

Mailit 11

FileMaker Plug-in Manual



Dacons

©2026 Dacons UG (haftungsbeschränkt), Am Fürstenweiher 31, 69118 Heidelberg, Germany. All rights reserved.

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.

This manual, as well as the software described in it, is furnished under a license and may not be copied, photocopied, reproduced, translated, or converted to any electronic or machine-readable form in whole or in part without prior written approval of Dacons UG (haftungsbeschränkt). Dacons assumes no responsibility or liability for any errors or inaccuracies that may appear in this manual.

Please send feedback to info@dacons.net

Website: <http://www.dacons.net>

This version of the manual was updated on May 29, 2026.

CONTENTS

INTRODUCTION	7
What is Mailit 11	7
Feature List	7
Quick Start	8
Installing plug-ins from database file for FileMaker® Pro on Apple macOS and MS Windows platforms	9
Installing plug-ins from database file for FileMaker® Server on Apple macOS / MS Windows / Linux platforms	9
Manual installation by copying the plug-in file for FileMaker® Pro on Apple macOS and MS Windows platforms	11
Manual installation by copying the plug-in file for FileMaker® Server on Apple macOS and MS Windows platforms	13
Manual installation by copying the plug-in file for FileMaker® Server on Linux platforms	15
Example Files.....	18
Path formats	19
REST API Support.....	20
OAuth 2.0	23
HTTPS Certificate Validation.....	23
 CHAPTER 1	 25
Fields to Receive Email	25
Functions to Receive Email	27
Scripts to Receive Emails.....	29
Scripts to Receive Email in a Thread	31
 CHAPTER 2	 34
Fields to Send Email.....	34
Functions to Send Email	36
Scripts to Send Email	37
Scripts to Send Email in a Thread.....	38
 CHAPTER 3	 41
Accessing external functions.....	41
Email_GetLastResult.....	42
Email_Version.....	43
Email_Connect.....	43
Email_Disconnect	45
Email_CreateThread	45

Email_DestroyThread	46
Email_GetThreadQueueItemsNumber	46
Email_ProcessMessageSource	47
Email_GetFileName	47
Email_GetFileSize	47
Email_GetFileIcon	48
Email_ShowFolderDialog	48
Email_ShowFileDialog	49
Email_ImportFile	50
Email_ExportFile	51
Email_GetUserFolder	52
Email_ConvertPath	52
Email_SetupTimer	53
Email_PerformScript	53
Email_ResetMessageHistory	53
Email_SetupStatusDialog	54
Email_SetupProgressMax	55
Email_SetCurrentProgress	55
Email_ShowStatusDialog	56
Email_FlashAppIcon	56
Email_SetupLog	56
Email_SetupTranscriptSize	57
Email_GetLastSessionTranscript	57
Email_OpenBrowserHtml	58
Email_ExportHtml	59
Email_FmTextToHtml	60
Email_HtmlToFmText	60
Email_SetDefaultCharset	61
Email_GetDefaultCharset	61
Email_RegisterSession	61
Email_RegisterServer	62
Email_UnregisterPlugin	62
Email_ImportHTMLFile	62
Email_ImportHTMLAttachmentCount	63
Email_ImportHTMLAttachmentPath	63
Email_LocalWebServerStart	63
Email_LocalWebServerStop	65
Email_LocalWebServerStatus	65
Email_URLRequest	65
Email_JSONGetElement	66
Email_ExecuteSQL	66

Email_GetLastDDLResult	67
Email_GetMaessageFromAccount	67
Email_awsGetAccountInfo	68
Email_awsSetCredentials	69
Email_awsSendMessage	70
Email_awsSendRawMessage	71
Email_awsUploadResource	72
Email_awsDownloadResource	72
Email_awsDeleteResource	73
Email_awsGetBucketsList	73
Email_awsCreateBucket	73
Email_awsRemoveBucket	73
Email_awsGetBucketListObjects	74
Email_awsGetBucketPolicy	74
Email_awsPutBucketPolicy	75
Email_awsDeleteBucketPolicy	75
Email_awsGetBucketPublicAccessBlock	76
Email_awsPutBucketPublicAccessBlock	76
Email_awsGetSendStatistics	77
Email_awsGetSendQuota	78
Email_awsCreateContactList	79
Email_awsGetContactList	80
Email_awsDeleteContactList	80
Email_awsListContactLists	80
Email_awsUpdateContactList	81
Email_awsGetContact	82
Email_awsGetListContacts	83
Email_awsCreateContact	84
Email_awsUpdateContact	84
Email_awsDeleteContact	85
Email_awsAddContactToSuppressionList	85
Email_awsRemoveContactFromSuppressionList	85
Email_GetCurrentUTCOffset	85
Email_ConvertDateRFC2822toFMDatetime	86
Email_SmimeMessageEncrypt	87
Email_SmimeMessageDecrypt	88
Email_SmimeMessageVerifySign	88
Email_SmimeMessageConvertToDetached	89
Email_PGPMMessageEncrypt	90
Email_PGPMMessageDecrypt	91
Email_PGPMMessageVerifySign	92

Email_GetEncryptionType.....	92
Email_PopGetHistory	93
Email_PopGetMessageCount	93
Email_PopGetMessageInfo	94
Email_PopRetrieveMessage.....	94
Email_PopRetrieveAllMessages	94
Email_PopGetMessageSource.....	95
Email_PopGetMessageSourceAsText	95
Email_PopGetContactsCount	95
Email_PopGetContact	96
Email_PopGetMessageHeader	96
Email_PopGetMessageBody	96
Email_PopGetMessageHTML.....	97
Email_PopGetMessageCharset.....	97
Email_PopGetMessagePriority.....	97
Email_PopGetMessageField	97
Email_PopGetRecvDate.....	98
Email_PopGetRecvTime	98
Email_PopGetRecvTimestamp.....	98
Email_PopGetMessageSize	98
Email_PopDeleteMessage	99
Email_PopGetAttachmentCount	99
Email_PopGetAttachment.....	100
Email_PopExportAttachment	101
Email_PopGetAttachmentName	101
Email_PopGetAttachmentSize	102
Email_PopGetAttachmentIcon	102
Email_PopGetAttachmentCID	103
Email_ImapListFolders	104
Email_ImapSelectFolder	104
Email_ImapGetMessageCount.....	105
Email_ImapRetrieveAllMessagesCount	105
Email_ImapGetMessageInfo.....	106
Email_ImapRetrieveMessage	106
Email_ImapRetrieveAllMessages	106
Email_ImapGetMessageUID	107
Email_ImapGetMessagesByFlags.....	107
Email_ImapGetDeleteUIDList	107
Email_ImapGetDeleteUIDListInFolder**	108
Email_ImapGetMessageSource	108
Email_ImapGetMessageSourceAsText	109

Email_ImapGetContactsCount	109
Email_ImapGetContact	109
Email_ImapGetMessageHeader	110
Email_ImapGetMessageBody	110
Email_ImapGetMessageHTML	110
Email_ImapGetMessageCharset	110
Email_ImapGetMessagePriority	111
Email_ImapGetMessageField	111
Email_ImapGetMessageFlags	111
Email_ImapSetMessagesFlags	112
Email_ImapRemoveMessagesFlags	112
Email_ImapGetRecvDate	113
Email_ImapGetRecvTime	113
Email_ImapRecvTimestamp	113
Email_ImapGetMessageSize	113
Email_ImapDeleteMessage	114
Email_ImapGetAttachmentCount	114
Email_ImapGetAttachment	115
Email_ImapExportAttachment	116
Email_ImapGetAttachmentName	116
Email_ImapGetAttachmentSize	117
Email_ImapGetAttachmentCID	117
Email_ImapGetAttachmentIcon	118
Email_ImapCreateFolder	118
Email_ImapRenameFolder**	119
Email_ImapDeleteFolder**	119
Email_ImapUploadMessageToFolder	120
Email_ImapMoveMessage	120
Email_ImapSetSearch	120
Email_ImapClearSearch	121
Email_ImapGetQuota	122
Email_ImapGetFolderStatus	122
Email_ImapGetQuota	123
Email_SmtpDestroyMessage	124
Email_SmtpImportMessageParameters	124
Email_SmtpResetMessage	125
Email_SmtpGenerateMessageID	125
Email_SmtpAddMessageField	126
Email_SmtpAddContact	126
Email_SmtpSetMessagePriority	127
Email_SmtpAddBody	128

Email_ImportJpegClipboard	129
Email_ImportPNGClipboard	129
Email_SmtpGenerateContentID.....	129
Email_SmtpAddAttachment.....	130
Email_SmtpSendMessage	131
Email_SmtpComposeMessage	131
Email_SmtpFalsifyAddress	132
CHAPTER 4	133
Receiving Attachments.....	133
Sending Attachments.....	134
Advanced Message Processing	135
CHAPTER 5	137
Sending Various Email Formats	137
CHAPTER 6	139
Receiving HTML Messages	139
Receiving Multi-Part Messages.....	140
CHAPTER 7	141
Script Timer	141
CHAPTER 8	142
Status Dialog	142
CHAPTER 9	143
Result Codes	143
CHAPTER 10	150
Mailit Functions.....	150
Rules.....	150
Server-Side Spam Protection	150
Client-Side Spam Protection Software	150
CHAPTER 11	152
Server-Based Virus Scan.....	152
Client-Based Virus Scan	152
CHAPTER 12	153
Troubleshooting	153
CHAPTER 13	155
Technical Notes	155

INTRODUCTION

Mailit 11 Features

What is Mailit 11

Mailit is a powerful FileMaker® Pro plug-in that lets you send, receive and manage email with your new or existing FileMaker® 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 For newsletters, Mailit 11 powers DSGVO double opt-in leveraging FileMaker® Server.

- 4 Integrate professional mass email editing features including HTML templates and layout tools (e.g. multi column layout) via Mailit and FileMaker® Web Viewer. Choose to encode images in Base64. Send mass mailings in the background or leverage sending via FileMaker® Server (including TLS 1.2/1.3) to keep your FileMaker® app responsive.
- 5 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.
- 6 Connect your FileMaker® online solution or your database-driven web site to Mailit to send highly customized and personalized newsletters and notifications.
- 7 Create email discussion list solutions with the database and administration power of FileMaker®
- 8 Manage and parse information received from online forms with FileMaker®.
- 9 In addition to SMTP, Mailit lets you relay personalized emails via Amazon SES®. Processing happens ultra fast even for larger amounts of messages. Mailit also supports Amazon SES statistics.
- 10 Import raw message (as text) into FileMaker®.
- 11 Add multiple attachments via drop zones.
- 12 Mailit provides a new powerful HTML editors that loads via the FileMaker® Web Viewer. This editor behaves like a real email application when composing messages. Pictures can be added in-line via drag & drop.
- 13 Mailit supports OAuth 2.0 and SSL, TLS 1.2/1.3 for SMTP authentication and secure connections.
- 14 Mailit supports encryption standards such as PGP and S/MIME to encrypt emails and attachments using the recipient's public key.

Find out more about customer solutions at:

<http://www.dacons.net/fmplugins/mailit11/customerstories/>

Quick Start

This chapter will get you started with Mailit 11 in minutes. All you need is the package you downloaded from the Internet and a supported copy of FileMaker® 15-26 (64-bit only) .

System Requirements for Windows (64-bit only):

FileMaker Pro/Advanced/Runtime 15–18 (64-bit)

FileMaker Pro 19–26, including Claris FileMaker 2026

FileMaker Server 15–26, including Claris FileMaker Server 2026

Requires: a Windows version supported by the installed FileMaker version.

System Requirements for Macintosh:

FileMaker Pro/Advanced/Runtime 16–18

FileMaker Pro 19–26, including Claris FileMaker 2026

FileMaker Server 16–26, including Claris FileMaker Server 2026
Requires: a macOS version supported by the installed FileMaker version.

System Requirements for Linux:

FileMaker Server 19.5–26, including Claris FileMaker Server 2026
Supported: Ubuntu LTS versions supported by the installed FileMaker Server release. AMD64 and ARM64 are supported where supported by the installed FileMaker Server release.

IMAP/POP3/SMTP Account

Note that MailIt needs an IMAP/POP3/SMTP email account.

It will NOT work with web based basic email accounts.

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.

Installing plug-ins from database file for FileMaker® Pro on Apple macOS and MS Windows platforms

- 1 Download the plug-in archive from the link: <https://www.dacons.net/downloads>.
- 2 Extract all files of the Mailit package to any location on your machine.
- 3 In the unpacked archive, find the file ../Plug-in/Install Plug-in.fmp12 and open the file using FileMaker® Pro.
- 4 In the file that opens, click the "Install Plug-in" button to install the plug-in.
- 5 After installing the plug-in, click the "Install Assets" button to install additional plug-in files, the files are needed for OAuth authentication and web editors. Note: the additional files have to be installed for the plug-in to function correctly.
- 6 If you follow these steps, you can proceed with testing the examples that go in the archive with the plug-in in the "Email Apps for FileMaker®" folder.

Installing plug-ins from database file for FileMaker® Server on Apple macOS / MS Windows / Linux platforms

- 1 Download the plug-in archive from the link: <https://www.dacons.net/downloads>.
- 2 Extract all files of the Mailit package to any location on your machine.

- 3 In the unpacked archive, find the file ../Plug-in/Install Plug-in.fmp12 and open the file using FileMaker® Pro.
- 4 In the file settings set a password for the file using the FileMaker menu “File -> File Options...”.
- 5 Using menu “File -> Sharing -> Upload to host...” upload the file to the FileMaker Server.
- 6 Open the FileMaker® Server Admin Console (<http://localhost:16001/admin-console/signin>), navigate to the Connectors -> Plug-ins tab and enable the options: FileMaker Script Engine Plug-ins and Install Plug-in File Script Step.
If you want to install the Web Publishing plug-in, set Web Publishing Plug-ins and Install Plug-in File Script Step to Enabled.
- 7 Open the database file “Install Plug-in.fmp12” from the server using FileMaker® Pro and click the “Install Plug-in on Server” button to install the plug-in.

To install the plug-in in WebDirect, open Open Launch Center, then open the “Install Plug-in.fmp12” file and click the “Install Plug-in on Server” button to install the plug-in.

The new plug-in should be visible in the list of plug-ins which FileMaker® Server can use.

NOTE: If you are updating a plug-in and during the upgrade process you are unable to update the plug-in, disable the plug-in in the Admin Console and delete the plug-in file from the plug-in folder on the server. After completing the steps, repeat the installation of the plug-in.

