



Mail Manipulator

A blue sphere with a white envelope icon inside, representing the Mail Manipulator application.

Developer's Guide

Revised October 26, 2011

Table of Contents

I. Introduction	3
II. Integration Steps	4
1) Installing the Plug-in	4
2) Registering the Plug-in	4
3) Selecting a Mail Account	6
4) Select a Folder.....	7
5) Importing Mail including Attachments.....	8
6) Sending Mail	11
7) Other Features.....	12
Move Mail.....	12
Display Mail	13
Get Default Mail Account	14
Delete Mails.....	14
8) Handling Errors	15
III. Contact Us	16

I. Introduction

Description:

The Mac Mail plug-in is a tool used to exchange data between FileMaker® and Mac Mail®. With this plug-in FileMaker users are able to bidirectionally exchange data between FileMaker and Mac Mail. More specifically you can import mail including attachments from Mac Mail into FileMaker, send mail with multiple attachments from FileMaker to Mac Mail, and move mail across folders within Mac Mail. These operations are accomplished using FileMaker function calls from within FileMaker calculations. These calculations are generally determined from within FileMaker "SetField," "SetVariable" or "If" script steps. For a list of the basic integration steps, please see the accompanying Developers Guide document.

Intended Audience:

FileMaker developers or persons, who have knowledge of FileMaker scripting, calculations and relationships as proper use of the plug-in requires that FileMaker integration scripts be created in your FileMaker solution.

Successful Integration Practices:

- 1) Read the Developer's Guide
- 2) Read the Functions Guide
- 3) Review our FileMaker demo and video tutorials

Demo and video tutorials can be found here: <http://www.productivecomputing.com/plugins/mail-manipulator-id-53/>

- 4) Familiarize yourself with Mac Mail

Error Handling:

Any of the plug-in functions may encounter an error during processing. When an error occurs during processing, immediately call the PCMM_GetLastError function in order to obtain a full description of the error or error number. This function returns the error message or error number associated with the last error in order to troubleshoot script or logic failures. Please see PCMM_GetLastError function description in the Functions's Guide and the "Handling Errors" section in the Developer's Guide for further clarification on how to properly trap for errors.

II. Integration Steps

Accessing and using the plug-in functions involve the following steps.

1) Installing the Plug-in

The first step is to install the plug-in into FileMaker.

- 1) Quit FileMaker Pro completely.
- 2) Locate the plug-in in your download which will be located in a folder called "Plug-in and Installer". On Mac the plug-in will have a ".fmplugin" extension.
- 3) Copy the actual plug-in and paste it to the Extensions folder which is inside the FileMaker program folder. On Mac this is normally located here: Volume/Applications/FileMaker X/Extensions (Volume is the name of the mounted volume).
- 4) Start FileMaker. Confirm that the plug-in has been successfully installed by navigating to "Preferences" in FileMaker Pro, then click the "Plug-ins" Tab. There you should see the plug-in listed with a corresponding check box. This indicates that you have successfully installed the plug-in.

2) Registering the Plug-in

The next step is to register the plug-in which enables all plug-in functions.

- 5) Confirm that you have access to the internet and open our FileMaker demo file, which can be found in the "FileMaker Demo File" folder in your original download.
- 6) If you are registering the plug-in in Demo mode, then simply click the "Register the Plug-in" button and do not change any of the fields. Your plug-in should now be running in "DEMO" mode. The mode is noted in our FileMaker Demo file on the Setup tab.
- 7) If you are registering a licensed copy, then simply enter your license number in the "LicenseID" field and click the "Register the Plug-in" button. Make sure you remove the Demo License ID and enter your registration information exactly as it appears in your confirmation email. Your plug-in should now be running in "LIVE" mode. The mode is indicated on the Setup tab in our FileMaker Demo file or by calling the PCMM_GetOperatingMode function.

Congratulations! You have now successfully installed and registered the plug-in!

Why do I need to Register?

In an effort to reduce software piracy, Productive Computing, Inc. has implemented a registration process for all plug-ins. The registration process sends information over the internet to a server managed by Productive Computing, Inc. The server uses this information to confirm that there is a valid license available and identifies the machine. If there is a license available, then the plug-in receives an acknowledgment from the server and installs a certificate on the machine. This certificate never expires. If the certificate is ever moved, modified or deleted, then the client will be required to register again. On Mac this certificate is in the form of a plist file.

