using this function to know the total number of contacts (needed for the contactIndex parameter).

Server: Supported

Parameters:

fieldName	`"from"`, `"to"`, `"cc"`, `"reply-to"` or `"sender"`.
fieldType	`"name"` or `"address"`.
contactIndex	Index of contact in this field (starting with 1).

Emai_ImapGetMessageHeader

Description: Returns the header of the current message.

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_ImapGetMessageBody

Description: Returns a plain `"text"` body of the current message, merging all `"text"` parts of the message. If no `"text"` body is discovered plug-in converts the first `"html"` part into text and returns it as result of this function.

Prerequisites: Make a message the current message by invoking `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`. Invoking `Emai_ImapGetMessageInfo` alone prior to calling this function will not work.

Server: Supported

Emai_ImapGetMessageHTML

Description: Returns the HTML code of the current message. Therefore, the first HTML part (in multi-part messages) is used.
If the message contains several HTML parts all further parts are returned as HTML file attachments.

Prerequisites: Make a message the current message by invoking `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`. Invoking `Emai_ImapGetMessageInfo` alone prior to calling this function will not work.

Server: Supported

Emai_ImapGetMessageCharset

Description: Returns the character set of the current message.

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_ImapGetMessagePriority

Description: Returns the priority of the current message as text value: `"Very High"`, `"High"`, `"Normal"`, `"Low"` or `"Very Low"`.

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_ImapGetMessageField

Description: Returns a specified header field of the current message.

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Parameters: fieldName Name of the header field, i.e. `"to"`, `"from"`, `"subject"`, etc.

Emai_ImapGetMessageFlags*

Description: Returns all IMAP flags for the current message, separated by carriage return character. Standard flags are: `"seen"`, `"unseen"`, `"recent"`, `"draft"`, `"deleted"`, `"flagged"` and `"answered"`. Other values are treated as custom flags.

Prerequisites: This function requires an active IMAP connection established. A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

{messageUID} Specify the IMAP message unique ID, in order to retrieve the message flags for this specified message. Otherwise the plug-in will use the last retrieved message UID.

{connectionUID} This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapSetMessagesFlags*

Description: Apply IMAP flags for the current message or for the list of messages, separated by carriage return character.

Prerequisites: This function requires an active IMAP connection established. A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

messageFlags	IMAP message flags to be applied, separated by carriage return character. Standard flags are: `"seen"`, `"recent"`, `"draft"`, `"deleted"`, `"flagged"` and `"answered"`. Other values are treated as custom flags.
{messageUID}	Specify the IMAP message unique ID list, separated by carriage return character. Otherwise the plug-in will use the last retrieved message UID.
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapRemoveMessagesFlags*

Description: Remove IMAP flags for the current message, separated by carriage return character.

Prerequisites: This function requires an active IMAP connection established. A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

messageFlags	IMAP message flags to be applied, separated by carriage return character. Standard flags are: `"seen"`, `"recent"`, `"draft"`, `"deleted"`, `"flagged"` and `"answered"`. Other values are treated as custom flags.
{messageUIDL}	Specify the IMAP message unique ID list, separated by carriage return character. Otherwise the plug-in will use the last retrieved message UID.
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapGetRecvDate

Description: Returns the IMAP server receiving date of current message. The date is returned as text in the following format: DD/MM/YYYY

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_ImapGetRecvTime

Description: Returns the IMAP server receiving time of current message. The time is returned as text in the following format: HH:MM:SS

Prerequisites: Make a message the current message by invoking `Emai_ImapGetMessageInfo`, `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`.

Server: Supported

Emai_ImapGetMessageSize

Description: Returns size in Bytes of a specified message on server.

Prerequisites: Retrieve the list of messages by invoking `Emai_ImapGetMessageCount`

Server: Supported

Parameters: messageIndex Index of message on server (beginning with 1)

Emai_ImapDeleteMessage*

Description: Deletes a specific message from IMAP server.

Prerequisites: Invoke `Emai_ImapGetMessageCount` to retrieve the number of new messages from server. Use `Emai_ImapGetMessageInfo` to retrieve the header of a specific message. Delete certain messages using this function if you would not like to download them.

Server: Supported

Parameters:

messageID	Index of the message to be deleted (beginning with 1) or message UID on server
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapGetAttachmentCount

Description: Returns the number of attachments of the current message.

Prerequisites: Make a message the current message by invoking `Emai_ImapRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_ImapGetMessageInfo` only since attachment information is not part of the message header.

Server: Supported

Parameters:

{attachmentType}	Type of attachments to be counted; `"inline"`, `"attachment"` or `"all"` (default)

Emai_ImapGetAttachment

Description: Returns specified attachment as binary object.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
{attachmentType}	Type of attachment; `"inline"`, `"attachment"` or `"all"` (default)

Note

The plug-in can not return the icon and the file name together with a binary object to FileMaker 10-14. Thus, the items of information object, file name and icon are returned separately. This affects attachments that are retrieved and files that are imported to FileMaker (i.e. for later sending as attachment). It also affects attachments that are added to a message from FileMaker since they might have only the default file name `"Untitled.dat"` when stored in FileMaker 10-14. Attachments with no name supplied shell get an automatic name `"ATT"` with a running number.

The plug-in provides separate functions to retrieve file names and icons of external files and attachments into separate fields.