MailIt Plug-ins, installed on FileMaker® Server as server-side or web-side plug-in, write a log file to help you determine what is going on, like for example to see if the plug-in is properly installed.

On Apple macOS you can find the logs here:

```
/Library/FileMaker Server/Logs/Mailit_FM[Version]Server.log
```

On MS Windows you can find the log here:

```
C:\Program Files\FileMaker\FileMaker Server\Logs  
\Mailit_FM[Version]Server.log
```

By default, the plugin log file is disabled. In order to enable the plugin log file, use the server-side examples from the plugin archive. Upload one or more sample files from the “Get Started with Mailit and FileMaker® Server” folder to your FileMaker® Server. Open the example from the server using FileMaker® Pro. In the file, on the Account layout, there is a “Log file” tab, in the “Log file creation” field, select the “Append” value and click the “Apply settings in FMS” button. This will activate the plugin log file on the server.

To register the plugin on the server, use the plugin function Email_RegisterServer or use the server-side examples from the archive that came with the plugin. It is sufficient to register the plugin once - do not add

the `Email_RegisterServer` function to the scripts that are executed every time the database is opened on the server. You can check the registration of the plugin using the plugin log file. Below, you find an example of an entry in the log file for that case the plugin is not registered:

```
[ERROR] - Registration info file is not found
```

And here is an example of an entry in the log file for the case the plugin has been registered successfully:

```
[INFO] - Registration information loaded from preferences.  
[INFO] - Registration info: 10 - user(s), server - yes,  
redistributable - no.  
[INFO] - GUI disabled.
```

Commands for stopping and starting FileMaker® Server.

On Apple macOS platforms, to stop FileMaker® Server, run the following command in Terminal.app:

```
sudo launchctl stop com.filemaker.fms
```

To start FileMaker® Server, run the command:

```
sudo launchctl start com.filemaker.fms
```

On MS Windows platform, to stop FileMaker® Server, run the command in the `cmd.exe` application (run as administrator):

```
net stop "FileMaker Server"
```

To start FileMaker Server, run the command:

```
net start "FileMaker Server"
```

On Linux platforms, to stop FileMaker® Server:

Start FileMaker Server processes:

```
FileMaker Server only: sudo service fmshelper start
```

```
Claris Server only: sudo service claris start
```

Stop FileMaker Server processes:

```
FileMaker Server only: sudo service fmshelper stop
```

```
Claris Server only: sudo service claris stop
```

Manual installation by copying the plug-in file for FileMaker® Pro on Apple macOS and MS Windows platforms

- 1 Download the plug-in archive from the link: <https://www.dacons.net/downloads>.

- 2 Extract all files of the Mailit package to any location on your machine.
- 3 In the unpacked archive, find the file `../Plug-in/Install Plug-in.fmp12` and open the file using FileMaker® Pro.
Move the mouse cursor over the required container field and right-click, select “Export Field Contents ...” from the context menu, select the location where to save the plug-in file.
For Apple macOS, the plug-in file has the name: Mailit.fmp12.
For MS Windows, the plug-in file has the name: Mailit.fmx (for 32bit FileMaker® Pro), Mailit.fmx64 (for 64bit FileMaker® Pro).
Notes: FileMaker® Pro can only install plug-ins that match the FileMaker® Pro application’s architecture. For example, the 64bit version of FileMaker® Pro can only install 64bit plug-ins. Use the `Get(ApplicationArchitecture)` function to determine the current application’s architecture.
- 4 Open FileMaker® and go to “Preferences” on the “Plug-ins” tab, click the “Reveal Plug-in Folder” button, the “Extensions” directory will open.
- 5 Copy the Mailit plug-in file into the Extensions directory.
- 6 Restart FileMaker®, and again go to “Preferences” on the “Plug-ins” tab, check that the Mailit plugin is included in the list of plug-ins, and that the plug-in’s name is checked (i.e., plug-in is installed and enabled).
- 7 Once you have installed and enabled the plug-in, install additional plug-in files, open the `../Plug-in/Install Plug-in.fmp12` file and click the “Install Assets” button.
- 8 If you follow these steps, you can proceed with testing the examples that go in the archive with the plug-in in the “Email Apps for FileMaker®” folder.

FileMaker® has advanced options for installing plug-ins. If you are installing or updating a plugin, you can copy the plugin file to a specific directory. By doing this, you can control features such as the availability of the plugin in different versions of FileMaker®.

A plug-in directory next to the application, for all system users, is available for FileMaker® Pro 15 - 18 Pro / Advanced:

- For Apple macOS:
`/Applications/FileMaker Pro <version> Advanced/Extensions/`
- For MS Windows:
`C:\Program Files (x86)\FileMaker
\FileMaker Pro <version> [Advanced]\Extensions (for 32-bit FileMaker)`

`C:\Program Files\FileMaker
\FileMaker Pro <version> [Advanced]\Extensions (for 64-bit FileMaker)`

Note: Starting with FileMaker® 19 for OSX, the Extensions folder is not located next to the application so use other directories to install plug-ins.

Directory for installing plug-ins:

- For Apple macOS:

```
/Users/user_name/Library/Application Support/FileMaker  
/FileMaker Pro/<version>/Extensions/
```

```
/Users/user_name/Library/Application Support/FileMaker  
/FileMaker Pro Advanced/<version>/Extensions/
```

- For MS Windows:

```
C:\users\user_name\AppData\Local\FileMaker  
\FileMakerPro\<version>\Extensions\
```

```
C:\users\user_name\AppData\Local\FileMaker  
\FileMakerProAdvanced\<version>\Extensions\
```

The directory for installing plug-ins for independent of the FileMaker® version:

- For Apple macOS:

```
/Users/user_name/Library/Application Support/FileMaker/Extensions/
```

- For MS Windows:

```
C:\Users\user_name\AppData\Local\FileMaker\Extensions\
```

Note: Whichever directory you choose to install plug-ins, only one instance of the plug-in should be installed.

Manual installation by copying the plug-in file for FileMaker® Server on Apple macOS and MS Windows platforms

- 1 Download the plug-in archive from the link: <https://www.dacons.net/downloads>. The archive will contain plug-in files for Mac OS and MS Windows platforms.
- 2 Extract all files of the Mailit package to any location on your machine.
- 3 In the unpacked archive, find the file ../Plug-in/Install Plug-in.fmp12 and open the file using FileMaker® Pro.
Move the mouse cursor over the required container field and right-click, select “Export Field Contents ...” from the context menu, select the location where to save the plug-in file. For Apple macOS, the plug-in file has the name: Mailit.fmpug-in.
For MS Windows, the plug-in file has the name: MailIt.fmx (for 32bit FileMaker® Server), MailIt.fmx64 (for 64bit FileMaker® Server).

- 4 Copy the MailIt Plug-in to the appropriate FileMaker® Server Extensions folder.

The path of the folder where you need to place a copy of plug-in is on macOS:

```
/Library/FileMaker Server/Database Server/Extensions
```

On MS Windows the path is:

```
C:\Program Files\FileMaker\FileMaker Server\Database Server  
\Extensions
```

You can also install the plug-in for the Web publishing engine by copying the plug-in file to the directory on macOS:

```
/Library/FileMaker Server/Web Publishing/publishing-engine/wpc  
/Plugins
```

On MS Windows the path is:

```
C:\Program Files\FileMaker\FileMaker Server\Web Publishing  
\publishing-engine\wpc\Plugins
```

- 5 After you placed a copy, you need to enable server-side plug-ins with the FileMaker® Server Admin console, which is accessible from the browser. Open your browser and go to:
<http://localhost:16001/admin-console/signin>. Go to "Database Server" and then the Server Plug-Ins tab. Make sure you enable the checkbox "Enable FileMaker Server to use plug-ins".

The new plug-in should be visible in the list of plug-ins which FileMaker® Server can use.

If the plug-in is not visible in the list of plug-ins or not enabled after installation try restarting FileMaker® Server. Commands for Stop/Start FileMaker® Server are provided at the end of this section.

NOTE: You may need to stop and restart FileMaker® Server in order for new plug-ins to appear in this list. Also stop FileMaker® Server if you are replacing a plug-in with a new version.

Once the plug-in appears don't forget to enable the plug-in via the checkbox next to plug-in. FileMaker® Server is now set-up to use the plug-in.

If the plug-in fails to run, see which user is running FileMaker® Server. In the file manager, open the properties of the MailIt plug-in file, on the "Security" tab, check that the user from whom the server is running has permissions to read and write to the disk. If the user is not in the list, add this user and give him read and write permissions to the disk.

MailIt Plug-ins, installed on FileMaker® Server as server-side or web-side plug-in, write a log file to help you determine what is going on, like for example to see if the plug-in is properly installed.

On Apple Mac OS you can find the logs here:

```
/Library/FileMaker Server/Logs/MailIt_FM[Version]Server.log
```

On MS Windows you can find the log here:

```
C:\Program Files\FileMaker\FileMaker Server\Logs
```

```
\Mailit_FM[Version]Server.log
```

By default, the plugin log file is disabled. In order to enable the plugin log file, use the server-side examples from the plugin archive. Upload one or more sample files from the "Get Started with Mailit and FileMaker® Server" folder to your FileMaker® Server. Open the example from the server using FileMaker® Pro. In the file, on the Account layout, there is a "Log file" tab, in the "Log file creation" field, select the "Append" value and click the "Apply settings in FMS" button. This will activate the plugin log file on the server.

To register the plugin on the server, use the plugin function `Emai_RegisterServer` or use the server-side examples from the archive that came with the plugin. It is sufficient to register the plugin once - do not add the `Emai_RegisterServer` function to the scripts that are executed every time the database is opened on the server. You can check the registration of the plugin using the plugin log file. Below, you find an example of an entry in the log file for that case the plugin is not registered:

```
[ERROR] - Registration info file is not found
```

And here is an example of an entry in the log file for the case the plugin has been registered successfully:

```
[INFO] - Registration information loaded from preferences.
[INFO] - Registration info: 10 - user(s), server - yes,
redistributable - no.
[INFO] - GUI disabled.
```

Commands for stopping and starting FileMaker® Server.

On Apple Mac OS platforms, to stop FileMaker® Server, run the following command in Terminal.app:

```
sudo launchctl stop com.filemaker.fms
```

To start FileMaker® Server, run the command:

```
sudo launchctl start com.filemaker.fms
```

On MS Windows platform, to stop FileMaker® Server, run the command in the cmd.exe application (run as administrator):

```
net stop "FileMaker Server"
```

To start FileMaker Server, run the command:

```
net start "FileMaker Server"
```

Manual installation by copying the plug-in file for FileMaker® Server on Linux platforms

- 1 Download the plugin archive from the link: <https://www.dacons.net/downloads>. The archive will contain plugin files for macOS / MS Windows / Linux platforms.

- 2 Extract all files of the Mailit package to any location on your machine.
- 3 In the unpacked archive, find the file `../Plug-in/Install Plug-in.fmp12` and open the file using FileMaker® Pro.
Move the mouse cursor over the required container field and right-click, select “Export Field Contents ...” from the context menu, select the location where to save the plug-in file.
- 4 Upload the MailIt.fmx plug-in file to the FileMaker® Server extensions folder:

```
"/opt/FileMaker/FileMaker Server/Database Server/Extensions/MailIt.fmx"
```
- 5 After you’ve copied the Mailit plug-in to the Extensions folder, you need to change the file permissions. For the “fmsadmin” group, set the “rwx” permissions, which will allow the plug-in to work with files of the local file system. To update permission run the following commands in the system terminal:
Change group owner:

```
$ sudo chgrp fmsadmin "/opt/FileMaker/FileMaker Server/Database Server/Extensions/MailIt.fmx"
```


Set permission of “group” users:

```
$ sudo chmod g=rwx "/opt/FileMaker/FileMaker Server/Database Server/Extensions/MailIt.fmx"
```


Check the plug-in file permissions:

```
$ sudo ls -l "/opt/FileMaker/FileMaker Server/Database Server/Extensions/MailIt.fmx"
```
- 6 After you placed a copy, you need to enable server-side plug-ins with the FileMaker® Server Admin console, which is accessible from the browser. Open your browser and go to:
<http://localhost:16001/admin-console/signin>.
Go to “Database Server” and then the Server Plug-Ins tab. Make sure you enable the checkbox “Enable FileMaker® Server to use plug-ins”

The new plug-in should be visible in the list of plug-ins which FileMaker® Server can use.

NOTE: You may need to stop and restart FileMaker® Server in order for new plug-ins to appear in this list. Also stop FileMaker® Server if you are replacing a plug-in with a new version. You do not need to restart all server processes, it is enough to restart the “fmsased” process, which is responsible for working with plug-ins. To do this, run the commands in the console:

- 1 Find the PID of the “fmsased” process that is responsible for working with plug-ins:

```
$ ps aux | grep fmsased
```


or you can use another command:

```
$ pidof fmsased
```
- 2 Force stop the “fmsased” process, do not worry, the process will start itself:

```
$ sudo kill [PID]
```

Once the plug-in appears don’t forget to enable the plug-in via the checkbox next to each plug-in. FileMaker Server is now set-up to use the plug-in as a server-side plug-in.

To check plugin registration and debug first connections, use the plugin log file. By default, the plugin log file is disabled, in order to enable the plugin log file, use the server-side examples from archive that came with the plugin. Upload one or more sample files from the "Get Started with Mailit and FileMaker® Server" folder to your FileMaker® Server. Open the example file from the server using FileMaker® Pro. In the example file, on the Account layout, there is a "Log file" tab, in the "Log file creation" field, select the "Append" value and click the "Apply settings in FMS" button. This will activate the plugin log file on the server.

The plugin creates a log file in the directory `/opt/FileMaker/FileMaker Server/Logs/`. The file name depends on the version of Filemaker Server and matches the pattern `"Mailit_FM[Version]Server.log"`. With the following command you can view the contents of the directory for the log files:

```
ls "/opt/FileMaker/FileMaker Server/Logs/"
```

If the list contains a file named `"Mailit_FM[Version]Server.log"` then the plugin is running and the log file is enabled.