The registration process also offers developers the ability to automatically register each client machine behind the scenes by hard coding the license ID in the PCMM_Register function. This proves beneficial by eliminating the need to manually enter the registration number on each client machine. There are other various functions available such as PCMM_GetOperatingMode and PCMM_Version which can assist you when developing an installation and registration process in your FileMaker solution.

How do I hard code the registration process?

You can hard code the registration process inside a simple "Plug-in Checker" script. The "Plug-in Checker" script should be called at the beginning of any script using a plug-in function and uses the PCMM_Register, PCMM_GetOperatingMode and PCMM_Version functions. This eliminates the need to manually register each machine and ensures that the plug-in is installed and properly registered. Below are the basic steps to create a "Plug-in Checker" script.

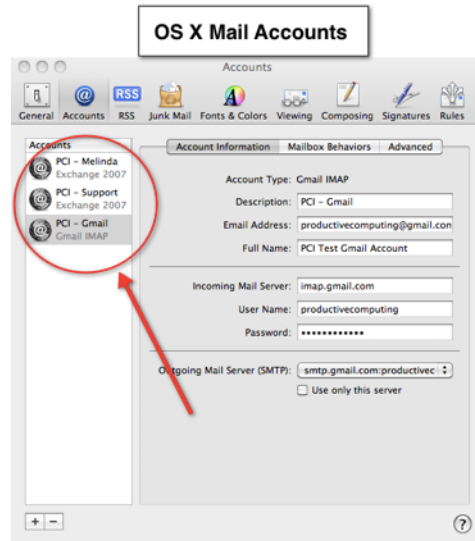
```
If [ PCMM_Version( "short" ) = "" or PCMM_Version( "short" ) = "?" ]  
Show Custom Dialog [ Title: "Warning"; Message: "Plug-in not installed."; Buttons: "OK" ]  
If [ PCMM_GetOperatingMode ≠ "LIVE" ]  
Set Field [Main::gRegResult; PCMM_Register( "licensing.productivecomputing.com" ; "80" ; "/PCIReg/pcireg.asp" ;  
"your license ID" )  
If [ Main::gRegResult ≠ 0 ]  
Show Custom Dialog [ Title: "Registration Error"; Message: "Plug-in Registration Failed"; Buttons: "OK" ]
```

Please also watch our setup video found here: www.productivecomputing.com/video/?p=1286 for additional setup information.

3) Selecting a Mail Account

When communicating with OS X Mac Mail, then you must first select the Mac Mail account that you would like to work with as Mac Mail allows you to setup multiple accounts as shown in figure 3.0 below.

Figure 3.0 - Account settings from Mac Mail with 3 different accounts



You can retrieve the names of all the accounts available in Mac Mail by calling the PCMM_GetFirstAccount and PCMM_GetNextAccount functions as shown in figure 3.1.

Figure 3.1 - Sample script looping through and obtaining the OS X Mail account names.

```
♦ #Get First Account.
♦ Set Variable [$Account; Value:PCMM_GetFirstAccount]
♦ If [$Account = "End"]
♦ #No accounts exist.
♦ Close Window [Current Window]
♦ Show Custom Dialog ["No accounts in Mac Mail."]
♦ Exit Script []
♦ Else If [$Account = "!ERROR!"]
♦ #Error with getting first account.
♦ Close Window [Current Window]
♦ Show Custom Dialog ["Error"; PCMM_GetLastError("Text")]
♦ Exit Script []
♦ Else
♦ #Add record with first account name.
♦ New Record/Request
♦ Set Field [Accounts::Account Name; $Account]
♦ End If
♦ #Get remaining Accounts.
♦ Loop
♦ Set Variable [$Account; Value:PCMM_GetNextAccount]
♦ If [$Account = "!ERROR!" and $Account ≠ "End"]
♦ #Add record with next folder name.
♦ New Record/Request
♦ Set Field [Accounts::Account Name; $Account]
♦ End If
♦ Exit Loop If [$Account = "End"]
♦ End Loop
♦ Close Window [Current Window]
♦ Commit Records/Requests [Skip data entry validation; No dialog]
♦ Go to Field [Main::gAccounts]
```

Of course you can construct your scripts however you desire and these are just samples taken from our demo file. Afterwards you call PCMM_OpenAccount(Name) and pass the desired Mac Mail account name to the "Name" parameter such as PCEM_OpenAccount("PCI - Melinda") using one of the example accounts above.