Emai_ImapExportAttachment

Description: Saves specified attachment to disk and returns the file path. Use this function to export incoming attachments directly to disk without storing them in FileMaker.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
destinationPath	Can be exact file path including file name or path to a folder (in second case the original file name of the attachment will be used)
{attachmentType}	Type of attachment; `"all"`, `"inline"` or `"attachment"` (default)
{saveMode}	`"strict"`, `"rename"` (default) or `"overwrite"`

Emai_ImapGetAttachmentName

Description: Returns original file name and extension of an attachment.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
destinationPath	Can be exact file path including file name or path to a folder (in second case the original file name of the attachment will be used)
{attachmentType}	Type of attachment; `"all"`, `"inline"` or `"attachment"` (default)

Emai_ImapGetAttachmentSize

Description: Returns size in Bytes of the specified attachment.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
destinationPath	Can be exact file path including file name or path to a folder (in second case the original file name of the attachment will be used)
{attachmentType}	Type of attachment; `"all"`, `"inline"` or `"attachment"` (default)

Emai_ImapGetAttachmentCID

Description: Returns attachment Content ID. This function is only relevant for `"inline"` attachments, for any other en empty string will be returned. `"attachmentType"` parameter is used for consistency with other functions, and convenience in scripting.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
destinationPath	Can be exact file path including file name or path to a folder (in second case the original file name of the attachment will be used)
{attachmentType}	Type of attachment; `"all"`, `"inline"` or `"attachment"` (default)

Emai_ImapGetAttachmentIcon

Description: Returns icon of the specified attachment as image.

Prerequisites: Make a message the current message by invoking `Emai_PopRetreiveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Emai_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Server: Supported

Parameters:

attachmentIndex	Index of attachment from current message (starting with 1)
destinationPath	Can be exact file path including file name or path to a folder (in second case the original file name of the attachment will be used)
{attachmentType}	Type of attachment; `"all"`, `"inline"` or `"attachment"` (default)
{iconSize}	Icon size in pixels, `"16"` (default) or `"32"`

Emai_ImapCreateFolder*

Description: Create a folder on the IMAP server side. For example: `"Important Messages"`.

Prerequisites: An IMAP connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

folderName	Folder name to be created.
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapUploadMessageToFolder*

Description: Uploads the message object to the IMAP folder specified.

Prerequisites: This function requires an active IMAP connection established. A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

messageObject	Message object to be uploaded. This can be either message object returned by the `Emai_SmtpComposeMessage`, `Emai_ImapGetMessageSource` or `Emai_PopGetMessageSource` functions; or this can be also a unique message ID, generated by the `Emai_SmtpCreateMessage` function.
{folderName}	Specify the target folder name. Otherwise, the plug-in will use the active IMAP folder. For example: `"Sent Messages"`.
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_ImapMoveMessage*

Description: Moves the message from current IMAP folder to the specified IMAP folder.

Prerequisites: This function requires an active IMAP connection established. A connection has to be established using `Emai_Connect` before using this function.

Server: Supported

Parameters:

messageID	Index of message to move on (beginning with 1), or message UID on server.
destinationFolder	Destination folder on server. For example: `"Important Messages"`.
{connectionUID}	This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded receiving.

Emai_SmtpDestroyMessage

Description: A message object has to be created using `Emai_SmtpCreateMessage` before using this function.

Server: Supported

Parameters:

messageUID	Message unique ID, returned by the `Emai_SmtpCreateMessage` function.

Emai_SmtpImportMessageParameters

Description: Use this function to import message parts, generated out of a particular message scope.

Prerequisites: A message object has to be created using `Emai_SmtpCreateMessage` before using this function.

Server: Not supported. On Server side use single-threaded functions.

Parameters:

targetMessageUID	Target message UID, that should receive the imported message information.
{importAttachments}	Use the `"importAttachments"` parameter value to import the attachment objects.
{importBodies}	Use the `"importBodies"` parameter value to import the body objects.
{importHeader}	Use the `"importHeader"` parameter value to import the header objects.

Emai_SmtpResetMessage

Description: Resets all internal message variables of the plug-in. Use this function in a sending loop before sending the next message.
This function will only reset the message information generated out of a particular message scope. In order to destroy a particular message use the `Emai_SmtpDestroyMessage` function.

Server: Supported

Parameters:

{resetAttachments} `"keepSelectedAttachments"` (default) resets all attachments except for those that have been set with persistency= `"keep"` ; `"resetAttachments"` resets all attachments including those that have been set with persistency= `"keep"` .

{resetBodies} `"keepSelectedBodies"` (default) resets all bodies except for those that have been set with persistency= `"keep"` ; `"resetBodies"` resets all bodies including those that have been set with persistency= `"keep"` .

{resetHeader} `"resetHeader"` (default) resets all header field values for the next message; `"keepHeader"` keeps all header fields values. The header can be modified or values can be added for the next message.

Note The use of the following values for the first parameter is still supported but deprecated as of version 3.0.5: `"keepSelected"` (results in default values for all three parameters), `"keepNone"` (results in `"resetAttachments"` , `"resetBodies"` , `"resetHeader"`).

Emai_SmtpGenerateMessageID

Description: Generates message ID, returns ID and uses it for next outgoing message. Use this function to specify the message ID on the client side. If the message ID is not specified it will be generated by the SMTP server and will be unknown to the sending client.

Server: Supported

Parameters:

{suffix} Custom suffix for message ID (i.e. `"MySolution"`)

{messageUID} Use this parameter to affect a specific message object.

Emai_SmtpAddMessageField

Description: Sets a header field for outgoing message; covers all header fields including `"from"`, `"sender"`, `"reply-to"`, `"to"`, `"cc"`, `"in-reply-to"`, `"subject"`.

Prerequisites: Call `Emai_SmtpResetMessage` to start a new message; otherwisethe header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Server: Supported

Parameters:

fieldName	Name of the header field to be set, i.e. `"from"`, `"to"` or `"subject"`
fieldValue	Content of the message field; format depends on content type (`fieldName` parameter value). This can be an email address or a subject, for instance.
{messageUID}	Use this parameter to affect a specific message object.