You can view the contents of the log file with the command:

```
tail -n 100 "/opt/FileMaker/FileMaker Server/Logs  
/MailIt_FM19Server.log"
```

To register the plugin on the server use the plugin function `Emai_RegisterServer` or use the server-side examples from the archive that came with the plugin. It is sufficient to register the plugin once, do not add the `Emai_RegisterServer` function to the scripts that are executed every time the database is opened on the server. You can check the registration of the plugin using the plugin log file. Below, please find an example of an entry in the log file if the plugin is not registered:

```
[ERROR] - Registration info file is not found
```

And here is an example of an entry in the log file if the plugin is successfully registered:

```
[INFO] - Registration information loaded from preferences.  
[INFO] - Registration info: 10 - user(s), server - yes,  
redistributable - no.  
[INFO] - GUI disabled.
```

Reasons why the Mailit plug-in may not run on Linux Server

The plug-in cannot start if it does not have write access to directories:

```
/opt/FileMaker/FileMaker Server/Logs/ - used to create a log file of plug-in actions;  
/opt/FileMaker/FileMaker Server/conf/ - used to store settings, license file and POP  
message download history;  
/tmp/ - used to store temporary files.
```

When you installed FileMaker® Server with standard settings, the "fmsadmin" group should have been created. Your current Linux system user must be a member of the "fmsadmin" group. In order for the plug-in to run, the plug-in file must have permissions that allow writing to disk for the group and user from which FileMaker® Server is running.

If you run the plugin without the correct access rights, the plugin will create a settings file on behalf of a different user. After that, it will not be able to start because it will not be able to access the settings file. To fix this, follow these steps:

- 1 Disable the plugin through the FileMaker® Server Web Management Interface.
- 2 Set the permissions for the plugin file using the following command in Terminal:

```
sudo chgrp fmsadmin '/opt/FileMaker/FileMaker Server/  
Database Server/Extensions/MailIt.fmx' && sudo chmod  
g=rwx '/opt/FileMaker/FileMaker Server/Database Server/  
Extensions/MailIt.fmx'
```
- 3 Remove the directory '/opt/FileMaker/FileMaker Server/conf/net.dacons.mailit' with the command:

```
sudo rm -r '/opt/FileMaker/FileMaker Server/conf/net.dacons.mailit'
```
- 4 Enable the plugin through the FileMaker Server Management Web interface.

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 11 requires paths of files and folder to be specified in a certain format. To make life easier, Mailit 11 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 11 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
```

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

```
filelinux:/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 11 does not support alternative path this way. It allows you to specify exactly one path at a time only to avoid ambiguities.

Some Mailit 11 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/
```

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

```
filelinux:/directoryName/
```

Mailit 11 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 `Email_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*.

REST API Support

Description

Starting from MailIt 11, the plug-in supports REST API access for modern mail providers.

The following REST providers are currently supported:

- gmailRest — Gmail REST API
- graphRest — Microsoft Graph API for Microsoft 365 / Outlook

This allows MailIt to work with modern cloud mail services over HTTPS, without relying only on classic IMAP and SMTP connections.

Compatibility

To preserve compatibility with older FileMaker solutions, REST support was added to the existing IMAP and SMTP function set instead of introducing separate REST-only functions.

This means:

- IMAP-style functions can also work through REST
- SMTP send functions can also work through REST
- migration to REST often requires only connection and authentication changes

Connection Types

Use the following values in `Email_Connect`:

- gmailRest
- graphRest

Typical REST connection uses OAuth authentication.

The OAuth access token must be passed in the password parameter and `authType` must be set to `oauth`.

Example:

- `Email_Connect ( "gmailRest" ; "" ; userName ; accessToken ; "oauth" )`
- `Email_Connect ( "graphRest" ; "" ; userName ; accessToken ; "oauth" )`

The server parameter may be left empty for REST connections or can optionally be used to specify a custom REST API endpoint. If empty, MailIt uses built-in provider endpoints automatically.

Quick Migration

Migration from IMAP or SMTP to REST is typically straightforward:

1. Replace connection type in `Email_Connect`

2. Replace login/password with OAuth access token
3. Keep existing message processing logic unchanged

Example:

Old style:

```
• Emai_Connect ( "imap" ; server ; userName ; password ; "login" )  
• Emai_ImapRetrieveMessage ( messageID )  
• Emai_Connect ( "smtp" ; server ; userName ; password ; "login" )  
• Emai_SmtpSendMessage
```

REST style:

```
• Emai_Connect ( "gmailRest" ; "" ; userName ; accessToken ; "oauth" )  
• Emai_ImapRetrieveMessage ( messageID )  
• Emai_SmtpSendMessage
```

Important Notes

Although the same functions are used, REST providers may differ from IMAP:

- folders may map to labels or provider-specific folders
- some IMAP flags may not exist in REST
- some folder status values may be unavailable
- message identifiers differ from IMAP UIDs

Message Identifiers and First Synchronization

When an existing IMAP-based solution is moved to REST API access, message identifiers should be treated as provider-specific values. IMAP UID values and REST message identifiers are not guaranteed to be the same. Because of this, the first synchronization after switching an account from IMAP to REST can return messages that already exist in the database if the solution only compares old IMAP identifiers with new REST identifiers.

To avoid downloading the complete mailbox during the first synchronization, the receiving script should limit the initial message search by date. The example file `Simple Receiving.fmp12`, included in the plug-in archive, already contains account settings for this purpose. In the account connection settings, the `Initial synchronization date` section contains the fields `Sync by latest date` and `Download from date`. These settings can be used to define the earliest message date that should be downloaded during the first synchronization.

Internally, this date restriction is applied with the `Emai_ImapSetSearch` function. The function is called after the IMAP or REST connection has been established and before retrieving the message count or downloading messages. The search expression uses the `SINCE` search key followed by the date from which messages should be downloaded. The date must be passed in RFC 2822 format.

Correct call order:

- `Emai_Connect`
- `Emai_ImapSelectFolder`
- `Emai_ImapSetSearch`
- `Emai_ImapGetMessageCount ( uidl )`
- message retrieval functions

Parameter `uidl` represents the list of known message identifiers stored in the database.

- During the first initialization, `uidl` is typically empty
- On subsequent calls, it should contain only message identifiers already stored locally
- When used together with `Emai_ImapSetSearch`, the list should correspond to messages within the selected date range

This allows the plug-in to avoid downloading duplicate messages and improves synchronization performance.

For example, if the account has an initial synchronization date stored in `$accountsRunningDate`, the receiving script can build and apply the search criteria as follows:

```
Set Variable [ $search; Value:"" ]
```

```
If [ not IsEmpty ( $accountsRunningDate ) ]
```

```
    Set Variable [ $search; Value:$search & " SINCE " & CF_FMDateToRFC2822( $accountsRunningDate ) ]
```

```
    Set Variable [ $search_result; Value:Emai_ImapSetSearch ( $search ) ]
```