4) Select a Folder

Next we must specify the desired folder. This is accomplished by first setting the folder delimiter using PCMM_SetFolderDelimiter, then by calling PCMM_GetFirstFolder and PCMM_GetNextFolder to obtain a list of all available folders as shown in figure 4.0.

Figure 4.0 - Sample script setting the folder delimiter and obtaining the folder names.

```
✦ #Set Folder Delimiter
✦ Set Variable [$Result; Value:PCMM_SetFolderDelimiter( "|" )]
✦ #Get First Folder
✦ Set Variable [$Folder; Value:PCMM_GetFirstFolder]
✦ If [$Folder = "End"]
✦ #No folders exist for the selected account.
✦ Close Window [Current Window]
✦ Show Custom Dialog ["No Mail folders for the selected account."]
✦ Exit Script []
✦ Else If [$Folder = "!!ERROR!!"]
✦ #Error with getting first folder.
✦ Close Window [Current Window]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ Else
✦ #Add record with first folder name.
✦ New Record/Request
✦ Set Field [Folders::Folder Name; $Folder]
✦ End If
✦ #Get remaining folders.
✦ Loop
✦ Set Variable [$Counter; Value:$Counter + 1]
✦ Set Variable [$Folder; Value:PCMM_GetNextFolder]
✦ If [Mod($Counter;1) = 0]
✦ #Present progress count for user with every folder created.
✦ Set Field [Main::gProgress String; $Counter & " folders imported."]
✦ Refresh Window []
✦ End If
✦ If [$Folder ≠ "!!ERROR!!" and $Folder ≠ "End"]
✦ #Add record with next folder name.
✦ New Record/Request
✦ Set Field [Folders::Folder Name; $Folder]
✦ End If
✦ Exit Loop If [$Folder = "End" or $Folder = "End"]
✦ End Loop
✦ Close Window [Current Window]
✦ Commit Records/Requests [Skip data entry validation; No dialog]
✦ Go to Field [Main::gFolder]
```

Once you know the folder location and name you would like to open, then call the PCMM_OpenFolder (FolderPath) function and pass the folder path to the FolderPath parameter such as PCMM_OpenFolder ("Inbox/Sales") or see figure 4.1 below.

Figure 4.1 - Sample script of how to open a folder.

```
✦ #Open folder to work with.
✦ If [0 ≠ PCMM_OpenFolder( Main::gFolder )]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" ) & " Please click the \"Refresh Folder List\" button."]
✦ Exit Script []
✦ End If
✦ #Get Record Count
```

5) Importing Mail including Attachments

Now that you have decided what Mac mail account and folder to work with, you can import mail from this folder. First you will want to obtain a record count using PCMM_GetRecordCount as shown in figure 5.0.

Figure 5.0

```
✦ #Get Record Count
✦ Set Variable [$RecordCount; Value:PCMM_GetRecordCount]
✦ If [PatternCount($RecordCount; "!!ERROR!!")]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
```

Then you can call PCMM_GetFirstRecord, enter your loop and get all desired field values using the PCMM_GetFieldData(FieldName) function as shown in figure 5.1 below.