Emai_SmtpAddContact

Description: Sets a name and address as message contact.

Prerequisites: Call `Emai_SmtpResetMessage` to start a new message; otherwisethe header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Server: Supported

Parameters:

fieldName	Name of the header field to be set, i.e. `"from"`, `"to"`, `"cc"`, `"bcc"`, `"reply-to"` or `"sender"`
name	Name of the contact.
address	Address of the contact The parameter name can also be used to pass a combined value (name and address) in the format of `"name"` <address> to the plug-in. In this case the address parameter should be empty.
{messageUID}	Use this parameter to affect a specific message object.
Note	According to email standards only one `"sender"` is allowed per message. If more than once `"sender"` is specified the existing `"sender"` value will be replaced for the current message.

Emai_SmtpSetMessagePriority

Description: Set the priority of the outgoing message.

Prerequisites: Call `Emai_SmtpResetMessage` to start a new message; otherwisethe header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Server: Supported

Parameters:

priority	`"Very High"`, `"High"`, `"Normal"` (default), `"Low"` or `"Very Low"`.
{messageUID}	Use this parameter to affect a specific message object.

Emai_SmtpAddBody

Description: Adds a message body part for the next outgoing email.

Prerequisites: Call `Emai_SmtpResetMessage` to start a new message; otherwisethe header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Server: Supported

Parameters:

format — `"text"` for plain text, `"html"` for HTML code or `"rich"` for formatted FileMaker text

content — Content (plain text, HTML code or formatted FileMaker text); this parameter can also pass a JPEG image to the plug-in which will then generate the HMTL code to show the JPEG as inline (format parameter to be set to `"html"`)

{persistency} — Indicates whether the body will persist after invoking `ResetMessage` function for next message; `"once"` (default) does not keep the body after `ResetMessage` ; `"keep"` keeps it if the `ResetMessage` function is invoked.

{messageUID} — Use this parameter to affect a specific message object.

Note — If `"rich"` is used as value for the content parameter the plug-in generated HTML code that describes the text formatting of the FileMaker text passed to it. There can only be one HTML body at a time for an email. If a new `"html"` or `"rich"` body is defined, earlier HTML bodies are replaced. The text which is prefixed as URL (`"http://"` , `"https://"` , `"ftp://"` , `"mailto:"` , and `"call-to:"`) will be converted to active links automatically.

An email can have a maximum of two parts (HTML and TEXT). If both are set, TEXT is interpreted as alternative text part for email clients that cannot display HTML.
If no TEXT part is set the plug-in will automatically generate one from the HTML part before sending the message.

Emai_ImportJpegClipboard

Description: Imports a JPEG from the clipboard and passes it to FileMaker. This function can be used together with `Emai_SmtpAddBody` to import the image of a layout (from Preview mode) and send it as inline image.

Prerequisites: The content to be retrieved as JPEG must be copied to clipboard before this function is invoked. To achieve this for a layout, go to Preview Mode and trigger the script step Copy.

Server: Not supported

Parameters: {quality} Specifies the quality of the JPEG that is created; 1 stands for smallest file size and 10 stands for best quality; default value is 8.

Emai_ImportPNGClipboard

Description: Imports a PNG from the clipboard and passes it to FileMaker. This function can be used together with `Emai_SmtpAddBody` to import the image of a layout (from Preview mode) and send it as inline image.

Prerequisites: The content to be retrieved as PNG must be copied to clipboard before this function is invoked. To achieve this for a layout, go to Preview Mode and trigger the script step Copy.

Server: Not supported

Emai_SmtpGenerateContentID

Description: Generates a content ID for inline attachments that is used to link an inline attachment to the appropriate tag in the HTML code of the message. Use the content ID in the HTML code like this:

```
<img src="cid:conent_id_here">
```

When composing the message, pass the content ID of an attachment to the plug-in using the `contentID` parameter of `Emai_SmtpAddAttachment`.

Server: Supported

Parameters: fileName Name of the file to be used as inline attachment.

Emai_SmtpAddAttachment

Description: Adds an attachment to the current message

Prerequisites: Call `Emai_SmtpResetMessage` to start a new message; otherwise the header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Server: Supported

Parameters:

fileObject	File object (from FM field) to be attached. If an external file is to be attached the function `Emai_ImportFile` is called for this parameter. Will accept "binary file" object, as well as "image", "movie", and other container field types. Also will accept the `Emai_ImportJpegClipboard` function result.
{attachmentType}	`"all"`, `"attachment"` (default) or `"inline"`
{fileName}	Name of the attachment object including extension; this value can be used to rename a file or object when sending it (especially plug-in saved objects that have only default name)
{persistency}	Indicates whether attachment will persist after calling `ResetMessage` function for next message; `"once"` (default) does not keep the attachment after `ResetMessage`; `"keep"` keeps it.
{contentID}	The function `Emai_SmtpGenerateContentID` is used to generate a content ID for attachments that is used here to link an inline attachment to the appropriate tag in the HTML code of the message. Content IDs can also be generated for attachments that are not set as inline but as "visible" attachments.
{windowsFriendly}	When set to `"windowsFriendly"` value, on Mac OS X will make the plug-in avoid using the Apple Double encoding.
{messageUID}	Use this parameter to affect a specific message object.

Emai_SmtpSendMessage

Description: Sends a message using the current SMTP connection. This function can send a pre-composed message (message source object) or it composes a message from the current message variables that have been set before.

