Productive Computing

Developer's Guide

Revised June 13, 2017

950 Boardwalk, Suite 205, San Marcos, CA 92078 • (760) 510-1200 • www.productivecomputing.com

Table of Contents

I. Introduction

Description:

The Outlook Manipulator plug-in is a powerful tool used to exchange data between FileMaker Pro® and Microsoft® Outlook. With this plug-in FileMaker Pro users are able to bidirectionally exchange data between FileMaker and Outlook. These operations are accomplished using FileMaker function calls from within FileMaker calculations. These calculations are generally determined from within FileMaker "SetField" or "If" script steps. This document describes the basic integration steps, features, and error handling. Please see the accompanying Functions Guide for a list of available plug-in functions and Outlook fields.

Product Version History:

http://www.productivecomputing.com/outlook-integration/version_history

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) Watch our video tutorials: http://www.productivecomputing.com/video
4) Review our FileMaker demo file
5) Familiarize yourself with Microsoft Outlook

II. Integration Steps

Accessing and using the plug-in involves the following steps.

1) Prerequisites For Plug-In Installation

32-bit vs 64-bit:

As of plug-in versions 6.0.2.2 of Outlook Manipulator and 4.0.2.2 of Exchange Manipulator are available in 32-bit or 64-bit mode. The plug-in bit version that you use depends upon your FileMaker Pro bit version and the bit version of Microsoft Outlook. All applications need to be running in the same bit version. Note: 32-bit applications and 32-bit plug-ins will work on a 64-bit operation system.

	Windows OS 32-bit FileMaker 32-bit	Windows OS 64-bit FileMaker 32-bit	Windows OS 64-bit FileMaker 64-bit
Microsoft Outlook or Exchange (32-bit)	Use 32-bit version of the plug-in	Use 32-bit version of the plug-in	Will not work. Requires Outlook/ Exchange 64-bit and 64-bit version of the plug-in
Microsoft Outlook or Exchange (64-bit)	Will not work. Requires Outlook/ Exchange 32-bit and 32-bit version of the plug-in	Will not work. Requires Outlook/ Exchange 32-bit and 32-bit version of the plug-in	Use 64-bit version of the plug-in

Installing the Microsoft Visual C++ 2013 Redistributable Package:

Included in the package is a download link.

Name of link is: "Download Microsoft Visual C++ 2013 Redistributable Package (x86)"

This link will direct you to download the Microsoft Visual C++ Redistributable Package (x86). Some systems do not have a Visual C++ 2013 Redistributable Package installed by default. However, certain programs may have added it to your machine during their installation process.

If the plug-in fails to be recognized by FileMaker after installation (i.e. does not show up in the Edit > Preferences > Plug-ins section), then please install the included redistributable package.

Machines running 64-bit versions of Windows need to install the 64-bit ("x64") version of the redistributable package, which is also available from Microsoft.

Please note: For older versions, use the 2008 redistributable package.

2) Installing the Plug-in with the Installer

We have introduced installers to make installation of our plug-ins even easier. These installers will not only install the FileMaker plug-in file, but will also install any third party software needed for the plug-in to function, the demo file, and additional resources you may need. We recommend using the installers to ensure that all components necessary for the plug-in to function are properly installed.

Once you download the Outlook Manipulator installation zip package, simply extract the package and open the resulting folder. Install the Outlook Manipulator with the following steps:

1) Run the "setup.exe" fil
2) If you are currently running FileMaker, please close FileMaker so that the plug-in will be installed correctly.
3) Accept the End User License Agreement ("EULA")
4) Select the location to install the plug-in*
5) Confirm the installation
6) If prompted by Windows user account control, allow the Installer to run
7) Your installation is complete!

*In order for FileMaker to properly recognize the plug-in, we suggest you do not change this default location. FileMaker plug-ins need to be installed in Extension folders recognized by the plug-in. By default, the plug-in will be installed to the base FileMaker/Extensions folder and will be available across multiple versions of FileMaker. However, if you wish to install the plug-in at a version-specific location like "FileMaker Pro Advanced/14.0/ Extensions", you may browse to the folder location to do so.