```
End If
```

Example of a direct search setup:

- `Emai_ImapSelectFolder ( "Inbox" )`
- `Emai_ImapSetSearch ( "SINCE 01-Jan-2026" )`
- `Emai_ImapGetMessageCount ( uidl )`
- `Emai_ImapRetrieveAllMessages`

This approach is especially important when the mailbox already contains a large number of messages. For the first run after switching to REST, the `Download from date` value should normally be set to the date from which the solution really needs to import messages. After the first synchronization has completed, the solution can continue to use its normal duplicate-detection logic and stored message identifiers for subsequent synchronizations.

The `SINCE` search condition limits the messages returned by the server before download starts. This is more efficient than downloading all messages and filtering them locally. It also reduces the chance of duplicate records during migration, because older messages that were already downloaded through IMAP can be intentionally excluded from the first REST synchronization.

OAuth 2.0

REST connections require OAuth 2.0 authentication.

Detailed instructions are included in the plug-in distribution:

- OAuth 2.0 General.pdf
- OAuth 2.0 Google.pdf
- OAuth 2.0 Microsoft.pdf

These documents are located in the root folder of the plug-in archive.

Summary:

1. Create application (Google Cloud / Azure)
2. Obtain Client ID and Client Secret
3. Get Access Token and Refresh Token
4. Use Access Token in `Emai_Connect`

HTTPS Certificate Validation

Mailit uses secure HTTPS connections for operations such as OAuth authentication, REST API access, loading remote HTML content, and downloading remote resources. When connecting to an HTTPS server, Mailit validates the server certificate in order to establish a secure and trusted connection.

Mailit can use the following certificate sources:

- 1 Built-in certificate bundle
- 2 External certificate file `cacert.pem`
- 3 System certificate store, if supported by the current platform

Normally no additional action is required. However, if secure HTTPS requests stop working because the built-in certificate bundle is no longer sufficient, you can provide an external certificate bundle file.

A trusted CA bundle in PEM format can be downloaded from:

<https://curl.se/ca/cacert.pem>

Place the file named `cacert.pem` in the same directory as the Mailit plug-in.

For Apple macOS and FileMaker® Pro this means the Mailit plug-in folder. For MS Windows place the file next to the Mailit plug-in file. For FileMaker® Server place the file in the same plug-in directory where the Mailit plug-in is installed.

Typical signs that the built-in certificate bundle is no longer sufficient:

- 1 OAuth authentication over HTTPS stops working
- 2 REST API requests over HTTPS fail
- 3 functions that load remote content over HTTPS return connection or URL errors
- 4 the Mailit log reports SSL, TLS, certificate, or certificate validation errors

Please note:

- 1 the certificate file must be in PEM format
- 2 the file name should be `cacert.pem`
- 3 the certificate bundle should be kept up to date

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	M	Text	Contains the email address of the email account
To	M	Text	Contains the recipient's email address
Cc	M	Text	Contains the Cc email address
Subject	M	Text	Contains the email subject of the email

Field Name	Table	Type	Description
Body	M	Text	Contains the body of the message either as plain text, as HTML code or as rich text (formatted text)
Message Object	M	Container	Original message object, necessary to display a message using Web Viewer
Format	M	Text	Displays the format of the email message
Header	M	Text	Contains the header of the message
Charset	M	Text	Contains the character set used for that message
Priority	M	Text	Contains the priority of the message
Date	M	Date	Contains the date of the message
Time	M	Time	Contains the time of the message
Message ID	M	Text	ID of the current message
Current Message	S	Text	Current message being processed
Message ID	A	Text	Contains unique ID for the message
Attachment ID	A	Text	Contains unique ID for the attachment
Name (Attachment)	A	Text	Contains the name of the attachment file
Object	A	Container	Contains the actual attachment
Icon	A	Container	Contains icon of file type for the attachment
External Path	A	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 "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 11 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:Email_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" ); $connectionID )]
Else
  Show Custom Dialog [ Title: "Email Download";
    Message: "Connection to the POP server could not be established!
    Check the account data and your internet/intranet connection.¶¶
    Plug-in result: " & System::Result; Buttons: "OK" ]
End If
Set Field [ System::Result; Email_PerformScript ($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 [ $$download_in_process; Value:"" ]
Set Variable [ $layout; Value:Get ( LayoutName ) ]
#Pull thread information from script parameter
Set Variable [ $connectionID;
  Value:Substitute (MiddleValues (Get (ScriptParameter); 1; 1);"¶";"") ]
Set Variable [ $threadID;
  Value:Substitute (MiddleValues (Get (ScriptParameter); 2; 1);"¶";"") ]
Go to Layout [ "Account" (System) ]
Set Field [ System::Download History; Email_PopGetHistory( $connectionID ) ]
# Check result
Set Variable [ $result; Value:Email_GetLastResult ($threadID;
  "Email_PopRetrieveAllMessages") ]
If [ Left($result; 3) = "000" ]
  # Retrieve number of new messages
  Go to Layout [ "List" (Messages) ]
  Set Variable [ $total;
    Value:Email_PopGetMessageCount(""; $connectionID) ]
  Set Variable [ $$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;
        Email_PopGetMessageSource( $current; $connectionID ) ]
      Set Variable [ $result;
        Value:Email_ProcessMessageSource( Messages::Message Object ) ]
      Exit Loop If [ Left($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 [ $$progress_done;
    Value:If ($current < $total; $current + 1;
        $total) ]
# Update list content
Sort Records [ Specified Sort Order: Messages::Date;
    descending Messages::Time; descending ][ Restore; No dialog ]
Refresh Window
Freeze Window
End Loop
End If
Else
    Show Custom Dialog [ Title: "Error"; Message: "Error downloading messages:¶ "
        & $result; Buttons: "OK" ]
End If
Set Variable [ $result; Value:Email_DestroyThread( $threadID )]
Set Variable [ $$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 11. 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
Message ID	A	Text	ID of the current message
Name (Attachment)	A	Text	Contains the name of the attachment file
Object	A	Container	Contains the actual attachment
Icon	A	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; Email_SmtpResetMessage ("resetAttachments"; "resetBodies"; "
    resetHeader") ]
#Setup status dialog
Set Field [ System::Result; Email_SetupStatusDialog ("Sending Message"; "allowCancelOn";
    "Cancel"; "autoText"; ""; "reset") ]
Set Field [ System::Result; Email_SetupProgressMax (1) ]
Set Field [ System::Result; Email_ShowStatusDialog ("show") ]
#Set "From" name and address
Set Field [ System::Result; Email_SmtpAddMessageField ("From" ; "\" & System::Sender
    Name & "\" & "<" & System::Email Address & ">") ]
#Set "To" name and address(es)
Set Field [ System::Result; Email_SmtpAddMessageField ("To" ; Messages::To) ]
#Set "Cc" name and address(es)
Set Field [ System::Result; Email_SmtpAddMessageField ("Cc" ; Messages::Cc) ]
#Set "Bcc" name and address(es)
Set Field [ System::Result; Email_SmtpAddMessageField ("Bcc" ; Messages::Bcc) ]
#Set priority
Set Field [ System::Result; Email_SmtpSetMessagePriority (Messages::Priority) ]
#Set subject
Set Field [ System::Result; Email_SmtpAddMessageField ("Subject"; Messages::Subject) ]
#Set message body
If [ Messages::Format = "Text" ]
    Set Field [ System::Result; Email_SmtpAddBody ("text"; Messages::Message; "")]
Else If [ Messages::Format = "HTML Code" ]
    Set Field [ System::Result; Email_SmtpAddBody ("html"; Messages::Message; "")]
Else If [ Messages::Format = "Rich" ]
    Set Field [ System::Result; Email_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; Email_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; Email_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 11 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; Emal_SmtpAddMessageField ("From" ;"\\" &
    System::Sender Name & "\\" & "<" & System::Email Address & ">";
    $messageID) ]
#Set "To" name and address(es)
Set Field [ System::Result; Emal_SmtpAddMessageField ("To"; Messages::To;
    $messageID) ]
#Set "Cc" name and address(es)
Set Field [ System::Result; Emal_SmtpAddMessageField ("Cc"; Messages::Cc;
    $messageID) ]
#Set "Bcc" name and address(es)
Set Field [ System::Result; Emal_SmtpAddMessageField ("Bcc"; Messages::Bcc;
    $messageID) ]
#Set priority
Set Field [ System::Result; Emal_SmtpSetMessagePriority (Messages::Priority;
    $messageID) ]
#Set subject
Set Field [ System::Result; Emal_SmtpAddMessageField ("Subject"; Messages::Subject;
    $messageID) ]
#Set message body
If [ Messages::Format = "Text" ]
    Set Field [ System::Result; Emal_SmtpAddBody ("text"; Messages::Message; "";
        $messageID) ]
Else If [ Messages::Format = "HTML Code" ]
    Set Field [ System::Result; Emal_SmtpAddBody ("html"; Messages::Message; "";
        $messageID) ]
Else If [ Messages::Format = "Rich" ]
    Set Field [ System::Result; Emal_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; Emal_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; Emal_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; ""; $threadID) ]
Set Field [ System::Result; Emal_GetLastResult ($threadID;
    "Emal_Connect") ]
#Send message to SMTP server
Set Field [ System::Result; Emal_SmtpSendMessage (""; "CancelOnRcptErr";
    $connectionID; $messageID) ]
#Disconnection from SMTP
Set Field [ System::Result; Emal_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 11 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 11 functions and their parameters.

Accessing external functions

Since Mailit 11 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 11 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 11 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® 15-26 (64-bit only)) 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 11 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 11 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 11.

Double-click on the Mailit 11 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 11 are marked with asterisk (*).
Functions introduced in Mailit 11 are marked with double asterisk (**).

Emai_GetLastResult

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>{threadUID}</code>	Use this parameter to retrieve a function result that was queued in a thread.
<code>{functionName}</code>	In case if a <code>{threadUID}</code> parameter was specified, this parameter should be used to specify which function result code should be requested from the queue. As soon as the <code>Emai_GetLastResult</code> 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.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: {option} "product" , "version" , "platform" or "copyright" .

In order to get information about the license, use additional values for the "option" parameter:

"licenseStatus" - returns the value: registered/unregistered;

"licenseTo" - returns the license name;

"licenseCode" - returns the license code, all characters except the last four are masked "**";

"licenseIncludesUsers" - returns the number of users included in the license;

"licenseIncludesServer" - returns value: Yes/No;

"licenseAllowRedistribution" - returns value: Yes/No;

"licenseDescription" - returns a string with information about the license;

"licenseType" - returns the license type: Perpetual, Subscription, Trial, Unregistered, or Invalid;

"licenseValidUntil" - returns the subscription expiration date or, for a perpetual license, the last date covered by maintenance/updates;

"pluginBuildDate" - returns the build date of the installed plugin version in YYYY-MM-DD format;

Empty All values are returned separated by space " " , except for the data license.

Emai_Connect

Description:

Establishes a connection to an SMTP server to send messages or to an IMAP / POP3 server to receive messages.

The function also supports REST-based connections using the "gmailRest" and "graphRest" connection types, providing access to Gmail API and Microsoft Graph mail services via OAuth.

For threaded SMTP sessions a unique thread identifier should be specified using the "threadUID" parameter. In this case the function shall return a unique connection identifier ("connectionUID"), instead of the result code. In order to retrieve the function call result use the "Emai_GetLastResult" function.

REST connections may also return a "connectionUID" used for subsequent operations.

Compatibility:

Win, macOS, Linux. Desktop & Server

Parameters:

connectionType	"pop", "imap", "smtp", "gmailRest", or "graphRest"
server	Server address. For REST connections this parameter may be empty or contain the API endpoint URL.
{userName}	POP, IMAP, SMTP or REST account user name.
{password}	POP, IMAP or SMTP password. For OAuth authentication ("authType = oauth") this parameter must contain an access token.
{authType}	For POP and IMAP connections this can be: "auto", "login", "apop", "plain", "password", "ntlm" and "oauth". For SMTP connections this can be: "auto", "login", "plain", "ntlm", "cram-md5" and "oauth". For REST connections only "oauth" is supported. If no preferred value is indicated the plug-in tries all.
{useSSL}	"sslOn" or "sslOff" (default); establishes a secure connection to a POP, IMAP or SMTP server; if secure connection is requested and the server does not support it the connection will not be established. For REST connections HTTPS is always used and this parameter is ignored.
{timeout}	Default is "60" (value in seconds). For REST connections this value defines the HTTP request timeout.
{port}	Default for POP is "110", for IMAP is "143", and for SMTP it is "25". For REST connections this parameter is ignored.
{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 <code>Emai_ImapGetMessageCount</code> function, as a parameter. For REST connections this parameter is reserved and may be ignored.
{threadUID}	This parameter may optionally contain a thread ID, created by the <code>Emai_CreateThread</code> plug-in function, used for multi-threaded sending. For REST sending this parameter may also be used for background execution.

Emai_Disconnect

Description: Closes the connection to an IMAP, POP, SMTP or REST mail service.

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>connectionType</code>	"imap" , "pop" , "smtp" , "gmailRest" or "graphRest"
<code>{deleteMessages}</code>	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. For REST connections this parameter is ignored.
<code>{connectionUID}</code>	Optional connection identifier returned by the <code>Emai_Connect</code> function. For SMTP this may refer to a threaded connection created via <code>Emai_CreateThread</code> . For REST connections this identifies the active REST session to be closed.

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_GetLastResult` function.

Compatibility: Win, macOS. Desktop only. 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.

Compatibility: Win, macOS. Desktop only. On Server side use single-threaded functions.

Parameters:

<code>threadUID</code>	Unique thread ID, generated by <code>Emai_CreateThread</code> function.
<code>{queue}</code>	This function call may get queued. Use <code>"queue"</code> 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.

Compatibility: Win, macOS. Desktop only. On Server side use single-threaded functions.

Parameters:

<code>threadUID</code>	Unique thread ID, generated by <code>Emai_CreateThread</code> 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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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

Emai_GetFileIcon

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

Compatibility: Win, macOS. Desktop & Server

Parameters:

filePath	Path pf the file including file name.
{iconSize}	Icon size in pixels, " 1 6 " (default) or " 3 2 "

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.

Compatibility: Win, macOS. Desktop only. 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

Email_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.

Compatibility: Win, macOS. Desktop only. On Server side will be ignored.

Parameters:

<code>dialogMode</code>	<code>"save" or "open"</code>
<code>{dialogTitle}</code>	Defines a custom title of the dialog; default title is empty
<code>{defaultFolderPath}</code>	Default folder which will be shown in dialog
<code>{defaultFileName}</code>	Default file will be selected (<code>dialogMode="open"</code>) or default new file name will be suggested (<code>dialogMode="save"</code>)
<code>{filter}</code>	Sets file filters for dialog; syntax for filter as follows: <code>[Adobe Photoshop; CompuServe GIF] *.psd; *.gif</code> <code>[Microsoft Word; Microsoft Excel] *.doc; *.xls</code> <code>[Movie Files] *.mov; *.avi</code> 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: <code>*.fp5; *.fp7; *.fmp12</code> <code>*.fp3</code> Default will be no filter (any file can be selected)

Email_ImportFile

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

Compatibility: Win, macOS, Linux. Desktop & Server

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® 15-26 (64-bit only) . The plug-in provides separate functions to retrieve file names and icons of external files and attachments into separate fields.

Email_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 `Email_ImportFile` (in combination with `Email_GetFileName` etc). To import attachments from a retrieved message into FileMaker® use `Email_PopGetAttachment` (in combination with `Email_PopGetAttachmentName` etc). Use `Email_GetUserFolder` to retrieve system folder paths such as Desktop or Temp. Alternatively, use `Email_ShowFileDialog` to let the user specify the target path.

Compatibility: Win, macOS, Linux. Desktop & Server

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 `protectFile="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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

path	This parameter specifies the source path which is to be converted. The source path must be in <code>filewin</code> , <code>filemac</code> , <code>moviewin</code> , <code>moviemac</code> , <code>imagewin</code> , <code>imagemac</code> 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 <code>filewin</code> format (Windows) or <code>filemac</code> format (Mac OS). The value "image" returns the path in the FileMaker® <code>imagewin</code> format (Windows) or the <code>imagemac</code> format (Mac OS) which is required for the <i>Insert Picture</i> script step. The value "movie" returns the path in the FileMaker® <code>moviewin</code> format (Windows) or in the <code>moviemac</code> format (Mac OS), which is required for the <i>Insert QuickTime</i> 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.

Compatibility: Win, macOS. Desktop only.

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.

Compatibility: Win, macOS. Desktop only.

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS. Desktop only. On Server side will be ignored.

Parameters:	<code>{dialogTitle}</code>	Title of the status dialog; default is empty
	<code>{allowCancel}</code>	"allowCancelOn" or (default) "AllowCancelOff"
	<code>{cancelButtonText}</code>	"cancelButtonText" or (default) "AllowCancelOff"
	<code>{allowCancel}</code>	(default) "Cancel" or custom value
	<code>{textMode}</code>	"autoText" (default), "combinedText" (custom text is shown on top of auto text) or "customText" (only custom text)
	<code>{customText}</code>	Custom text to be shown
	<code>{resetProgress}</code>	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 settings 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.
	<code>{threadUID}</code>	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.
	<code>{fileName}</code>	Name of the file containing the script to be triggered, in case if the user will cancel the operation
	<code>{scriptName}</code>	Name of the script to be triggered, in case if the user will cancel the operation
	<code>{scriptParameter}</code>	Script parameter to be used for the script triggered, in case if the user will cancel the operation

Email_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 `Email_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.

Compatibility: Win, macOS. Desktop only. On Server side will be ignored.

Parameters:

<code>{totalMessages}</code>	Number of messages the progress bar will be based on.
<code>{threadUID}</code>	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.

Email_SetCurrentProgress

Description: Sets the current progress value in the status dialog. The maximum value is set using the `Email_SetupProgressMax` function.

Compatibility: Win, macOS. Desktop only. On Server side will be ignored.

Parameters:

<code>{currentMessage}</code>	An integer in the range from 0 to the value set in the <code>Email_SetupProgressMax</code> function.
<code>{threadUID}</code>	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`.

Compatibility: Win, macOS. Desktop only. 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.

Compatibility: Win, macOS. Desktop only. 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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Email_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 `Email_GetLastSessionTranscript` function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: lineNumber Maximum number of text lines to keep

Email_GetLastSessionTranscript

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: connectionType "pop" , "imap" , "smtp" , "gmailRest" or "graphRest".

{connectionUID} Optional connection identifier returned by the `Emai_Connect` function.

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`.

Compatibility: Win, macOS. Desktop only.

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.

Email_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 `Email_PopGetMessageHTML`.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>html</code>	HTML code or HTML object to be exported.
	<code>{destinationPath}</code>	Destination folder, where HTML object should be saved to. By default the object will be saved at the system default Temporary folder with file name "index.html" and folder with name "attachments" If the path ends with the name of the file "your_file_name.html", a file with the specified name and a folder for storing the attachments with the same name will be created.
	<code>{charset}</code>	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.
	<code>{convertImages}</code>	By default <code>false</code> , this means that the images will be saved separately in the <code>attachments</code> folder. If the value of the parameter is <code>true</code> , the images will be coded in the base64 string and embedded in the html file.

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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 11). If a session is registered using this function the registration takes priority for the current session over the registration information from the preferences.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Server only

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_UnregisterPlugin

Description: Removes the stored MailIt registration name and license key. The plug-in returns to trial mode. If the trial period has expired, all MailIt functions remain blocked until a valid license is registered.

Compatibility: Win, macOS, Linux. FileMaker Pro and FileMaker Server

Parameters: None This function does not require any parameters.

Emai_ImportHTMLFile

Description: Returns the prepared body of the message in HTML, external images are added to the email as attachments, image addresses are replaced with cid. We recommend that you use inline styles to design an HTML message.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: filePath The path to the HTML file on the local disk or on a remote web server (supported protocols: http, https).

 {downloadImages} The default is `false`, if `true` - the plugin loads images from external links in the temporary folder, after that it will be possible to add images as an inline attachment with the cid, using `Emai_ImportHTMLAttachmentCount` and `Emai_ImportHTMLAttachmentPath( pathIndex )`.

 {templateVariables} Use this parameter for pass the key = value parameters through one line is separated by a `","`.

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

Emai_ImportHTMLAttachmentCount

Description: Returns the number of file path for file attachment, after cell Emai_ImportHTMLFile

Compatibility: Win, macOS, Linux. Desktop & Server

Emai_ImportHTMLAttachmentPath

Description: Returns the path to the image in the temporary folder loaded with Emai_ImportHTMLFile

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: pathIndex Index of file path in this field [starting with 0].

Emai_LocalWebServerStart

The server is used for:

- Capturing a redirect of an authorization request when passing through the OAuth authorization stages.
- Saving and loading the message body in HTML format, as well as images when used in FileMaker Web Viewer, the editors "Summernote" and "Automizy Editor" to design message templates both based on JavaScript libraries.

For OAuth authentication, a file is used that is located in "../Mailit Assets / oauth / index.html".

The "../Mailit Assets/editor/automizy" folder contains a library and related files for the "Automizy Editor".

The "../Mailit Assets/editor/summernote" folder contains a library and related files for "Summernote".

The "../Mailit Assets/editor/upload" folder is used for temporary storage of images when working with "Summernote" and "Automizy Editor".

To test the examples that are in the Mailit download package, the "Mailit Assets" folder must be copied together with the Mailit plug-in to the FileMaker Extensions folder. The Mailit plugin automatically looks for the "Mailit Assets" folder within the FileMaker Extensions folder. If necessary, you can change the name of the "Mailit Assets" folder to any other one and place the directory in any convenient place on your local drive. But when changing the name or location of the folder, you need to pass in the third parameter of the function, "path", the new location and name of the folder in the format of the FileMaker paths using an absolute path, for example:

filewin:/driveletter:/directoryName/

filemac:/volumeName/directoryName/

After starting the server, the following URLs are available:

http://localhost:1234/ html files and directories in the directory "../Mailit Assets/" are available at this address. Only static files are available, PHP and other server-side scripts are not executed. Only the GET request type is supported.

http://localhost:1234/editor/upload Only the POST request type is supported. Accepts image files from the editor and uploads images to a directory "../FileMaker Pro 18 Advanced/Extensions/Mailit Assets/editor/upload/".

http://localhost:1234/editor/load Only the POST request type is supported. In the request, you need to pass the fieldName parameter with the name of the field in the database from which you want to return the result in response, for example fieldName = Messages :: Body, it will return the current active content of the "Body" field from the "Messages" table as a text string, provided that the user currently on the layout in which the table and field are available. To get the value "Messages :: Body", use the functions from the FileMaker API "Evaluate".

http://localhost:1234/editor/save Only the POST request type is supported. In the request, you need to pass the parameters: scriptFileName name of the database file; scriptName the name of the script in the database; scriptParameter parameters that will be passed to the executable script.

The scriptParameter parameter must be processed by the javascript encodeURIComponent() function, inside the plug-in the value of the scriptParameter parameter will be decoded back.

In order to quickly understand the operation of the function and the local web server, see the example "Simple Sending Automizy Editor.fmp12" and "Simple Sending WYSIWYG Editor.fmp12".

Description: Starts a local web server that supports HTTP. The server only supports static files.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

host	IP address or host name. Default is "localhost"
port	The parameter defines the port through which the local server will be available, by default the port "1234" is used.
path	The path to the mount point. By default, this is the path to additional plug-in files in the "Mailit Assets" folder, which must be copied in the place with the plug-in to the FileMaker® extension directory. The path is specified in FileMaker® format, that is, with the prefix "filewin://" or "filemac://" depending on the platform you are using.

Email_LocalWebServerStop

Description: Stops a local web server.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_LocalWebServerStatus

Description: Returns the status and settings of the local web server.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: property "isServerRunning", "getUploadPath", "getPath", "getHost", "getPort".

Email_URLRequest

Description: Performs an HTTP / HTTPS request to the specified URL resource using the "POST" or "GET" method with the ability to pass a query string using the optional parameter:{fields}. As a result of execution returns a text string.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

- type HTTP request method "POST" or "GET"
- url a URL (Uniform Resource Locator) is used to uniquely identify a resource over the web.
- {fields} query string for URL, example: "field1=value1&field2=value2&field3=value3..."

Emai_JSONGetElement

Description: Queries JSON data for an element specified by an object name or a path.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

json	any text expression or field that contains a JSON object.
keyOrIndexOrPath	any text expression or field that specifies a JSON object name (key) or a path.

Note The function is used to get the token of the key and additional information from the response of the authorization server that comes in JSON format, at the time of going through the OAuth authorization stages. The function was created to overlap compatibility with Filemaker 15, and partially copies the functionality of the "JSONGetElement" function built into FileMaker 16. The function is able to work with JSON objects and strings without supporting arrays.

Emai_ExecuteSQL

Description: Performs SQL commands on FileMaker tables.

Compatibility: Win, macOS. Desktop only

Parameters:

sqlStatement	the command to perform.
{columnSeparator}	the column separator for the output data - single characters only.
{rowSeparator}	the row separator for the output data - single characters only.
{database}	allows you to specify an open database other than the current one.
{textOrObject}	the parameter determines the type of the returned result. Available values are "text" and "object", if the parameter is not specified, "text" is used by default.
{parameters}	pass parameters here. Use one parameter to this function for each parameter you want to pass to the SQL statement.

Emai_GetLastDDLResult

Description: The type of SQL commands DDL(Data Definition Language) that change the structure of the table, the CREATE, ALTER, DROP commands are executed by the plugin at the time of FileMaker® idle time, so the execution result is not immediately available in the script step, to get the result, use the Emai_GetLastDDLResult plugin function.

Compatibility: Win, macOS. Desktop only

Emai_GetMaessageFromAccount

Description: This function downloads messages via POP/IMAP protocol and sends messages via SMTP protocol in the background. The plugin interacts with FileMaker® using SQL queries. The function requires a specific database structure to work. The structure of the database has been implemented in the example file “Personal Mail (background).fmp12” for your reference.

Compatibility: Win, macOS. Desktop only

Email_awsGetAccountInfo

Description: Obtain information about the email sending status and capabilities of your Amazon SES account in the current AWS Region. When you call the Email_awsGetAccountInfo function, you will get the following information:

DedicatedIpAutoWarmupEnabled - Indicates whether or not the automatic warm-up feature is enabled for dedicated IP addresses that are associated with your account.

EnforcementStatus - The reputation status of your Amazon SES account. The status can be one of the following:

HEALTHY – There are no reputation-related issues that currently impact your account.

PROBATION – Amazon has identified potential issues with your Amazon SES account. Amazon has placed your account under review while you work on correcting these issues.

SHUTDOWN – Your account's ability to send email is currently paused by Amazon because of an issue with the email sent from your account. When you correct the issue, you can contact Amazon and request that your account's ability to send email is resumed.

ProductionAccessEnabled - Indicates whether or not your account has production access in the current AWS Region.

If the value is false, then your account is sandbox mode. When your account is in sandbox mode, you can only send email to verified identities. Additionally, the maximum number of emails you can send in a 24-hour period (your sending quota) is 200, and the maximum number of emails you can send per second (your maximum sending rate) is 1.

SendQuota - An object that contains information about the per-day and per-second sending limits for your Amazon SES account in the current AWS Region.

The function returns a JSON object, example:

```
{
  "DedicatedIpAutoWarmupEnabled": "true",
  "EnforcementStatus": "HEALTHY",
  "ProductionAccessEnabled": "false",
  "SendingEnabled": "true",
  "SendQuota":
  [
    {
      "MaxSendRate": "1",
      "Max24HourSend": "200",
      "SentLast24Hours": "0"
    }
  ]
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Email_awsSetCredentials

Description: Call this function before calling any other AWS related functions. Due to a feature of the Amazon API, the function does not validate the data if the accessKey or secretKey was not correct, you will know about this when executing any other function that is related to AWS SES.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: accessKey Access Key for IAM user;
 secretKey Secret Key for IAM user;
 region region code (us-east-1, eu-west-1, eu-central-1, ...).

The following table lists the name and code of each Region.

Name	Code
US East (Ohio)	us-east-2
US East (N. Virginia)	us-east-1
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Africa (Cape Town)	af-south-1
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Jakarta)	ap-southeast-3
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Milan)	eu-south-1
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
Middle East (Bahrain)	me-south-1
Middle East (UAE)	me-central-1
South America (São Paulo)	sa-east-1
AWS GovCloud (US-East)	us-gov-east-1
AWS GovCloud (US-West)	us-gov-west-1

Emai_awsSendMessage

Description: Composing an e-mail message and immediately sends.

Compatibility: Win, macOS, Linux. Desktop & Server

Returns: message ID;

Parameters:

htmlMessage	The content of the message, in HTML format. Use this for email clients that can process HTML. You can include clickable links, formatted text, and much more in an HTML message;
plainTextMessage	The content of the message, in text format. Use this for text-based email clients, or clients on high-latency networks (such as mobile devices);
subject	The subject of the message. A short summary of the content, which appears in the recipient's inbox;
fromAddress	The email address that is sending the email. This email address must be either individually verified with Amazon SES, or from a domain that has been verified with Amazon SES. For information about verifying identities;
toAddress	The recipients to place on the To: line of the message;
ccAddress	The recipients to place on the CC: line of the message;
bccAddress	The recipients to place on the BCC: line of the message;
replyToAddress	The reply-to email address for the message;
{tags}	A list of tags, in the form of namevalue pairs, to apply to an email that you send using Emai_awsSendMessage. Tags correspond to characteristics of the email that you define, so that you can publish email sending events, example: "name1=value1;name2=value2;name3=value3;"
{configurationSetName}	The name of the configuration set;
{contactListName}	The name of the contact list, will be used when a contact chooses to unsubscribe;
{topicName}	The name of the topic, will be used when a contact chooses to unsubscribe.

Emai_awsSendRawMessage

Description: Composing an e-mail message and immediately sends.

Compatibility: Win, macOS, Linux. Desktop & Server

Returns: message ID;

Parameters:

messageSource	The message object from the Emai_SmtpComposeMessage function;
{tags}	A list of tags, in the form of name/value pairs, to apply to an email that you send using Emai_awsSendRawMessage. Tags correspond to characteristics of the email that you define, so that you can publish email sending events, example: "name1=value1;name2=value2;name3=value3;"
{configurationSetName}	The name of the configuration set;
{contactListName}	The name of the contact list, will be used when a contact chooses to unsubscribe;
{topicName}	The name of the topic, will be used when a contact chooses to unsubscribe.

This function is more flexible than the Emai_awsSendMessage function. When you use the Emai_awsSendRawMessage function, you can specify the headers of the message as well as its content. This flexibility is useful, for example, when you need to send a multipart MIME email (such a message that contains both a text and an HTML version). You can also use this function to send messages that include attachments.

The Emai_awsSendRawMessage function has the following requirements:

You can only send email from verified email addresses or domains. If you try to send email from an address that isn't verified, the operation results in an "Email address not verified" error.

If your account is still in the Amazon SES sandbox, you can only send email to other verified addresses in your account, or to addresses that are associated with the Amazon SES mailbox simulator.

The maximum message size, including attachments, is 10 MB.

Each message has to include at least one recipient address. A recipient address includes any address on the To:, CC:, or BCC: lines.

If you send a single message to more than one recipient address, and one of the recipient addresses isn't in a valid format (that is, it's not in the format User-Name@[SubDomain.]Domain.TopLevelDomain), Amazon SES rejects the entire message, even if the other addresses are valid.

Each message can include up to 50 recipient addresses across the To:, CC:, or BCC: lines. If you need to send a single message to more than 50 recipients, you have to split the list of recipient addresses into groups of less than 50 recipients, and send separate messages to each group.

Although you can customize the message headers when using the `Email_awsSendMessage` function, Amazon SES automatically applies its own Message-ID and Date headers; if you passed these headers when creating the message, they are overwritten by the values that Amazon SES provides.

For every message that you send, the total number of recipients (including each recipient in the To:, CC: and BCC: fields) is counted against the maximum number of emails you can send in a 24-hour period (your sending quota). For more information about sending quotas in Amazon SES, see *Managing Your Amazon SES Sending Limits* in the Amazon SES Developer Guide.

Email_awsUploadResource

Description: This function allows you to upload resources such as graphics, PDF files, or other file types that can then be hyperlinked to your e-mail message. Be aware that anybody can download any resource if they know the name of the file, so if you are using this for confidential information, include random characters as part of the uploadName, so that it cannot be guessed. To check function execution errors, use `Email_GetLastResult`.

Compatibility: Win, macOS, Linux. Desktop & Server

Returns: URL to resource;

Parameters: resourcePathOrContainer uploaded file, can be a file path in FileMaker® format or a link to a container field;

 uploadName file name;

 bucketName the name of the bucket to which the file will be loaded.

Email_awsDownloadResource

Description: The function downloads an object to FileMaker® that is located in the Amazon S3 bucket. To check function execution errors, use `Email_GetLastResult`.

Compatibility: Win, macOS, Linux. Desktop & Server

Returns: URL to resource;

Parameters: name name of the object in the bucket (object key).

 bucketName the name of the bucket to which the file will be loaded.

Email_awsDeleteResource

Description: This function removes an item that was previously uploaded to Amazon S3 using the Email_awsUploadResource function. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: uploadName file name;
 bucketName the name of the bucket to which the file will be loaded.

Email_awsGetBucketsList

Description: The function returns lists all of your S3 buckets as a list. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_awsCreateBucket

Description: Creates a new S3 bucket. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: name the name of the bucket;

Email_awsRemoveBucket

Description: Deletes the S3 bucket. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: name the name of the bucket;

Email_awsGetBucketListObjects

Description: The function returns a list of objects that are in the Amazon S3 bucket. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: name the name of the bucket;

Email_awsGetBucketPolicy

Description: The function returns the policy for the specified Amazon S3 bucket. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: name the name of the bucket;

Email_awsPutBucketPolicy

Description: The function adds a policy for the specified Amazon S3 bucket. To check function execution errors, use Email_GetLastResult.

Example public policy:

```
{
  "Version": "2012-10-17",
  "Id": "Policy63824243072398462",
  "Statement": [
    {
      "Sid": "Stmt63824243072398477",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::EXAMPLE-BUCKET/*"
    }
  ]
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

name	the name of the bucket;
policyJsonString	the bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts.;

Email_awsDeleteBucketPolicy

Description: The function deletes the policy for the specified Amazon S3 bucket. To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: name the name of the bucket;

Email_awsGetBucketPublicAccessBlock

Description: Retrieves the PublicAccessBlock configuration for an Amazon S3 bucket. To check function execution errors, use Email_GetLastResult.

The function returns a JSON object, example:

```
{
  "blockPublicAcls": "true",
  "blockPublicPolicy": "true",
  "ignorePublicAcls": "true",
  "restrictPublicBuckets": "true",
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: name the name of the bucket;

Email_awsPutBucketPublicAccessBlock

Description: The PublicAccessBlock configuration that you want to apply to this Amazon S3 bucket. You can enable the configuration options in any combination. Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. See the link for more information:

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-control-block-public-access.html>

To check function execution errors, use Email_GetLastResult.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

name	the name of the bucket;
blockPublicAcls	specifies whether Amazon S3 should block public access control lists (ACLs) for this bucket and objects in this bucket. Type: Boolean. Default: True;
blockPublicPolicy	specifies whether Amazon S3 should block public bucket policies for this bucket. Type: Boolean. Default: True;
ignorePublicAcls	specifies whether Amazon S3 should ignore public ACLs for this bucket and objects in this bucket. Type: Boolean. Default: True;
restrictPublicBuckets	specifies whether Amazon S3 should restrict public bucket policies for this bucket. Setting this element to TRUE restricts access to this bucket to only AWS service principals and authorized users within this account if the bucket has a public policy. Type: Boolean. Default: True;

Email_awsGetSendStatistics

Description: Provides sending statistics for the current AWS Region. The result is a list of data points, representing the last two weeks of sending activity. Each data point in the list contains statistics for a 15-minute period of time. To check function execution errors, use Email_GetLastResult.

You can execute this function no more than once per second.