Prerequisites: Establish a connection to a SMTP server using `Emai_Connect` before invoking this function. Set message variables before sending a message (i.e. `Emai_SmtpAddBody`). If sending a pre-composed message source object invoke `Emai_SmtpComposeMessage` prior to this function.

Make sure that message contains at least one recipient, otherwise the function will fail.

Specify a message UID in order to send a specific message object.

Server: Supported

Parameters:

{messageObject} Message object that has been composed earlier. If `"messageObject"` parameter is empty a message is sent using the current internal message variables.

{options} By default this parameter is set to `"CancelOnRcptErr"`. If a recipient is not accepted by the SMTP server, the message is not sent to any of the recipients. To make the plug-in try to send the message even though at least one of the recipients is not accepted set this parameter to the value `"ContinueOnRcptErr"`. The result returned will contain a listing of all failed recipients.

{connectionUID} This parameter may optionally contain a connection ID, created by the `Emai_Connect` plug-in function, used for multi-threaded sending.

{messageUID} Use this parameter to affect a specific message object.

Emai_SmtpComposeMessage

Description: Composes message object of current internal variables and returns the message object (for later sending).

Prerequisites: Before composing the message object set all message variables including body and attachments. Once a message object has been composed it cannot be modified.

Server: Supported

Parameters:

{messageUID} Use this parameter to affect a specific message object.

Emai_SmtpFalsifyAddress

Description: Validates an email address without sending a message. The function tries to ask the receiving server if the specified address exists. Such requests can be successful but they might also fail. Only use this function for mass mailing solutions. Do not rely on the result.

Prerequisites: This function requires an Internet connection. However, the plug-in does not have to be connected before using this function (so invoking `Emai_Connect` is not required).

Server: Supported

Parameters:

address	Email address to be validated.
{timeout}	Timeout for each server connection in seconds; `"10"` (default); Lookup path can be recursive

Result: If no network or email address format error occurred, this function returns one of the following result codes:

038	"Address probably valid"
039	"Address invalid: domain not found"
040	"Address invalid: mailbox not found"
041	"Address could not be checked: reason unknown"

CHAPTER 4
Attachments

Mailit 6 enables you to send and receive messages with attachments. You can store attachments in FileMaker container fields (optional), or export them to disk and open them.

Receiving Attachments

When receiving attachments you can decide if you want to store files in FileMaker container fields and have users export them from there as needed or if attachments should be exported to disk immediately. When storing incoming attachments in FileMaker container fields you can ensure that they are always associated with the correct email message by using an *Attachment* table that is related to a Message table (one-to-many relationship). In addition attachments can be protected using access privileges and they can be shared with other FileMaker clients.

Store attachments in a FileMaker container field
The functions `Emai_PopGetAttachment` or `Emai_ImapGetAttachment` is invoked after a message has been retrieved using `Emai_PopRetrieveMessage` or `Emai_ImapRetrieveMessage`. The function `Emai_PopGetAttachment` or `Emai_ImapGetAttachment` returns one attachment at a time. To retrieve all attachments of a message invoke this function in a loop until an empty result is returned.

Attachments are stored to FileMaker container fields as binary objects. Attachment name, icon and size can be obtained from Mailit using the functions
`Emai_PopGetAttachmentName` / `Emai_ImapGetAttachmentName`,
`Emai_PopGetAttachmentIcon` / `Emai_ImapGetAttachmentIcon` and
`Emai_PopGetAttachmentSize` / `Emai_PopGetAttachmentSize`. Invoke these functions after `Emai_PopGetAttachment` / `Emai_ImapGetAttachment` and store the results in separate FileMaker fields. To store the result of the function `Emai_PopGetAttachmentIcon` / `Emai_ImapGetAttachmentIcon` a container should be used.

Save an attachment that has been stored in a container field as a file to disk
The function `Emai_ExportFile` is invoked. Use the function `Emai_ShowFileDalog` to show a

file dialog that lets the user specify a target location. Instead, you can also trigger `Emai_GetUserFolder` to retrieve the path of the Desktop, the temp folder or other user-specific folders. The temp folder path should be used as default export location when you want to open a temporary copy of an attachment that is stored in a container field. Having exported an attachment, it can be opened using the function `Emai_OpenFile`.

Export attachments to a volume without storing them in FileMaker
Invoke the function `Emai_PopExportAttachment` / `Emai_ImapExportAttachment` instead of `Emai_PopGetAttachment` / `Emai_ImapGetAttachment` in a loop. For this setup it is recommended that you use a global attachment folder for all attachments that are retrieved. Invoke the function `Emai_ShowFolderDialog` to let the user select such an attachment folder. The path is then used by `Emai_PopExportAttachment` / `Emai_ImapExportAttachment` for all enclosures.

If you decide to not store attachments in FileMaker you can still import the name, icon and size of attachments using `Emai_PopGetAttachmentName` / `Emai_ImapGetAttachmentName`, `Emai_PopGetAttachmentIcon` / `Emai_ImapGetAttachmentIcon` and `Emai_PopGetAttachmentSize` / `Emai_ImapGetAttachmentSize`.

Sending Attachments

Just like with incoming attachments Mailit gives you two main options when sending attachments. You can either import files into FileMaker before sending them to keep them stored in the database or you can send files from disk directly as attachments without storing them in FileMaker. For most cases it is recommended that you store attachments related to messages in FileMaker for further reference. This way you ensure that you keep a copy of the exact file you sent with a message.

Use the function `Emai_ShowFileDialog` to let users locate a file, which is to be used attachment. To import the selected file into FileMaker the path returned is used as input for the function `Emai_ImportFile`. In case you do not want to store attachments in FileMaker save the selected path to a field at this point. No matter if you decide to import outgoing attachments into FileMaker or not you can use the functions `Emai_GetFileName`, `Emai_GetFileIcon` and `Emai_GetFileSize` to import the attachment name, icon and file size into FileMaker fields.