Figure 5.1

```
✦ #Get First record (email).
✦ Set Variable [$ID Mail; Value:PCMM_GetFirstRecord]
✦ If [$ID Mail = "End"]
✦   #No mail records exist for the selected folder.
✦   Close Window [Current Window]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ Else If [$ID Mail = "!!ERROR!!"]
✦   #Error with getting first record.
✦   Close Window [Current Window]
✦   Refresh Window []
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ Else
✦   Loop
✦     #Add record
✦     Set Variable [$Counter; Value:$Counter + 1]
✦     New Record/Request
✦     Set Field [Email::ID Mac Mail; $ID Mail]
✦     Set Field [Email::Direction; "Incoming"]
✦     Set Field [Email::Sender; PCMM_GetFieldData( "Sender" )]
✦     Set Field [Email::To; PCMM_GetFieldData( "To" )]
✦     Set Field [Email::CC; PCMM_GetFieldData( "Cc" )]
✦     Set Field [Email::BCC; PCMM_GetFieldData( "Bcc" )]
✦     Set Field [Email::Subject; PCMM_GetFieldData( "Subject" )]
✦     Set Field [Email::Body; PCMM_GetFieldData( "Body" )]
✦     Set Field [Email::Header; PCMM_GetFieldData( "Header" )]
✦     Set Field [Email::Size; PCMM_GetFieldData( "Size" )]
✦     Set Field [Email::Received; PCMM_GetFieldData( "Received" )]
✦     Set Field [Email::Sent; PCMM_GetFieldData( "Sent" )]
✦     Set Field [Email::Read; PCMM_GetFieldData( "Read" )]
✦     Set Field [Email::Forwarded; PCMM_GetFieldData( "Forwarded" )]
✦     Set Field [Email::Redirected; PCMM_GetFieldData( "Redirected" )]
✦     Set Field [Email::Replied; PCMM_GetFieldData( "Replied" )]
✦     Set Field [Email::ReplyTo; PCMM_GetFieldData( "ReplyTo" )]
✦     Set Field [Email::Flagged; PCMM_GetFieldData( "Flagged" )]
✦     Set Field [Email::Junked; PCMM_GetFieldData( "Junked" )]
✦     Set Field [Email::Folder; Main::gFolder]
✦     Set Field [Email::Account; Main::gAccount]
```

For a complete list of field names see the "Available Fields" section in the accompanying Functions Guide.

Then if there are still records in the count you can call PCMM_GetNextRecord as shown in figure 5.2 continue through the loop and import the additional mail records in the specified folder.

Figure 5.2

```
✦ #  
✦ Set Variable [$SID Mail; Value:PCMM_GetNextRecord]  
✦ If [$SID Mail = "!!ERROR!!"]  
✦ #Error with getting next record.  
✦ Close Window [Current Window]  
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]  
✦ Exit Script []  
✦ End If  
✦ Exit Loop If [$SID Mail = "End"]  
✦ End Loop  
✦ End If
```

You will most likely want to determine if the mail records have attachments and import the attachments into FileMaker. This can be accomplished with the PCMM_GetAttachmentCount function as shown in figure 5.3.

Figure 5.3

```
✦ #Import Attachments if applicable.  
✦ If [$SaveAttachments]  
✦ #Determine if there are attachments.  
✦ Set Variable [$AttachmentCount; Value:PCMM_GetAttachmentCount]  
✦ Set Variable [$SID PK; Value:Email::ID_PK]  
✦ If [$AttachmentCount = "!!ERROR!!"]  
✦ Close Window [Current Window]  
✦ Refresh Window []  
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]  
✦ Exit Script []  
✦ Else If [$AttachmentCount > 0]  
✦ #Prep data for processing dialog on the attachments screen.  
✦ Set Field [Main::gProgress String; "Subject: " & Email::Subject & ¶ & "Sender: " & Email::Sender & ¶ & "To: " & Email::To]  
✦ #Import Attachments  
✦ Perform Script ["Import Attachments (store as a reference)"]  
✦ End If  
✦ End If
```

If attachments exist, then the next step would be to get the name of the attachments using PCMM_GetAttachmentName, followed by PCMM_SaveAttachment to save the attachment in FileMaker as shown in figure 5.4 below.

Please note that in order to save the attachment in FileMaker from Mac Mail, the attachment must first be saved to a folder on a hard drive. Typically you would use a temporary folder as once the attachment has been successfully imported in FileMaker, then the attachment is no longer needed on your hard drive. The folder on your hard drive works as a middle man to transport the attachment from Mac Mail to FileMaker.