3) Installing the Plug-in Manually

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

FileMaker 12 or later:

1. Open the FileMaker demo file available in the plug-in bundle (www.productivecomputing.com).

2. Select the "Install" button.

For FileMaker 11 or earlier, follow the steps below to manually install the plug-in into the FileMaker Extensions folder. NOTE: This plug-in is verified compatible with FileMaker Pro 12 or above and may work with earlier versions.

1. Quit FileMaker Pro completely.

2. Locate the plug-in in your download which will be located in a folder called "Plug-in." On Windows the plug-in will have a ".fmx" extension.

3. Copy the actual plug-in and paste it to the Extensions folder which is inside the FileMaker program folder.

   On Windows this is normally located here: C:\Program Files\FileMaker\FileMaker X\Extensions

4. Start FileMaker Pro. Confirm that the plug-in has been successfully installed by navigating to "Preferences" in FileMaker, then select 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.

4) Troubleshooting Plug-in Installation

When installing the plug-in using the "Install Plug-in" script step, there are certain situations that may cause a 1550 or 1551 error to arise. If such a situation occurs, please refer to the troubleshooting steps involving the most common problems that may cause those errors.

1) Invalid Bitness of FileMaker

    a. In some cases, FileMaker Pro may be attempting to install a plug-in with a different bitness than the FileMaker Pro application. This is most common with Windows plug-ins. The general rule is that the plug-in and FileMaker Pro must be the same bitness.

    b. To resolve this, ensure that the container field holding the plug-in contains the correct bitness of the plug-in. You can verify the plug-in's bitness by checking the file extension: if the extension is .fmx, the plug-in is a 32-bit plug-in; if the extension is .fmx64, the plug-in is a 64-bit plug-in. You can verify the bitness of FileMaker Pro itself by viewing the "About FileMaker Pro" menu option in the Help menu, and clicking the "Info" button to see more information; bitness is found under "Architecture".

2) Missing Dependencies

    a. Every plug-in has dependencies, which are system files present in the machine's operating system that the plug-in requires in order to function. If a plug-in is "installed" into an Extensions folder, but the plug-in does not load or is not visible in the Preferences > Plug-ins panel in FileMaker Pro's preferences, it's likely that there are files missing.

    b. To ensure that the appropriate dependencies are installed, please verify that the Visual Studio 2013 C++ Redistributable Package is installed. This can be located by opening Control Panel and checking the Installed Programs list (usually found under "Add/Remove Programs"). Older plug-ins may require the Visual C++ 2008 redistributable package, instead of the 2013 version.

    c. Some plug-ins also have a .NET Framework component that is also required. All such plug-ins of ours will require the .NET Framework 3.5, which can be downloaded from the following link:

    https://www.microsoft.com/en-us/download/details.aspx?id=21

3) Duplicate Plug-in Files

    a. When installing plug-ins, it is possible to have the plug-in located in different folders that are considered "valid" when FileMaker Pro attempts to load plug-ins for use. There is a possibility that having multiple versions of the same plug-in in place in these folders could cause FileMaker Pro to fail to load a newly-installed plug-in during the installation process.

    b. To resolve this, navigate to the different folders listed in the earlier installation steps and ensure that the plug-in is not present there by deleting the plug-in file(s). Once complete, restart FileMaker and attempt the installation again. If you installed the plug-in using a plug-in installer file, if on Windows, run the installer again and choose the "Uninstall" option, or if on Mac, run the "uninstall.tool" file to uninstall the plug-in.

If the three troubleshooting steps above do not resolve the issue, please feel free to reach out to our support team for further assistance.

5) Registering the Plug-in

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

1. 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.

2. If you are registering the plug-in in Demo mode, then simply click the "Register" button and do not change any of the fields. Your plug-in should now be running in "DEMO" mode. The mode is always noted on the Setup tab of the FileMaker demo.

3. If you are registering a licensed copy, then simply enter your license number in the "LicenseID" field and select the "Register" button. Ensure you have removed 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 always noted on the Setup tab of the FileMaker demo, or by calling the PCEM_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. 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-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 Windows this certificate is in the form of a ".pci" file.

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

6) FileMaker 16 Plug-in Script Steps

Newly introduced in FileMaker Pro 16, all plug-ins have been updated to allow a developer to specify plug-in functions as script steps instead of as calculation results. The plug-in script steps function identically to calling a plug-in within a calculation dialog.