To set an attachment for a message the function `Emai_SmtpAddAttachment` is invoked. If the attachment has not been stored in FileMaker embed the function `Emai_ImportFile` into this function call and use the attachment path that has been obtained earlier using `Emai_ShowFileDialog`. This lets Mailit import a file directly when setting it as attachment without storing it in FileMaker.

The following example adds an external file to the next message as attachment without importing it into FileMaker. This example assumes that the path of the file is stored in a text field.

```
Emai_SmtpAddAttachment(Emai_ImportFile(Globals::AttachmentPath);
```

```
"attachment"; Emai_GetFileName(Globals::AttachmentPath);
"once";"" )
```

Before adding attachments to an outgoing message using `Emai_SmtpAddAttachment` the function `Emai_SmtpResetMessage` is invoked, when using a non-threaded mode. It resets all internal message variables (from last message) and has options that enable you to reuse encoded attachments from the previous message. This way, attachments that are sent with multiple messages have to be encoded only once which saves time when sending many messages. When sending mass mailings it is even possible to combine global attachments (same for all recipients; encoded only once) and individual attachments (different for every message).

Just like incoming attachments you can export and open outgoing attachments that have been imported into FileMaker by using the function `Emai_ExportFile` and `Emai_OpenFile`.

Advanced Message Processing

In addition to the approaches described so far to handle message contents, Mailit provides advanced techniques that improve the message processing performance.

After retrieving a message using `Emai_PopRetrieveMessage` / `Emai_ImapRetrieveMessage` there are several options of processing the information retrieved and especially attachments.

Decode attachments directly after receiving (slower)
One alternative is to decode a message including all attachments completely after receiving it. All header fields and message bodies are retrieved using the functions `Emai_PopGetMessageBody` / `Emai_ImapGetMessageBody` and `Emai_PopGetMessageField` / `Emai_ImapGetMessageField`. Attachments are decoded immediately and are stored to FileMaker container fields by invoking `Emai_PopGetAttachment` / `Emai_ImapGetAttachment`. Attachments can also be exported to disk immediately using `Emai_PopExportAttachment` / `Emai_PopExportAttachment`.

The drawback of decoding all message information including attachments when receiving a message is that all the decoding work is performed before the next message is retrieved. This delays the receiving process when downloading many messages. Thus, it is recommended to follow the alternative approach described below.

Decode attachments as needed (faster)
Instead of invoking the function `Emai_PopRetrieveMessage` / `Emai_ImapRetrieveMessage` the function `Emai_PopGetMessageInfo` / `Emai_ImapGetMessageInfo` is triggered to download the header of a message. This function provides all information about a message that is necessary to show it in a list view. After calling `Emai_PopGetMessageInfo` / `Emai_ImapGetMessageInfo` functions such as `Emai_PopGetMessageField` / `Emai_ImapGetMessageField` can be invoked to retrieve the sender and the subject. Recipients can be retrieved using

Emai_PopGetMessageContact / Emai_PopGetMessageContact.
In addition to Emai_PopGetMessageInfo / Emai_ImapGetMessageInfo, the function Emai_PopGetMessageSource / Emai_ImapGetMessageSource is called. It retrieves the un-decoded message as a binary object that is stored in a FileMaker container field.
No more information about the message is retrieved to FileMaker in the receiving script. Only at a later point (i.e. when the user selects a message list view entry to open it) the message is decoded. Therefore, the message object from a container field is passed to the function Emai_ProcessMessageSource. After that, all message bodies can be retrieved using Emai_PopGetMessageBody / Emai_ImapGetMessageBody. Moreover, information about the attachments of the message is now made available and can be retrieved using functions like Emai_PopGetAttachmentName / Emai_PopGetAttachmentName.
Attachment names and icons can be shown as part of the opened message just as you would expect it from a stand-alone email client. Note that the attachments of the opened message have neither been decoded nor stored at this point. This only happens when the user decides to open or save an attachment. Only then the function Emai_PopGetAttachment / Emai_ImapGetAttachment is invoked and the decoding job for that attachment is done.

Separate attachment encoding from sending
Working with message objects that are generated before processing a message can also be helpful when sending messages. When composing a message for sending you can choose to generate a message object instead of sending the message using Emai_SmtpSendMessage . Therefore, the function Emai_SmtpComposeMessage is invoked after passing all message values to the plug-in. This function encodes all attachments and composes a message object that can be saved in a container field or exported to disk.

Pre-composed messages can be sent using Emai_SmtpSendMessage at a later point without any encoding needed. Use this approach if separating the encoding from the sending seems optimal for your solution due to performance reasons.

CHAPTER 5
Sending Various Email Formats

Mailit enables you to send a number of different email formats:

Plain Text Sends plain text without any formatting.

Rich Text Sends formatted FileMaker text (including font size, color, style, etc). Mailit generates the HTML code needed for the formatted text automatically.

HTML
Sends HTML message, the message body needs to be HTML code. In order to send HTML emails with Mailit a basic knowledge of HTML is required. Paste HTML code (starting with `<html>` and ending with `</html>`) into the email body. The format is indicated as a parameter when adding a body to a message using `Emai_SmtpAddBody`.
There are numerous applications available that can help to write an HTML code. You basically create a document and these applications generate the HTML code for you:

Microsoft Word for simple layouts Mozilla SeaMonkey – Composer http://www.seamonkey-project.org for all platforms (free)

Before sending the HTML code it is a good idea to check the code against standards to be compatible with the largest amount of email clients. Go to http://validator.w3.org to check your HTML code online.