Figure 5.4

```

✦ #This script is a subscript called from the parent script "Import Email."
✦ Set Variable [$Counter; Value:0]
✦ Close Window [Name: "Attachments"; Current file]
✦ New Window [Name: "Attachments"]
✦ Adjust Window [Hide]
✦ Go to Layout ["Attachments" (Attachments)]
✦ Loop
✦ Set Variable [$Counter; Value:$Counter + 1]
✦ Set Variable [$AttachmentName; Value:PCMM_GetAttachmentName( $Counter )]
✦ If [$AttachmentName = "!!ERROR!!"]
✦ Close Window [Current Window]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ End If
✦ #Pull attachments
✦ New Record/Request
✦ #Parse the attachment extension (i.e. .gif, .pdf, .doc, .xls, etc.)
✦ Set Variable [$AttachmentExtension; Value:Right($AttachmentName; Length($AttachmentName) - Position($AttachmentName;".";1)+1)]
✦ Set Field [Attachments::ID_Email; $$ID PK]
✦ #Set the path for the attachment.
✦ If [GetAsNumber ( Get ( ApplicationVersion ) ) ≥ 8 and GetAsNumber ( Get ( ApplicationVersion ) ) < 9 // For FileMaker 8 or 8.5]
✦ #FMP 8 or 8.5 (use Desktop)
✦ Set Variable [$PathFM; Value:"filemac:" & Get ( DesktopPath ) & Attachments::ID_PK & $AttachmentExtension]
✦ Set Variable [$PathMac; Value:Get ( DesktopPath ) & Attachments::ID_PK & $AttachmentExtension]
✦ Else
✦ #FMP 9 or greater (use Temporary Folder)
✦ Set Variable [$PathFM; Value:"filemac:" & Get ( TemporaryPath ) & Attachments::ID_PK & $AttachmentExtension]
✦ Set Variable [$PathMac; Value:Get ( TemporaryPath ) & Attachments::ID_PK & $AttachmentExtension]
✦ End If
✦ #Save Attachment to specified path (on the hard drive).
✦ If [0 ≠ PCMM_SaveAttachment( $PathMac ; $Counter )]
✦ Close Window [Current Window]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ End If
✦ Insert File [Reference; Attachments::Item; "$PathFM"]
✦ If [Get(LastError)]
✦ #Test for errors inserting the file.
✦ Show Custom Dialog ["Error"; "There was a problem with FileMaker inserting the attachment. FileMaker Error: " & Get(LastError)]
✦ Exit Script []
✦ End If
✦ Set Field [Attachments::Original Name; $AttachmentName]
✦ Exit Loop If [$Counter = $$AttachmentCount]
✦ End Loop
✦ Set Variable [$$AttachmentCount; Value:""]
✦ Close Window [Current Window]

```

6) Sending Mail

Sending emails requires a few less steps. You still need to open a OS X Mail account as mentioned previously in the "3) Selecting a Mail Account" section above. Then you call the PCMM_SendMail(To ; CC ; Bcc ; Subject ; Body ; Attachments) function. This function allows you to send mail with multiple attachments in a single script step.

The examples below demonstrate how to hard code values or how to reference values in the parameters of the PCMM_SendMail function.

Example of sending an email by hardcoding parameter values:

```
- PCMM_SendMail( "joe@somecompany.com" ; "frank@anothercompany.com" ;  
"manager@somecompany.com" ; "New Release" ; "A new product was just released ... please contact us!" ;  
"/SomeHD/Users/Paul/Documents/SomeFile.pdf" )
```

Example of sending an email by using field names:

```
PCMM_SendMail( Email::To; Email::CC; Email::BCC; Email::Subject; Email::Body; Attachment::Path )
```

The Attachment::Path represent the field that stores the full file path.

Figure 6.0 - example of using the PCMM_SendMail function in a script.

```
◆ #Setup  
◆ Allow User Abort [On]  
◆ Set Error Capture [On]  
◆ Perform Script ["Check for Browse Mode"]  
◆ #Validate that the plug-in is ready.  
◆ Perform Script ["Plug-in Checker"]  
◆ #  
◆ If [IsEmpty(Email::To) and IsEmpty(Email::CC) and IsEmpty(Email::BCC)]  
◆ #Validate that there is a recipient in any field.  
◆ Show Custom Dialog ["There is no recipient in the \"To\" \"Cc\" or \"Bcc\" field."  
◆ Exit Script []  
◆ End If  
◆ #Open Account to work with.  
◆ If [0 ≠ PCMM_OpenAccount( Main::gAccount )]  
◆ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]  
◆ Exit Script []  
◆ End If  
◆ #Send Email: Note we only include options to send the first 10 attachments.  
◆ If [0 ≠ PCMM_SendMail( Email::To; Email::CC; Email::BCC; Email::Subject; Email::Body; Case ( IsValid ( GetNthRecord ( Attachments::cPath ; 1 ) ) ;  
◆ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]  
◆ Exit Script []  
◆ Else  
◆ Show Custom Dialog ["Mail has been sent."  
◆ End If
```

7) Other Features

Move Mail

In order to move mail from one Mac Mail folder to another you call the `PCMM_MoveRecord(FolderPath)` function and pass the destination folder path to the "FolderPath" parameter. Before you can move mail you must first open the desired account, open the folder containing the mail record and open the mail record you desire to move. These steps are explained in the previous sections of this document and are illustrated in figure 7.0 below.