The function returns a JSON object, example:

```
{
  "SendDataPoints":
  [
    {
      "Timestamp": "1663950300000",
      "DeliveryAttempts": "1",
      "Bounces": "0",
      "Complaints": "0",
      "Rejects": "0"
    },
    {
      "Timestamp": "1663954800000",
      "DeliveryAttempts": "1",
      "Bounces": "0",
      "Complaints": "0",
      "Rejects": "0"
    }
  ]
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Emai_awsGetSendQuota

Description: The function makes a request to the Amazon SES API provides a GetSendQuota action that returns your send quotas. When you call the Emai_awsGetSendQuota function, you will get the following information:

Max24HourSend - The maximum number of emails that you can send in the current AWS Region over a 24-hour period. A value of -1 signifies an unlimited quota. (This value is also referred to as your sending quota.)

MaxSendRate - The maximum number of emails that you can send per second in the current AWS Region. This value is also called your maximum sending rate or your maximum TPS (transactions per second) rate.

SentLast24Hours - The number of emails sent from your Amazon SES account in the current AWS Region over the past 24 hours.

To check function execution errors, use Emai_GetLastResult.

The function returns a JSON object, example:

```
{
  "MaxSendRate": "1",
  "Max24HourSend": "200",
  "SentLast24Hours": "0"
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Email_awsCreateContactList

Description: Creates a contact list.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	contactListName	The name of the contact list;
	contactListDescription	A description of what the contact list is about;
	topicsJson	An interest group, theme, or label within a list. A contact list can have multiple topics. The parameter accepts a JSON object, example: <pre>{ "Topics":[{ "DefaultSubscriptionStatus":"string", "Description":"string", "DisplayName":"string", "TopicName":"string" }] }</pre>
	DefaultSubscriptionStatus	The default subscription status to be applied to a contact if the contact has not noted their preference for subscribing to a topic. Valid Values: OPT_IN OPT_OUT.
	Description	A description of what the topic is about, which the contact will see.
	DisplayName	The name of the topic the contact will see.
	TopicName	The name of the topic.

Emai_awsGetContactList

Description: Returns contact list metadata. It does not return any information about the contacts present in the list. The function returns a JSON object, example:

```
{
  "ContactListName": "string",
  "CreatedTimestamp": number,
  "Description": "string",
  "LastUpdatedTimestamp": number,
  "Topics": [
    {
      "DefaultSubscriptionStatus": "string",
      "Description": "string",
      "DisplayName": "string",
      "TopicName": "string"
    }
  ]
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: contactListName The name of the contact list;

Emai_awsDeleteContactList

Description: Deletes a contact list and all of the contacts on that list.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: contactListName The name of the contact list;

Emai_awsListContactLists

Description: Returns a list of all available contact lists.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_awsUpdateContactList

Description: Creates a contact list.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	contactListName	The name of the contact list;
	contactListDescription	A description of what the contact list is about;
	topicsJson	An interest group, theme, or label within a list. A contact list can have multiple topics. The parameter accepts a JSON object, example: <pre>{ "Topics":[{ "DefaultSubscriptionStatus":"string", "Description":"string", "DisplayName":"string", "TopicName":"string" }] }</pre>
	DefaultSubscriptionStatus	The default subscription status to be applied to a contact if the contact has not noted their preference for subscribing to a topic. Valid Values: OPT_IN OPT_OUT.
	Description	A description of what the topic is about, which the contact will see.
	DisplayName	The name of the topic the contact will see.
	TopicName	The name of the topic.

Email_awsGetContact

Description: Returns a contact from a contact list. The function returns a JSON object, example:

```
{
  "AttributesData": "string",
  "ContactListName": "string",
  "CreatedTimestamp": number,
  "EmailAddress": "string",
  "LastUpdatedTimestamp": number,
  "TopicDefaultPreferences": [
    {
      "SubscriptionStatus": "string",
      "TopicName": "string"
    }
  ],
  "TopicPreferences": [
    {
      "SubscriptionStatus": "string",
      "TopicName": "string"
    }
  ],
  "UnsubscribeAll": boolean
}
```

AttributesData - The attribute data attached to a contact.

ContactListName - The name of the contact list to which the contact belongs.

CreatedTimestamp - A timestamp noting when the contact was created.

EmailAddress - The contact's email address.

LastUpdatedTimestamp - A timestamp noting the last time the contact's information was updated.

TopicDefaultPreferences - The default topic preferences applied to the contact.

TopicPreferences - The contact's preference for being opted-in to or opted-out of a topic.

UnsubscribeAll - A boolean value status noting if the contact is unsubscribed from all contact list topics.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

contactListName	The name of the contact list to which the contact belongs;
emailAddress	The contact's email address.

Email_awsGetListContacts

Description: Lists the contacts present in a specific contact list. The function returns a JSON object, example:

```
{
  "Contacts": [
    {
      "EmailAddress": "string",
      "LastUpdatedTimestamp": number,
      "TopicDefaultPreferences": [
        {
          "SubscriptionStatus": "string",
          "TopicName": "string"
        }
      ],
      "TopicPreferences": [
        {
          "SubscriptionStatus": "string",
          "TopicName": "string"
        }
      ],
      "UnsubscribeAll": boolean
    }
  ]
}
```

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	contactListName	The name of the contact list;
	topicName	The name of the topic;
	subscriptionStatus	The contact's subscription status to a topic which is either OPT_IN or OPT_OUT.

Email_awsCreateContact

Description: Creates a contact, which is an end-user who is receiving the email, and adds them to a contact list.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	contactListName	The name of the contact list to which the contact should be added;
	emailAddress	The contact's email address;
	unsubscribeAll	A boolean value status noting if the contact is unsubscribed from all contact list topics;
	topicName	The name of the topic;
	subscriptionStatus	The contact's subscription status to a topic which is either OPT_IN or OPT_OUT;
	attributesData	The attribute data attached to a contact. The parameter accepts JSON as a string, example: "{firstname:"&Recipients::First Name&"", lastname:"&Recipients::Last Name&"}"

Email_awsUpdateContact

Description: Updates a contact's preferences for a list.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	contactListName	The name of the contact list;
	emailAddress	The contact's email address;
	{unsubscribeAll}	A boolean value status noting if the contact is unsubscribed from all contact list topics;
	{topicName}	The name of the topic;
	{subscriptionStatus}	The contact's subscription status to a topic which is either OPT_IN or OPT_OUT;
	{attributesData}	The attribute data attached to a contact. The parameter accepts JSON as a string, example: "{firstname:"&Recipients::First Name&"", lastname:"&Recipients::Last Name&"}"

Emai_awsDeleteContact

Description: Removes a contact from a contact list.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

contactListName	The name of the contact list from which the contact should be removed.
emailAddress	The contact's email address;

Emai_awsAddContactToSuppressionList

Description: Adds an email address to the suppression list for your account.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

emailAddress	The email address that should be added to the suppression list for your account;
reason	The factors that should cause the email address to be added to the suppression list for your account. Valid values: "BOUNCE", "COMPLAINT".

Emai_awsRemoveContactFromSuppressionList

Description: Removes an email address from the suppression list for your account.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

emailAddress	The suppressed email destination to remove from the account suppression list.
--------------	---

Emai_GetCurrentUTCOffset

Description: Returns the difference, between a date as evaluated in the UTC time zone, and the same date as evaluated in the local time zone. For example, New York is UTC-05:00, executing the Emai_GetCurrentUTCOffset function without specifying a parameter will return the result "-0500", Emai_GetCurrentUTCOffset("inMinutes") will return "-300", Emai_GetCurrentUTCOffset("inSeconds") will return the result "-18000".

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

{format}	If the parameter is not specified, the result will be in ±hhmm (ISO 8601) format, if "inMinutes" is specified, the result will be in minutes, if "inSeconds" is specified, the result will be in seconds.
----------	---

Email_ConvertDateRFC2822toFMDateTime

Description: The function converts a date and time from a text string in RFC2822 format (Fri, Mar 30 2023 20:33:05 +0100) to a FileMaker Timestamp. Used to get the Timestamp from the Date message header.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: dateRFC2822 A text string that contains a date in RFC2822 format.

 {type} "UTC" or "Local". The default is "UTC".

Emai_SmimeMessageEncrypt

Description: To encrypt a message, sign a message or sign and encrypt, use the `Emai_SmimeMessageEncrypt` function. The result of the function execution is the source of the message, which can be saved in a container field or immediately passed as a parameter to the message sending function `Emai_SmtpSendMessage`.

Use the `Emai_GetLastResult` function to check whether the function completed successfully. Error codes are described in this document, in Chapter 9, Result Codes. Error codes in the range -100 ... -133.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>signedAndEncrypted</code>	Possible action: "encrypted", "signed" or "signedAndEncrypted". The parameter is specified as a text string, case sensitive.
	<code>recipientCertificate</code>	The recipient's public certificate in PEM format, as an object.
	<code>senderCertificate</code>	Sender's public certificate in PEM format, as an object.
	<code>senderPrivateKey</code>	Sender's private key in PEM format, as an object.
	<code>{senderPrivateKeyPswd}</code>	Password from the sender's private key, passed in the "senderPrivateKey" parameter.
	<code>{cipher}</code>	is the symmetric cipher to use. The default is "aes_128_cbc". Possible values: "des_ede3_cbc", "aes_128_cbc", "aes_256_cbc", "aes_256_cbc_hmac_sha256", "rc2_40_cbc", "rc2_cbc". The parameter is specified as a text string, case sensitive.
	<code>{signatureType}</code>	The default is "detached". Possible values: "detached" or "opaque". The parameter is specified as a text string, case sensitive. The parameter affects the digital signature, so it is only relevant if the value "signed" or "signedAndEncrypted" was specified in the "signedAndEncrypted" parameter.
	<code>{messageUID}</code>	Use this parameter to affect a specific message object.

Emai_SmimeMessageDecrypt

Description: You can decrypt the S/MIME message using the `Emai_SmimeMessageDecrypt` function. The function does not check the digital signature. In order to check the digital signature, you need to pass the decrypted message to the `Emai_ProcessMessageSource` function, then execute the `Emai_ImapGetEncryptionType` function to make sure that the message has a digital signature. If the message has a digital signature, the signature can be verified using the `Emai_SmimeMessageVerifySign` function. The result of the function execution is the decrypted source of the message, which can be saved in a container field or immediately passed as a parameter to the `Emai_ProcessMessageSource` function.

Use the `Emai_GetLastResult` function to check whether the function completed successfully. Error codes are described in this document, in Chapter 9, Result Codes. Error codes in the range -100 ... -133.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>messageObject</code>	message object received from the <code>Emai_ImapGetMessageSource</code> function.
	<code>recipientCertificate</code>	Recipient's public certificate in PEM format, as an object.
	<code>recipientPrivateKey</code>	Recipient's private key in PEM format, as an object.
	<code>{recipientPrivateKeyPswd}</code>	Password from the recipient's private key passed in the <code>"recipientPrivateKey"</code> parameter.

Emai_SmimeMessageVerifySign

Description: The function `Emai_SmimeMessageVerifySign` allows you to check the digital signature of a message. The function supports detached and opaque digital signatures.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>messageObject</code>	Message object obtained from the <code>Emai_ImapGetMessageSource</code> function or the <code>Emai_SmimeMessageDecrypt</code> function.
	<code>senderCertificate</code>	The sender's public certificate in PEM format, as an object.
	<code>{CACertificate}</code>	The authentication root CA certificate for the certificate passed in the <code>"senderCertificate"</code> parameter. Certificate in PEM format as an object. If a certificate is passed to the function parameter, then the certificate will be used to verify the chain. If the certificate is not specified, then the signer's certificates <code>"senderCertificate"</code> are not verified by the chain.

Email_SmimeTypeConvertToDetached

Description: The `Email_SmimeTypeConvertToDetached` function converts an opaque signature to detached. The result of the function execution is the converted message source, which can be saved in a container field or immediately passed as a parameter to the `Email_ProcessMessageSource` or `Email_SmimeTypeVerifySign` function.

Use the `Email_GetLastResult` function to check whether the function completed successfully.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>messageObject</code>	Message object obtained from the <code>Email_ImapGetMessageSource</code> function or the <code>Email_SmimeTypeDecrypt</code> function.
	<code>senderCertificate</code>	The sender's public certificate in PEM format, as an object.
	<code>{CACertificate}</code>	Message object obtained from the <code>Email_ImapGetMessageSource</code> function or the <code>Email_SmimeTypeDecrypt</code> function.

Email_PGPMessageEncrypt

Description: The function creates an encrypted message source using PGP technology. The result of the function execution is the source of the message, which can be saved in a container field or immediately passed as a parameter to the message sending function `Email_SmtpSendMessage`. Use the `Email_GetLastResult` function to check whether the function completed successfully.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>signedAndEncrypted</code>	Possible action: "encrypted", "signed" or "signedAndEncrypted". The parameter is specified as a text string, case sensitive. The default is "signedAndEncrypted".
	<code>recipientPublicKey</code>	The recipient's public key, as an object.
	<code>senderPrivateKey</code>	Sender's private key, as an object.
	<code>{senderPrivateKeyPswd}</code>	Password from the sender's private key, passed in the "senderPrivateKey" parameter.
	<code>{senderPublicKey}</code>	The sender's public key as an object. If the parameter is defined, the sender's public key will be added to the message as an attachment.
	<code>{hideSubject}</code>	The parameter is defined as a native FileMaker Boolean data type, "True" or "False". The default value is "False". If the parameter is set to "True," then the Subject of the message will be replaced with "...". You can get the value of the Subject field after decoding the message from the protected headers. Using the plugin, you need to execute the <code>Email_PGPMessageDecrypt</code> function, after which the decoded message is passed to the <code>Email_ProcessMessageSource</code> function, then to get the value of the Subject field, use the <code>Email_ImapGetMessageField("Subject")</code> function.
	<code>{decryptPswd}</code>	You can set a passphrase to decode the message, then the recipient will be able to decode the message without the private key, using only the passphrase. If a key and passphrase were specified, the recipient has two options to decrypt the message using the keys or using the passphrase.

`{cipher}` Is the symmetric encryption algorithm. The default is "aes_256". Possible values are: "aes_128", "aes_192", "aes_256", "idea", "tripledes", "cast5", "blowfish", "twofish", "camellia_128", "camellia_192", "camellia_256", "sm4". The parameter is specified as a text string, case sensitive.

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

Emai_PGPMessageDecrypt

Description: You can decrypt the PGP message using the `Emai_PGPMessageDecrypt` function. The function does not check the digital signature. In order to check the digital signature, you need to pass the decrypted message to the `Emai_ProcessMessageSource` function, then execute the `Emai_ImapGetEncryptionType` function to make sure that the message has a digital signature. If the message has a digital signature, the signature can be verified using the `Emai_PGPMessageVerifySign` function. The result of the function execution is the decrypted source of the message, which can be saved in a container field or immediately passed as a parameter to the `Emai_ProcessMessageSource` function. Use the `Emai_GetLastResult` function to check whether the function completed successfully. Error codes are described in this document, in Chapter 9, Result Codes. Error codes in the range -100 ... -133.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:	<code>messageObject</code>	Message object received from the <code>Emai_ImapGetMessageSource</code> function.
	<code>{recipientPrivateKey}</code>	Recipient's private key, as an object.
	<code>{recipientPrivateKeyPswd}</code>	Password from the recipient's private key passed in the "recipientPrivateKey" parameter.
	<code>{decryptPswd}</code>	The passphrase that the sender used to create the encrypted message.

Emai_PGPMessageVerifySign

Description: The `Emai_PGPMessageVerifySign` function allows you to verify the digital signature of a message that was signed using PGP technology.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageObject</code>	Message object obtained from the <code>Emai_ImapGetMessageSource</code> function or the <code>Emai_SmimeMessageDecrypt</code> function.
<code>senderPublicKey</code>	The sender's public key, as an object.

Emai_GetEncryptionType

Description: The `Emai_GetEncryptionType` function is used to find out what technology was used to encrypt the message, S/MIME or PGP. The function is executed during the process of downloading messages from the server. The result of the function is a text string. Possible results of executing the function:

"pkcs7-mime" - if the message is encrypted using S/MIME technology;

"pkcs7-signature" - if the message contains S/MIME signature;

"pkcs7-signature-opaque" - if the message contains an S/MIME opaque signature;

"pgp-encrypted" - a message encrypted using PGP technology;

"pgp-signature" - if the message has a PGP signature.

The function returns an empty string if the message is not encrypted and does not have a digital signature

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_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 (`Email_PopGetMessageCount` has to be called to before retrieving message size).

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

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_PopRetrieveMessage

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_PopRetrieveAllMessages

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

Prerequisites: A POP3 connection has to be established using `Email_Connect` in a threaded context before using this function.

This function does not require `Email_PopGetMessageCount` call before use.

Compatibility: Win, macOS. Desktop only. On Server side use single-threaded functions.

Parameters: `{connectionUID}` Connection ID, created by the `Email_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.

Compatibility: Win, macOS, Linux. Desktop & Server

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

Emai_PopGetMessageSourceAsText

Description: Returns the entire source of the current message as text.

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.

Compatibility: Win, macOS, Linux. Desktop & Server

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_PopRetrieveMessage` or `Emai_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

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).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>fieldName</code>	<code>"from"</code> , <code>"to"</code> , <code>"cc"</code> , <code>"reply-to"</code> or <code>"sender"</code>
<code>fieldType</code>	<code>"name"</code> or <code>"address"</code>
<code>contactIndex</code>	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`.

Compatibility: Win, macOS, Linux. Desktop & Server

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 `"html"` 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.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_PopGetMessageHTML

Description: Returns the HTML code of the current message.

Prerequisites: Make a message the current message by invoking `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`. Invoking `Email_PopGetMessageInfo` alone prior to calling this function will not work.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_PopGetMessageCharset

Description: Returns the character set of the current message.

Prerequisites: Make a message the current message by invoking `Email_PopGetMessageInfo`, `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_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 `Email_PopGetMessageInfo`, `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_PopGetMessageField

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

Prerequisites: Make a message the current message by invoking `Email_PopGetMessageInfo`, `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_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 `Email_PopGetMessageInfo`, `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_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 `Email_PopGetMessageInfo`, `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_PopGetRecvTimestamp

Description: Returns the POP server receiving Timestamp of current message. For example, if your local time is "11/10/2019 9:10:30 AM" and the time offset is +0100, then UTC time will be "11/10/2019 8:10:30 AM".

Prerequisites: Make a message the current message by invoking `Email_PopGetMessageInfo`, `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: {type} "UTC" or "Local". The default is "UTC".

Email_PopGetMessageSize

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_PopDeleteMessage

Description: Deletes a specific message from POP server.

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

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_PopGetAttachmentCount

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

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_PopRetrieveMessage` 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).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>{attachmentType}</code>	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® 15-26 (64-bit only) . 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® 15-26 (64-bit only) .

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

Email_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 `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`. This function can not be called after invoking `Email_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Email_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>destinationPath</code>	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)
<code>{attachmentType}</code>	Type of attachment; "inline", "all" or "attachment" (default)
<code>{saveMode}</code>	"strict", "rename" (default) or "overwrite"

Email_PopGetAttachmentName

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Email_PopGetAttachmentSize

Description: Returns size in Bytes of the specified attachment.

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Email_PopGetAttachmentIcon

Description: Returns icon of the specified attachment as image.

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

Compatibility: Win, macOS. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>{attachmentType}</code>	Type of attachment; "inline", "attachment" or "all" (default)
<code>{iconSize}</code>	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 an 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_PopRetrieveMessage` 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).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Emai_ImapListFolders

Description: Returns the list of folders available for the current connection.
For IMAP connections returns IMAP folders.
For REST connections returns provider-specific folders (e.g. Gmail labels or Graph mail folders).

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: `{connectionUID}` Optional connection identifier returned by the `Emai_Connect` function.

Emai_ImapSelectFolder

Description: Selects a folder to be used with subsequent message retrieval functions. For example: `"Drafts"`.
If no folder is selected, the plug-in uses the `"Inbox"` folder by default.
For IMAP connections the folder parameter specifies IMAP folder name.
For REST connections the folder parameter specifies provider-specific folder (e.g. Gmail label or Graph mail folder).

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: `folder` Folder name to be selected.
 `{connectionUID}` Optional connection identifier returned by the `Emai_Connect` function.

Email_ImpGetMessageCount

Description: Returns the number of messages for current IMAP or REST connection waiting for download on server.

For single-threaded connections should be triggered prior to `Email_ImpRetrieveMessage` loop.

For multi-threaded connections should be triggered after `Email_ImpRetrieveAllMessages` call.

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

Compatibility: Win, macOS, Linux. Desktop & Server

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 `Email_ImpGetMessageUID` 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 `Email_Connect` plug-in function, used for multi-threaded receiving.

Email_ImpRetrieveAllMessagesCount

Description: Returns the number of messages for the current connection waiting for download on the server. Supported for IMAP and REST connections. Used for multi-threaded receiving.

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: connectionUID Connection identifier returned by the `Email_Connect` plug-in function, used for multi-threaded receiving.

 {messageUIDs} Optional list of message unique identifiers already stored in the database. The list must be separated by carriage return characters. The plug-in uses this list to determine which messages should be downloaded.

Emai_ImapGetMessageInfo

Description: Uses current IMAP or REST 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).

Compatibility: Win, macOS, Linux. Desktop & Server

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

Emai_ImapRetrieveMessage

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

Prerequisites: An IMAP or REST 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).

Compatibility: Win, macOS, Linux. Desktop & Server

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

Emai_ImapRetrieveAllMessages

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

Prerequisites: An IMAP or REST 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.

Compatibility: Win, macOS. Desktop only. On Server side use single-threaded functions.

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

Email_ImpGetMessageUID

Description: Returns current message IMAP or REST unique ID.

Prerequisites: An IMAP or REST connection has to be established using `Email_Connect` before using this function. Invoke `Email_ImpGetMessageInfo` or `Email_ImpRetrieveMessage` before using this function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>{connectionUID}</code>	This parameter should contain a connection ID, created by the <code>Email_Connect</code> plug-in function, used for multi-threaded receiving.
<code>{messageIndex}</code>	Index of new message on server (beginning with 1), used for multi-threaded receiving.

Email_ImpGetMessagesByFlags

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageFlags</code>	"seen", "unseen", "recent", "draft", "deleted", "flagged" or "answered".
---------------------------	--

Email_ImpGetDeleteUIDList

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

Prerequisites: An IMAP or REST connection has to be established using `Email_Connect` before using this function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageUIDL</code>	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 <code>Email_ImpGetMessageUID</code> function to obtain a UID for each downloaded message.
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Email_Connect</code> plug-in function, used for multi-threaded receiving.

Email_ImpGetDeleteUIDListInFolder**

Description: Returns a list of unique message IDs that are no longer available on the IMAP or REST server in the specified folder. This list should be used to delete those messages from the database, as they are no longer relevant.

Prerequisites: An IMAP or REST connection has to be established using `Email_Connect` before using this function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageUIDL</code>	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 <code>Email_ImpGetMessageUID</code> function to obtain a UID for each downloaded message.
<code>folderName</code>	Specify IMAP folder name.
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Email_Connect</code> plug-in function, used for multi-threaded receiving.

Email_ImpGetMessageSource

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

Prerequisites: An IMAP or REST connection has to be established using `Email_Connect` before using this function. Invoke `Email_ImpGetMessageInfo` or `Email_ImpRetrieveMessage` before using this function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Email_Connect</code> plug-in function, used for multi-threaded receiving.
<code>messageID</code>	Index of new message (beginning with 1) or message UID on server

Emai_ImapGetMessageSourceAsText

Description: Returns the entire source of the current message as text.

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Emai_Connect</code> plug-in function, used for multi-threaded receiving.
<code>messageID</code>	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_PopRetrieveMessage` or `Emai_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>fieldName</code>	"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_ImapRetrieveMessage` or `Emai_ProcessMessageSource`. Call `Emai_ImapGetContactsCount` before using this function to know the total number of contacts (needed for the `contactIndex` parameter).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Email_ImpGetMessageHeader

Description: Returns the header of the current message.

Prerequisites: Make a message the current message by invoking `Email_ImpGetMessageInfo`, `Email_ImpRetreiveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_ImpGetMessageBody

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 "html" into text and returns it as result of this function.

Prerequisites: Make a message the current message by invoking `Email_ImpRetreiveMessage` or `Email_ProcessMessageSource`. Invoking `Email_ImpGetMessageInfo` alone prior to calling this function will not work.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_ImpGetMessageHTML

Description: Returns the HTML code of the current message.

Prerequisites: Make a message the current message by invoking `Email_ImpRetreiveMessage` or `Email_ProcessMessageSource`. Invoking `Email_ImpGetMessageInfo` alone prior to calling this function will not work.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_ImpGetMessageCharset

Description: Returns the character set of the current message.

Prerequisites: Make a message the current message by invoking `Email_ImpGetMessageInfo`, `Email_ImpRetreiveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_ImpGetMessagePriority

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 `Email_ImpGetMessageInfo`, `Email_ImpRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Email_ImpGetMessageField

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

Prerequisites: Make a message the current message by invoking `Email_ImpGetMessageInfo`, `Email_ImpRetrieveMessage` or `Email_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_ImpGetMessageFlags

Description: Returns all 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 `Email_Connect` before using this function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: `{messageUID}` Specify the IMAP or REST 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 `Email_Connect` plug-in function, used for multi-threaded receiving.

Email_ImpSetMessagesFlags

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageFlags</code>	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.
<code>{messageUID}</code>	Specify the IMAP or REST message unique ID list, separated by carriage return character. Otherwise the plug-in will use the last retrieved message UID.
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Email_Connect</code> plug-in function, used for multi-threaded receiving.

Email_ImpRemoveMessagesFlags

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageFlags</code>	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.
<code>{messageUIDL}</code>	Specify the IMAP or REST message unique ID list, separated by carriage return character. Otherwise the plug-in will use the last retrieved message UID.
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Email_Connect</code> plug-in function, used for multi-threaded receiving.

Emai_ImapGetRecvDate

Description: Returns the mail 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_ImapRetrieveMessage` or `Emai_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Emai_ImapGetRecvTime

Description: Returns the mail 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_ImapRetrieveMessage` or `Emai_ProcessMessageSource`.

Compatibility: Win, macOS, Linux. Desktop & Server

Emai_ImapRecvTimestamp

Description: Returns the mail server receiving Timestamp of current message. For example, if your local time is "11/10/2019 9:10:30 AM" and the time offset is +0100, then UTC time will be "11/10/2019 8:10:30 AM".

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters: {type} "UTC" or "Local". The default is "UTC".

Emai_ImapGetMessageSize

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

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

Notes: For REST connections, the returned size may be approximate and include additional service-specific overhead (e.g. metadata and internal storage properties).

Emai_ImapDeleteMessage

Description: Deletes a specific message from mail 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.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageID</code>	Index of the message to be deleted (beginning with 1) or message UID on server
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Emai_Connect</code> 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_ImapRetrieveMessage` or `Emai_ProcessMessageSource`. This function can not be called after invoking `Emai_ImapGetMessageInfo` only since attachment information is not part of the message header.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>{attachmentType}</code>	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_PopRetrieveMessage` 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).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>{attachmentType}</code>	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® 15-26 (64-bit only) . 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® 15-26 (64-bit only) . 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.

Email_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 `Email_PopRetrieveMessage` or `Email_ProcessMessageSource`. This function can not be called after invoking `Email_PopGetMessageInfo` only since attachment information is not part of the message header. Invoke `Email_PopGetAttachmentCount` prior to using this function to know how many attachments the current message contains (needed for the `attachmentIndex` parameter).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>destinationPath</code>	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)
<code>{attachmentType}</code>	Type of attachment; "all", "inline" or "attachment" (default)
<code>{saveMode}</code>	"strict", "rename" (default) or "overwrite"

Email_ImapGetAttachmentName

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>destinationPath</code>	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)
<code>{attachmentType}</code>	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_PopRetrieveMessage` 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).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>destinationPath</code>	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)
<code>{attachmentType}</code>	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 an 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_PopRetrieveMessage` 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).

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>attachmentIndex</code>	Index of attachment from current message (starting with 1)
<code>destinationPath</code>	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)
<code>{attachmentType}</code>	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_PopRetrieveMessage` 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).

Compatibility: Win, macOS. Desktop & Server

Parameters:

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

Emai_ImapCreateFolder

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Emai_ImapRenameFolder**

Description: Rename a folder on the IMAP or REST server side.

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>oldFolderName</code>	The name of the folder that needs to be renamed.
<code>newFolderName</code>	New folder name.
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Emai_Connect</code> plug-in function, used for multi-threaded receiving.

Emai_ImapDeleteFolder**

Description: Delete a folder on the IMAP or REST server side.

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Emai_ImapUploadMessageToFolder

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

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>messageObject</code>	Message object to be uploaded. This can be either message object returned by the <code>Emai_SmtpComposeMessage</code> , <code>Emai_ImapGetMessageSource</code> or <code>Emai_PopGetMessageSource</code> functions; or this can be also a unique message ID, generated by the <code>Emai_SmtpCreateMessage</code> function.
<code>{folderName}</code>	Specify the target folder name. Otherwise, the plug-in will use the active folder. For example: "Sent Messages".
<code>{connectionUID}</code>	This parameter may optionally contain a connection ID, created by the <code>Emai_Connect</code> plug-in function, used for multi-threaded receiving.
<code>{internalDate}</code>	Sets the INTERNALDATE on the appended message (IMAP only). This parameter is not supported for REST connections and will be ignored. The value should be a date string that conforms to the RFC2060 specifications for a date_time value. Example: 30-Mar-2023 19:53:14 +0000.

Emai_ImapMoveMessage

Description: Moves a message from the current folder to the specified folder. Supported for IMAP and REST connections.

Prerequisites: A connection has to be established using `Emai_Connect` before using this function. For REST connections, a folder must be selected using `Emai_ImapSelectFolder`.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Emai_ImapSetSearch

Description:

Sets a search filter for the current folder to return a batch of messages matching the specified criteria. Supported for IMAP and REST connections.

Prerequisites:

A connection has to be established using `Emai_Connect` before using this function. For REST connections, a folder must be selected using `Emai_ImapSelectFolder`.

Compatibility:

Win, macOS, Linux. Desktop & Server

Parameters:

- | | |
|-----------------|---|
| criteria | String delimited by spaces. Multi-word arguments must be quoted. All criteria entries must match. |
| {connectionUID} | Optional connection identifier returned by the <code>Emai_Connect</code> plug-in function, used for multi-threaded receiving. |

Notes:

For IMAP, search syntax follows RFC3501 section 6.4.4. For REST connections, the same syntax is used and internally translated to provider-specific queries where possible.

Recommended criteria for cross-provider compatibility:

```
"ALL", "ANSWERED", "DELETED", "DRAFT", "FLAGGED", "SEEN", "UNSEEN",  
"UNANSWERED", "UNDELETED", "UNDRAFT", "UNFLAGGED", "FROM",  
"TO", "CC", "BCC", "SUBJECT", "BODY", "TEXT", "BEFORE", "SINCE",  
"ON", "LARGER", "SMALLER".
```

Emai_ImapClearSearch

Description: Resets the result of a function `Emai_ImapSearch`.

Compatibility: Win, macOS, Linux. Desktop & Server

Emai_ImapGetQuota

Description: The function returns information about storage usage and message quota. The result of the function can be in the form of a FileMaker List or as a specific value for the specified resource parameter.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

{resourceName}	If the parameter is empty (the default), it returns a line-separated list: storageUsed ¶ storageTotal ¶ messagesCount ¶ messagesLimit. If one of the parameters is specified: "storageUsed", "storageTotal", "messagesCount", "messagesLimit", the function returns only the value for the specified parameter.
{connectionUID}	This parameter may optionally contain a connection ID, created by the <code>Emai_Connect</code> plug-in function, used for multi-threaded receiving.

Emai_ImapGetFolderStatus

Description: Returns status information for the selected folder. The result of the function can be returned as a FileMaker List or as a specific value for the specified resource parameter.

The function allows you to obtain the following information: `exists` - the number of messages in the folder; `recent` - the number of recent messages; `unseen` - the number of unseen (new) messages in the folder; `nextUid` - the next UID to be used in the folder; `uidValidity` - a constant that changes when UIDs for the folder may no longer be valid.

For REST connections only `exists` and `unseen` values are returned; other values are empty.

Prerequisites: Use `Emai_ImapSelectFolder` before using this function.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

{resourceName}	If the parameter is empty (default), returns a line-separated list: nextUid ¶ uidValidity ¶ recent ¶ exists ¶ unseen. If one of the parameters is specified: "nextUid", "uidValidity", "recent", "exists", "unseen", the function returns only the value for the specified parameter.
{connectionUID}	Optional connection identifier returned by the <code>Emai_Connect</code> function.

Emai_ImapGetQuota

Description: Returns information about storage usage and message quota. The result can be returned as a FileMaker List or as a specific value for the specified resource parameter.

The function provides the following values: `storageUsed` - storage used (bytes); `storageTotal` - total storage limit (bytes); `messagesCount` - number of messages; `messagesLimit` - maximum allowed number of messages.

For REST connections values are returned on a best-effort basis; unsupported or unavailable values are returned as `"0"`.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>{resourceName}</code>	If the parameter is empty (default), returns a line-separated list: <code>storageUsed ¶ storageTotal ¶ messagesCount ¶ messagesLimit</code> . If one of the parameters is specified: <code>"storageUsed"</code> , <code>"storageTotal"</code> , <code>"messagesCount"</code> , <code>"messagesLimit"</code> , the function returns only the value for the specified parameter.
<code>{connectionUID}</code>	Optional connection identifier returned by the <code>Emai_Connect</code> function.

Emai_SmtpDestroyMessage

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

Compatibility: Win, macOS, Linux. Desktop & Server

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.

Compatibility: Win, macOS. Desktop only. On Server side use single-threaded functions.

Parameters:

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

Email_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 `Email_SmtpDestroyMessage` function.

Compatibility: Win, macOS, Linux. Desktop & Server

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"`).

Email_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.

Compatibility: Win, macOS, Linux. Desktop & Server

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; otherwise the header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>fieldName</code>	Name of the header field to be set, i.e. <code>"from"</code> , <code>"to"</code> or <code>"subject"</code>
<code>fieldValue</code>	Content of the message field; format depends on content type (<code>fieldName</code> parameter value). This can be an email address or a subject, for instance.
<code>{messageUID}</code>	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; otherwise the header field will be added to the current message. Otherwise, specify a message UID in order to affect a specific message object.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

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

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>priority</code>	<code>"Very High"</code> , <code>"High"</code> , <code>"Normal"</code> (default), <code>"Low"</code> or <code>"Very Low"</code> .
-----------------------	---

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

Email_SmtpAddBody

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

Prerequisites: Call `Email_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.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>format</code>	"text" for plain text, "html" for HTML code or "rich" for formatted FileMaker® text
<code>content</code>	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 HTML code to show the JPEG as inline (format parameter to be set to "html")
<code>{persistency}</code>	Indicates whether the body will persist after invoking <code>ResetMessage</code> function for next message; "once" (default) does not keep the body after <code>ResetMessage</code> ; "keep" keeps it if the <code>ResetMessage</code> function is invoked.
<code>{messageUID}</code>	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.

Compatibility: Win, macOS. Desktop only.

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.

Compatibility: Win, macOS. Desktop only.

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:

```

```

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

Compatibility: Win, macOS, Linux. Desktop & Server

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

Email_SmtpAddAttachment

Description: Adds an attachment to the current message

Prerequisites: Call `Email_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.

Compatibility: Win, macOS, Linux. Desktop & Server

Parameters:

<code>fileObject</code>	File object (from FM field) to be attached. If an external file is to be attached the function <code>Email_ImportFile</code> is called for this parameter. Will accept "binary file" object, as well as "image", "movie", and other container field types. Also will accept the <code>Email_ImportJpegClipboard</code> function result.
<code>{attachmentType}</code>	"all", "attachment" (default) or "inline"
<code>{fileName}</code>	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)
<code>{persistency}</code>	Indicates whether attachment will persist after calling <code>ResetMessage</code> function for next message; "once" (default) does not keep the attachment after <code>ResetMessage</code> ; "keep" keeps it.
<code>{contentID}</code>	The function <code>Email_SmtpGenerateContentID</code> 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.
<code>{windowsFriendly}</code>	When set to "windowsFriendly" value, on Mac OS X will make the plug-in avoid using the Apple Double encoding.
<code>{messageUID}</code>	Use this parameter to affect a specific message object.

Email_SmtpSendMessage

Description: Sends a message using the current SMTP or REST 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 or REST server using `Email_Connect` before invoking this function. Set message variables before sending a message (i.e. `Email_SmtpAddBody`). If sending a pre-composed message source object invoke `Email_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.

Compatibility: Win, macOS, Linux. Desktop & Server

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 `Email_Connect` plug-in function, used for multi-threaded sending.

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

Email_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.

Compatibility: Win, macOS, Linux. Desktop & Server

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

Emai_SmtpFalsifyAddress

Description: Validates an email address using SMTP without sending a message. The function tries to ask the receiving server if the specified address exists using SMTP commands. Such requests can be successful but they might also fail or return unreliable results. Only use this function for mass mailing solutions. Do not rely on the result. This function is supported for SMTP connections only and is not available for REST connections.

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).

Compatibility: Win, macOS, Linux. Desktop & Server

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 11 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_ShowFileDialog` 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 `Email_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:

```

```

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 HTML code of the 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 <code>authType</code> parameter in the <code>Email_Connect</code> 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 <i>exactly</i> as supplied)
032	""	Reserved (not returned by plug-in)
033	"Not connected"	Connect to server first
034	"No received message"	Call <code>Emai_ProcessMessageSource</code> or <code>Emai_PopRetrieveMessage</code> 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"	<code>Emai_GetLastResultCode</code> 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.
-53	"Cannot access POP3 history"	The plug-in could not access the POP3 message history. Check that the history file exists and is readable and writable by the current user or server process.

Code	Explanation	Possible Cause
-54	"Bad server response"	The server returned an invalid, incomplete, or unexpected response that could not be processed by the plug-in.
-55	"Server error"	The server returned an error response while processing the request. Check the server reply in the log file.
-56	"Too many requests"	The server rejected the request because too many requests were sent in a short period of time. Wait before retrying the operation.
-57	"Connection timed out"	The connection timed out before the server completed the operation. Possible causes include network problems, server overload, or an incorrect timeout setting.
-58	"Out of memory"	The plug-in could not allocate enough memory to complete the operation. Close unused files or applications and try again.
061	"The local web server is not running"	The server was not started or problems occurred during startup attempts.
062	"Cannot open url. Please check this url is correct"	The specified URL refers to a resource that does not exist.
065	"Incorrect JSON string"	Check the JSON string passed to the function, maybe the string has syntax errors or does not match the JSON format at all
066	"Key not found in JSON data"	The wrong key may be specified or the specified key does not exist
071	"Unable to save settings. The settings/license file is not writable."	It may be due to the plugin not having access rights to the local disk for reading/writing on behalf of the current user
100	"Signature is valid but expired"	The signature was verified successfully, but the signing certificate or key validity period has expired.
101	"Public key to verify signature was not available"	The required public key was not found or was not supplied to the verification function.
102	"Data or signature was modified"	The signed data, detached signature, or message content no longer matches the original signature.
103	"Signature has unknown format"	The signature format is not recognized or is not supported by the plug-in.

Code	Explanation	Possible Cause
104	"Failed to read public key"	The public key data is missing, corrupted, incorrectly formatted, or cannot be loaded.
105	"Failed to read private key"	The private key data is missing, corrupted, incorrectly formatted, encrypted with an incorrect password, or cannot be loaded.
106	"Failed to read signature"	The signature data is missing, corrupted, incomplete, or incorrectly formatted.
107	"Could not determine the beginning of the OpenPGP/MIME part of the message"	The message does not contain a recognizable OpenPGP/MIME part or the MIME structure is invalid.
108	"Could not determine the beginning of the boundaries"	The MIME boundary marker is missing, damaged, or cannot be parsed.
109	"Failed to determine first boundaries"	The first MIME boundary required to parse the signed or encrypted part could not be found.
110	"Failed to determine last boundary"	The closing MIME boundary required to finish parsing the signed or encrypted part could not be found.
111	"Failed to create output object"	The plug-in could not create an internal output object for the cryptographic operation.
112	"Failed to create input object"	The plug-in could not create an internal input object for the cryptographic operation.
113	"Failed to initialize FFI object"	The cryptographic library interface could not be initialized.
114	"Failed to create sign operation"	The plug-in could not initialize the signing operation context.
115	"Failed to sign"	The signing operation failed. Check the private key, key password, input data, and cryptographic library availability.
116	"Failed to create encrypt operation"	The plug-in could not initialize the encryption operation context.
117	"Failed to execute encryption operation"	Encryption failed. Check recipient keys, passwords, input data, and cryptographic library availability.
118	"Failed to locate signing key"	The requested signing key was not found in the provided key data or key store.
119	"Failed to add signature for key"	The selected key could not be added to the signing operation.

Code	Explanation	Possible Cause
120	"Failed to get memory buffer"	The plug-in could not allocate or retrieve the internal memory buffer for the operation result.
121	"Failed to add recipient"	The recipient key or recipient data could not be added to the encryption operation.
122	"Failed to export a key"	The key export operation failed. The key may be invalid, unavailable, or not exportable.
123	"Failed to add encryption password"	The password could not be added to the encryption operation. Check that the password value is valid.
124	"Failed to create verification operation context for detached signature"	The plug-in could not initialize the verification context for a detached signature.
125	"Failed to execute verification operation"	Signature verification failed due to invalid input data, invalid signature data, missing key data, or a cryptographic library error.
126	"Failed to get signature"	The signature object could not be extracted from the verification result.
127	"Failed to get key id"	The key identifier could not be read from the signature or key data.
128	"Failed to get signature's key"	The public key corresponding to the signature could not be found or extracted.
129	"Failed to decrypt encrypted data"	Decryption failed. Check the private key, password, encrypted data, and recipient key.
130	"The beginning of the pgp public key block could not be determined"	The input does not contain a recognizable beginning marker for a PGP public key block.
131	"The end of the pgp public key block could not be determined"	The input does not contain a recognizable ending marker for a PGP public key block.
132	"The beginning of the pgp signature block could not be determined"	The input does not contain a recognizable beginning marker for a PGP signature block.
133	"The end of the pgp signature block could not be determined"	The input does not contain a recognizable ending marker for a PGP signature block.
134	"No suitable key"	No suitable key was found for the requested cryptographic operation.
135	"Failed to convert PKCS#7 structure to S/MIME format"	The PKCS#7 data could not be converted into a valid S/MIME message format.

Code	Explanation	Possible Cause
136	"Failed to constructs a BIO chain in preparation for putting data into or retrieving data out of p7"	The plug-in could not create the internal BIO chain required for the PKCS#7 operation.
-140	"MailIt 11 license is invalid."	The license key is missing, corrupted, incorrectly formatted, or cannot be validated by the plug-in.
-141	"MailIt 11 trial period has expired."	The installed trial license has reached its expiration date.
-142	"MailIt 11 subscription has expired."	The subscription license has reached its expiration date.
-143	"This perpetual MailIt 11 license does not cover this plug-in build."	The installed plug-in was built after the maintenance or updates period covered by the perpetual license.
-144	"The registration name does not match the license key."	Enter the registration name exactly as it was issued with the license key.
-145	"This license is not valid for the current platform."	The license does not include the operating system platform where the plug-in is currently running.
-146	"This license does not allow FileMaker Server use."	The license does not include FileMaker Server usage rights.
-147	"This license does not include a desktop client seat."	The license does not include a seat for FileMaker Pro/Desktop use.
-148	"Could not remove the MailIt 11 license file."	The license file could not be deleted because it is missing, locked, or not writable by the current user or server process.
-149	"Internal MailIt 11 licensing error."	An unexpected internal licensing error occurred. Please refer to Dacons tech support and attach the log file.

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 attachments. 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.

MailIt plug-in is compatible with Runtime Solutions FileMaker® 15 - 18 (64bit only)

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.