This service is free of charge and is part of the World Wide Web Consortium, the group that sets standards for the World Wide Web.

HTML with inline image(s)
HTML messages can also contain embedded files such as inline images. The following describes procedure in the `Simple Mass Mail` example file.

Select an attachment in the attachment portal. Define it as `"inline"` in the pop-up menu (beside the actual attachment file). Click the `"CID"` button, which will generate a **Content ID** for this image to be embedded

and places it in the clipboard. i.e.: `MyPicture.jpg@4D5C9F1F.CA30.11D9.A446.000393A54002`
In the HTML code place the following line where the image should be appear:
<img src="cid: `MyPicture.jpg@7760B1DF.BD89.11D9.AD7A.000393A54002` ">

In general other file formats (like sound) can be embedded into emails the same way. However, JPEG, PNG and GIF are the only file formats that are supported by most receiving email clients.

Sending layout: sends a FileMaker layout as email
Mailit lets you send FileMaker layouts (i.e. report or invoice layouts) by converting them into a JPEG or PNG image and embed this image into a message. To email a layout make your script switch to "Preview" mode and then invoke the "Copy" script step which copies an image of the preview mode to clipboard. By using the function `Emai_ImportJpegClipboard` or `Emai_ImportPNGClipboard` you can then retrieve this picture as JPEG or PNG image accordingly, which can be set as an inline image of a message as shown above.
There are two techniques, that can be used to create a layout messages. The simple way, illustrated by the `Layout Sending Single Page` example solution, allows you to build a message body within a single script step, however this way you can send only a single page messages. In order to send multiple pages layout messages you should utilize a sophisticated technique, illustrated by the `Layout Sending Multiple Pages` example solution.

Multi-part Email When sending HTML messages you should be aware that some email clients cannot process this message format. The multi-part email format enables you to specify alternative plain text content when sending messages in HTML format, which is displayed by those receiving applications that cannot handle HTML. After passing your HTML content to the plug-in using `Emai_SmtpAddBody` invoke this function again and set the alternative plain text. The `"format"` parameter for the function needs to be set to the value `"text"` for sending the alternative text. In case if no alternative part was specified, the plug-in will automatically generate a plain text alternative part, based on the HTML message part. Details can be found in the chapter `"External Functions"`.

CHAPTER 6
HTML and Multi-Part Messages

In addition to retrieve plain text messages with Mailit it also enables you to process incoming HTML and multi-part emails with FileMaker.

Receiving HTML Messages

FileMaker is not able to render the HTML code of incoming HTML messages within a field. However, Mailit offers a good alternative of handling incoming HTML emails with FileMaker.

Use the function `Emai_PopGetMessageHTML` to retrieve the HTML code of a message into FileMaker after it has been downloaded by the plug-in. The function `Emai_OpenBrowserHtml` can be invoked to pass this code to the default web browser that renders it.

In addition, the function `Emai_PopGetMessageBody` will return the plain text part of the email message no matter if it has been sent as a plain text or a HTML message. Use this version of the message for reading and processing with FileMaker.

In order to display the message content using the FileMaker native "Web Viewer" use the `Emai_ExportHtml` function. For the best result, save the incoming message object, and use the saved object for the HTML parameter of the `Emai_ExportHtml` function call. In this case the plug-in will automatically rebuild the images reference tree, and store all the inlines accordingly.

Summary

A typical incoming HTML received with Mailit consists of:

1. A plain text version of the message which is shown in FileMaker for reading and further processing.
2. A HTML code version of the message which is also stored in FileMaker and can be opened with the default web browser or internal Web Viewer for rendering.

The inline images of a HTML email will be saved as attachments. When viewing retrieved HTML code using `Emai_OpenBrowserHtml` inline images will not be shown. To view the message with inline images a full message object should be passed to the `Emai_OpenBrowserHtml` function. To retrieve

the full message object the `Emai_PopGetMessageSource` or `Emai_ImapGetMessageSource` functions should be used.

Receiving Multi-Part Messages

If Mailit receives a multi-part message the all plain text message parts are getting merged, and then returned by the plug-in functions `Emai_PopGetMessageBody` or `Emai_ImapGetMessageBody`. The functions `Emai_PopGetMessageHTML` and `Emai_ImapGetMessageHTML` return the first HTML code of the first HTML message part exactly as if you would retrieve an HTML email. Invoke the function `Emai_OpenBrowserHtml` to render this code with the default web browser.

CHAPTER 7
Script Timer

Mailit provides a timer that can trigger any FileMaker script in a custom interval.

A download timer feature has been implemented in the `Personal Mail` example files to demonstrate this functionality. It lets you set a custom interval in which Mailit will check for messages.

The timer function:
Emai_SetupTimer

The timer can be used for any script in a FileMaker file. The parameters needed are the file name, the script name, the script parameter and the interval in which the timer should trigger given script.
In the "Personal Mail" example file the timer is used to receive mail in a certain interval. For more details on `Emai_SetupTimer` consult the "External Function" chapter.

If the user is currently working with a different application other than FileMaker you can use the function `Emai_FlashAppIcon` to inform the user about new mail. This function animates the FileMaker application icon on all platforms.

CHAPTER 8
Script Timer

When sending or receiving messages with Mailit, you can show a status dialog with a progress bar that indicates the processing status to the user. For SMTP connections that utilize the threaded model, Status dialog should be attached to a thread using a corresponding function call parameter.

Use the following functions to set up the status dialog:

Emai_SetupStatusDialog
...specifies the settings of the status dialog. You can allow or disallow a "Cancel" button and customize the title and text of the status dialog.