Figure 7.0

```
✦ #Validate Destination Folder
✦ If [IsEmpty ( Main::gFolder Destination )]
✦   Show Custom Dialog ["Warning"; "You cannot move mail until you've selected a destination folder."]
✦   Exit Script []
✦ End If
✦ If [IsEmpty( Email::ID Mac Mail )]
✦   #Validate existence of Mail ID
✦   Show Custom Dialog ["Warning"; "You cannot move an email that doesn't have a Mail ID.¶Pull Email into FileMaker first before attem..."]
✦   Exit Script []
✦ End If
✦ If [Email::Folder = Main::gFolder Destination]
✦   #Validate unique folders
✦   Show Custom Dialog ["Warning"; "Your destination folder is the same as the source folder.  Change your destination folder."]
✦   Exit Script []
✦ End If
✦ #Open Account to work with.
✦ If [0 ≠ PCMM_OpenAccount( Main::gAccount )]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
✦ #Open folder to work with.
✦ If [0 ≠ PCMM_OpenFolder( Email::Folder )]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" ) & " Please click the \"Refresh Folder List\" button."]
✦   Exit Script []
✦ End If
✦ #Open Record to work with.
✦ If [0 ≠ PCMM_OpenRecord (Email::ID Mac Mail) ≠ 0]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
✦ #Move Mail
✦ If [0 ≠ PCMM_MoveRecord( Main::gFolder Destination )]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
✦ #Update folder field to represent the new location for this email.
✦ Set Field [Email::Folder; Main::gFolder Destination]
✦ Set Field [Main::gFolder Destination; "" ]
✦ Commit Records/Requests [No dialog]
✦ Show Custom Dialog ["You have successfully moved this mail to the " & Email::Folder & " folder."]
```

Moving mail may be an effective way to organize mail that has already been pulled into FileMaker and gives you additional options for mail organization and management in your solution.

Display Mail

In order to display the desired mail record in Mac Mail directly from FileMaker you call the PCMM_Display (MacMailID) function and pass the unique Mac Mail ID of the mail record to be displayed. Before you can display a mail record you must first open the desired account, open the folder containing the mail record and open the mail record you desire to display. These steps are explained in the previous sections of this document and are illustrated in figure 7.1 below

Figure 7.1

```
✦ #Setup
✦ Allow User Abort [On]
✦ Set Error Capture [On]
✦ Perform Script ["Check for Browse Mode"]
✦ #Validate that the plug-in is ready.
✦ Perform Script ["Plug-in Checker"]
✦ #Check for Mail ID
✦ If [IsEmpty(Email::ID Mac Mail)]
✦   Show Custom Dialog ["Warning"; "You cannot display a record that doesn't have a Mail ID."]
✦   Exit Script []
✦ End If
✦ #
✦ #Open Account to work with.
✦ If [0 ≠ PCMM_OpenAccount( Main::gAccount )]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
✦ #Open Folder to work with.
✦ If [0 ≠ PCMM_OpenFolder( Main::gFolder )]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
✦ #Open Record to work with.
✦ If [0 ≠ PCMM_OpenRecord( Email::ID Mac Mail)]
✦   Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
✦ #Display
✦ Set Variable [$Result; Value:PCMM_Display]
✦ If [0 ≠ PCMM_Display]
✦   Show Custom Dialog [PCMM_GetLastError( "Text" )]
✦   Exit Script []
✦ End If
```

Displaying email records gives you a convenient way to view the email record in FileMaker directly in Mac Mail. This way you do not have to rummage through your Mac Mail application to find the corresponding email in FileMaker.

Get Default Mail Account

Since OS X Mail allows you to open various mail accounts, you can call the PCMM_GetFirstAccount function to retrieve the first or default Mac Mail account as shown in figure 7.2 below.