Emai_SetupProgressMax
...specifies the number of messages to define a maximum for the progress bar of the status dialog.

Emai_ShowStatusDialog
...shows or hides the status dialog

More details on how to use these functions can be found in the "External Functions" chapter.

CHAPTER 9
Result Codes

The plug-in function generated the following result codes. The explanation string is returned together with the result code for easier debugging.

Code	Explanation	Possible Cause
000	"OK"	Works as expected – no problems
001	"Connect failed"	Plug-in was unable to connect to the specified server. Causes could be incorrect server address or problems with network connection.
002	"Connection not confirmed"	Server rejects the connection. It might be busy. Try to connect to server later.
003	"Wrong auth type"	Check for incorrect account data (especially authType parameter in the `Emai_Connect` call)
004	""	Reserved (not returned by plug-in)
005	"Interrupted by user"	User pressed Cancel button in status dialog
006	"Connection reset"	Network connection problem
007	"Active session command failed"	The server returned unexpected result to the last session command sent
008	"Error working with temporary files"	Plug-in was unable to create temporary file. Cause might be low disk space on system volume
009	"Unknown error"	Plug-in produced unexpected behavior. Please refer to Dacons tech support (do not forget to attach log file)
010	"Some recipients not accepted, list of failed recipients follows"	SMTP server rejects one or more recipients and plug-in has been configured to continue sending a message in such a case

Code	Explanation	Possible Cause
011	"Recipient not accepted"	SMTP server rejects a recipient of a message and plug-in has been configured to stop sending in such a case (or you didn't specify any message recipient in TO, CC or BCC message header fields)
012	"Could not greet the server"	SMTP server rejects HELO or EHLO command. Server answer is dumped in log file.
013	"Greeting/Authentication failed"	Check for incorrect account data for SMTP (especially user name and password)
014	"Sender not accepted"	SMTP server does not know sender or you didn't specified sender at all
015	"Data transaction failed"	SMTP server rejects message for some reason. Server reply is dumped in log file.
016	"Error while closing SMTP connection"	SMTP server rejects QUIT command for some reason. Server reply is dumped in log file
020	"Authentication failed"	Check for incorrect account data for IMAP/POP3 (especially user name and password)
021	"Cannot retrieve message list"	POP3 server rejects LIST command. Server reply in dumped in log file
022	"Error while retrieving message"	POP3 server rejects RETR or TOP command. Server reply in dumped in log file)
023	"Error deleting the message"	POP3 server rejects DELE command. Server reply in dumped in log file
024	"Error getting UIDL" (message history)"	POP3 server rejects UIDL command. Server reply in dumped in log file)
026	"Error while closing POP connection"	POP3 server rejects QUIT command for some reason. Server reply in dumped in log file
028	"SSL handshake failed"	The most probably server (SMTP or IMAP/POP3) doesn't support secured connections)
030	"Incorrect option"	One of the parameters passed in external function has incorrect value
031	"Invalid registration information"	Enter registration *exactly* as supplied)
032	""	Reserved (not returned by plug-in)
033	"Not connected"	Connect to server first
034	"No received message"	Call `Emai_ProcessMessageSource` or `Emai_PopRetrieveMessage` first

Code	Explanation	Possible Cause
035	"Requested header field not found" "Missing Sender field" "Missing Recipients field"	Specified field was not found in message header
036	""	Reserved (not returned by plug-in)
037	"Connection timed out"	Connection was timed out. Possible network error occurred
038	"Address probably valid"	Destination mailbox accepts e-mail address so address is probably valid
039	"Address invalid: domain not found"	Domain name in e-mail address cannot be resolved
040	"Address invalid: mailbox not found"	Email address was rejected by destination mailbox
041	"Address could not be checked: reason unknown"	An error occurred while communicating with destination mailbox. Please try to check this email address later.
042	"File not found"	File with specified path doesn't exist.
043	"File i/o error"	An error occurred during reading/writing file operations
044	"File already exists"	File with specified path doesn't exist.
045	"File could not be opened"	Specified file cannot be opened. Most likely this file does not have an application associated with it
046	"Could not receive or process the list of obsolete messages"	The plug-in was unable to retrieve or process the list of messages that should be deleted from the server.
050	"Function call prerequisites are not sufficient"	This function call requires certain conditions to comply the requested functions call.
051	"Wrong charset"	The plug-in was unable to recognize the charset used by the message.
052	"Function call status not available either because it is out of thread context or it hasn't been triggered yet"	`Emai_GetLastResultCode` was unable to retrieve the function call result, because the function hasn't been triggered yet, was attached to a different thread, or doesn't exist.

CHAPTER 10
Spam Protection

Depending of the volume of emails you have to manage, several options exist that help dealing with Spam (unsolicited email).

Mailit Functions

To save you from unsolicited email and to keep your email traffic to an optimum you might want to take advantage of a powerful functions called: `Emai_PopGetMessageInfo` and `Emai_ImapGetMessageInfo`. These functions enable you to check each message that is waiting on the mail server for certain subjects, sizes, recipients or senders before you download it including all of its content and attach- ments. You can use the functions `Emai_PopDeleteMessage` and `Emai_ImapDeleteMessage` to delete messages from server without downloading them.

Rules

Rules are a good way to check incoming email for Subject, From, To, etc. and have FileMaker sort suspicious emails into a "Junk Mail" folder for review (before deleting). The Mailit package comes with the Personal Mail example file that shows how rules can be implemented into your solution.

Server-Side Spam Protection

Most ISPs (Internet Server Provider) offer to manage Spam directly on the mail server. You will either not receive them at all (your ISP will delete them from your email server) or your ISP will mark suspect spam emails as such. Contact your ISP for the service that is best for you.

Client-Side Spam Protection Software

There are numerous applications on the market that can help you deal with Spam. Some virus protection software has spam protection included.

A good source to search for spam applications is:
http://www.snapfiles.com for Windows users

and

http://www.versiontracker.com for Macintosh users

Look for applications that work independent of any email applications.

CHAPTER 11
Anti-Virus Protection

Anti-Virus protection is essential with any regular email client and this is no different when managing emails with Mailit.

There are two possible solutions for virus protection.

Server-Based Virus Scan

Most ISPs (Internet Service Provider) offer server based email virus scan. That means that emails will be scanned for viruses BEFORE you download them. Most ISPs offer this service for a small monthly fee. The advantage of these kinds of online scanners is that they are always up-to-date and require no additional software or administration on your end.

Client-Based Virus Scan

There are numerous good applications on the market that scan the email ports on your PC/Mac for viruses. Some popular applications include:

Norton AntiVirus
http://www.symantec.com/nav/nav_9xnt/features.html

McAfee Virus Scan
http://mcafee-virus.com/mcafee.html

Dr.Web Antivirus
http://www.drweb.com/

An email itself cannot contain any virus but attachments can. It is not recommended to open any attachments that arrive from senders that are not known to you.

Never open any ".exe" attachment files under Windows.

CHAPTER 12
Troubleshooting

Please consider the following points if you are experiencing connection problems with Mailit. The following list provides solutions to the most common issues.

1 Firewall Port Settings:

Mailit uses the IMAP, POP3 and SMTP protocols.
This requires access to port 143 and 110 for receiving, and port 25 for sending emails (SSL connection requires ports 993, 995 and 587/465 accordingly). Make sure your Firewall settings make these ports available.

A. Windows XP SP2:

Windows XP with SP2 installed (and later) requires you to give email applications permission to send email; in particular email port 25 (to send messages). Windows XP pops up a dialog, which asks the user if an application is allowed to access a port. In the case of Mailit the application name "FileMaker" will be shown in this dialog. Give the permission to FileMaker to access network in order to use Mailit.

B. Norton Anti Virus, McAfee and other Anti-Virus applications:

It is important to specify exactly the name of the applications (i.e. "FileMaker Pro.exe", "FileMaker Developer.exe", "FileMaker Advanced.exe", etc) in the settings of Anti-Virus applications to allow FileMaker access to port 25/465/587 (to send messages).

2 Mailit only sends a certain number of emails, then stops:

ISPs set limits on how many emails can be sent per connection to prevent Spam.

You can resolve the issue by adding a disconnect-reconnect script step into your sending script (Mailit will disconnect-reconnect after x messages sent).

This is implemented in *Simple Mass Mail* example file that comes with the Mailit package.

3 Check Your Account Settings:

A.
The most common connection errors occur due to incorrectly entered email account data. Email servers are very unforgiving when it comes to spelling the account/server data incorrectly.
Double-check your account data and the spelling.

As an example:
smpt.mymail.com will not work while smtp.mymail.com will work (note the word "smtp" is misspelled in the first line)

B.
Make sure that you know the user name and password for your email account. The most common mistake here is to enter the email address as user name.
If you are unsure about your user name and password check your regular email client or contact your ISP or system administrator.

C.
Make sure you know if your SMTP server requires authentication. If your server does not require SMTP authentication leave the SMTP user name and the SMTP password fields empty. If your SMTP server requires authentication use the POP username and POP password for the SMTP authentication. If you are unsure about those settings contact your ISP or system administrator.

D.
Make sure you know if your SMTP and IMAP/POP server require a secure connection (SSL) and set your email account accordingly. In most cases this is an additional feature that can be used if needed. Check with your ISP regarding the availability of SSL for your email account.

4 Relaying Denied

"Relaying not allowed" or "denied" means that your SMTP server has a SPAM protection enabled.
You are trying to send with a "From" email address that your SMTP server does not "know". Your "From" address needs to be one that is listed as a legitimate email address.
Enabling SMTP authentication should also resolve this problem, in case if "SMTP authentication" is supported.

5 Check You Scripting

If you experience trouble with your solution, we suggest you try to recreate the issue with one of the example files that come with the Mailit package and compare the scripting of both solutions.

CHAPTER 13
Technical Notes

1 Mailit and FileMaker Runtime Solutions

To use a FileMaker plug-in with a runtime solution you need to create a "Extensions" folder in the runtime application folder. Place a copy of the plug-in in the "Extensions" folder.
Please notice that using the plug-in with a FileMaker Runtime solution requires a Developer license.

2 Microsoft Exchange Server

In order to send and receive email through a Microsoft Exchange Server you will need to install the IMAP or POP3 extension (shipped with the server) on your Exchange Server.
You need to specify the TCP/IP name of your Exchange server computer, or its IP address in both SMTP (Outgoing) and IMAP/POP3 (Incoming) servers address fields. The ports should only be changed, if you are sure about this!
Please also make sure that the Exchange Server SMTP protocol is enabled.

3 Import Emails From Other Email Clients

Mac OS X users:
To transfer emails from Outlook or Entourage into FileMaker we suggest using "Archive X" solution from http://www.softhing.com/eeax/info.html or "Mail to FileMaker Importer" from http://www.automatedworkflows.com/products/software/mail-to-filemaker-importer/

Windows users:
Using Outlooks export function export the emails into a format that your FileMaker version accepts for import (different Outlook version have different options).
FileMaker 10-14 accepts among others "Tab-separated" text files or "Excel" files.

Another option to transfer the message from email client application to FileMaker is moving the messages from local folders to IMAP folders, and then downloading the IMAP folders using the Mailit 6 plug-in.