Figure 7.2

```
✦ #Setup
✦ Allow User Abort [On]
✦ Set Error Capture [On]
✦ Perform Script ["Check for Browse Mode"]
✦ #Validate that the plug-in is ready.
✦ Perform Script ["Plug-in Checker"]
✦ #
✦ #Get Default Account
✦ Set Variable [$Default Account; Value:PCMM_GetFirstAccount]
✦ If [$Default Account ≠ "!!ERROR!!"]
✦ Show Custom Dialog ["Default Mail Account is: " & $Default Account]
✦ Else
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ End If
```

Delete Mails

The 3 functions available to delete records are PCMM_DeleteRecord(MacMailID), PCMM_DeleteAllRecords and PCMM_EmptyAllDeletedItems. Please see the accompanying Functions Guide for the differences between each function. We also encourage you to use these functions with caution as you do not want to permanently delete important emails. Figure 7.3 illustrates the functions necessary to call the PCMM_DeleteRecord function.

Figure 7.3

```
✦ #Check for Mail ID
✦ If [IsEmpty(Email::ID Mac Mail)]
✦ Show Custom Dialog ["Warning"; "You cannot delete a record that doesn't have a Mail ID."]
✦ Exit Script []
✦ End If
✦ #
✦ #Open Account to work with.
✦ If [0 ≠ PCMM_OpenAccount( Main::gAccount )]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ End If
✦ #Open Folder to work with.
✦ If [0 ≠ PCMM_OpenFolder( Main::gFolder )]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ End If
✦ #Open Record to work with.
✦ If [0 ≠ PCMM_OpenRecord( Email::ID Mac Mail)]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ End If
✦ #Delete Mail
✦ If [0 ≠ PCMM_DeleteRecord( Email::ID Mac Mail)]
✦ Show Custom Dialog ["Error"; PCMM_GetLastError( "Text" )]
✦ Exit Script []
✦ Else
✦ #Clear the "ID Mail" field now that we deleted the message in Mail.
✦ Set Field [Email::ID Mac Mail; "" ]
✦ End If
```

8) Handling Errors

When something unexpected happens, a plug-in function will return a result of !!ERROR!!. This makes it simple to check for errors. If a plug-in function returns !!ERROR!!, then immediately after call PCMM_GetLastError("Text") function for a detailed description of what the exact error was or PCMM_GetLastError("Number") for an error number.

We find that most developers run into issues due to a lack of error trapping. Please ensure that you properly trap for errors in your solutions. Here are a few samples of how you can check for errors.

```
Set Variable [ $result = MyPluginFunction( "a" ; "b" ; "c" ) ]
If [ $result = !!ERROR!! ]
Show Custom Dialog [ "An error occurred: " & PCMM_GetLastError( "Text" ) ]
End If
```

The PCMM_GetLastError(format) function gives you the option to display the error description or error number. Displaying the error number is more user friendly in international environments, where an English error description may not be desired. If the format parameter is set to "Number" such as PCMM_GetLastError("Number"), then an error number will be returned. If format parameter is empty such as PCMM_GetLastError or PCMM_GetLastError("Text"), then an English error description will be returned. The error numbers and their meanings can be found below.

Error Number	Error Text
0	Success
-1	Plug-in not registered or session expired
-3	Invalid # of Parameters
-4	Invalid Parameter value(s)
-10	Failed Registration
-10001	There are no folders in the current account.
-10002	There is no account by that name.
-10003	There are no accounts in your mail application.
-10004	There is no folder by that name.
-10005	No folder is currently selected.
-10006	There are no messages in the current folder.
-10007	No message is currently selected.
-10008	There is no message by that name.
-10009	Attachment wasn't found.
-10010	Attachment save error.
-10011	No account open.
-10012	Invalid path.
-10013	Index parameter exceeded attachment count.

III. Contact Us

Successful integration of a FileMaker plug-in requires the creation of integration scripts within your FileMaker solution. A working knowledge of FileMaker Pro, especially in the areas of scripting and calculations is necessary. If you need additional support for scripting, customization or setup (excluding registration) after reviewing the videos, documentation, FileMaker demo and sample scripts, then please contact us via the avenues listed below.

Phone: 760-510-1200

Email: support@productivecomputing.com

Forum: www.productivecomputing.com/forum

Please note assisting you with implementing this plug-in (excluding registration) is billable at our standard hourly rate. We bill on a time and materials basis billing only for the time in minutes it takes to assist you. We will be happy to create your integration scripts for you and can provide you with a free estimate if you fill out a Request For Quote (RFQ) at www.productivecomputing.com/rfq . We are ready to assist and look forward to hearing from you!