In this example, we use the FM Books Connector plug-in's script steps to demonstrate the difference. The same scripting differences would be found for any of the Productive Computing plug-in product lines.

For an example of using plug-in script steps, compare two versions of the same script from the FM Books Connector demo file: Pull Customer__Existing Session.

Script 1 - Pull Customer__Existing Session with calculation ("traditional") plug-in scripting:

```
Set Error Capture [On]
Allow User Abort [Off]
# It is assumed the session is already opened from the previous script calling this
script.
# Query customers in QB (Request)

Set Variable [$$Result; Value: PCQB_RqNew( "CustomerQuery" ; "" )]
Set Variable [$$Result; Value: PCQB_RqAddFieldWithValue( "ListID" ;
Main::gCust_ListID )]
If [0 <> PCQB_RqExecute]
      Exit Script [Text Result:PCQB_SGetStatus]
End If

# Pull customer info into FileMaker (Response)
Set Variable [$$Result; Value: PCQB_RsOpenFirstRecord]
Set Field [main_CUST__Customers::ListID; PCQB_RsGetFirstFieldValue( "ListID" )]
Set Field [main_CUST__Customers::FullName; PCQB_RsGetFirstFieldValue( "FullName" )]
Set Field [main_CUST__Customers::First Name;
PCQB_RsGetFirstFieldValue( "FirstName" )]
Set Field [main_CUST__Customers::Last Name; PCQB_RsGetFirstFieldValue( "LastName" )]
Set Field [main_CUST__Customers::Company; PCQB_RsGetFirstFieldValue( "CompanyName" )]
Set Field [main_CUST__Customers::Bill_Address 1;
PCQB_RsGetFirstFieldValue( "BillAddress::Addr1" )]
Set Field [main_CUST__Customers::Bill_Address 2;
PCQB_RsGetFirstFieldValue( "BillAddress::Addr2" )]
Set Field [main_CUST__Customers::Bill_Address 3;
PCQB_RsGetFirstFieldValue( "BillAddress::Addr3" )]
Set Field [main_CUST__Customers::Bill_Address 4;
PCQB_RsGetFirstFieldValue( "BillAddress::Addr4" )]
Set Field [main_CUST__Customers::Bill_City;
PCQB_RsGetFirstFieldValue( "BillAddress::City" )]
Set Field [main_CUST__Customers::Bill_State;
PCQB_RsGetFirstFieldValue( "BillAddress::State" )]
Set Field [main_CUST__Customers::Bill_Postal Code;
PCQB_RsGetFirstFieldValue( "BillAddress::PostalCode" )]
Set Field [main_CUST__Customers::Phone; PCQB_RsGetFirstFieldValue( "Phone" )]
Set Field [main_CUST__Customers::Email; PCQB_RsGetFirstFieldValue( "Email" )]

Exit Script [Text Result:0]
```

Script 2 – Pull Customer__Existing Session with plug-in script steps:

```
Set Error Capture [On]
Allow User Abort [Off]
# It is assumed the session is already opened from the previous script calling this
script.
# Query customers in QB (Request)

PCQB_RqNew [Select; Results:$$Result; Request Type:"CustomerQuery"]
PCQB_RqAddFieldWithValue [Select; Results:$$Result; QB Field Name:"ListID"; Field
Value:Main::gCust_ListID]
PCQB_RqExecute [Select; Results:$$Result]
If [$$Result <> 0]
      Exit Script [Text Result:PCQB_SGetStatus]
End If

# Pull customer info into FileMaker (Response)
PCQB_RsOpenFirstRecord [Select; Results:$$Result]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::ListID; Field
Name:"ListID"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::FullName;
FieldName:"FullName"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::First Name; Field
Name:" FirstName"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Last Name; Field
Name:" LastName"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Company; Field
Name:" CompanyName"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_Address 1;
Field Name:" BillAddress::Addr1"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_Address 2;
Field Name:" BillAddress::Addr2"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_Address 3;
Field Name:" BillAddress::Addr3"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_Address 4;
Field Name:" BillAddress::Addr4"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_City; Field
Name:" BillAddress::City"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_State; Field
Name:" BillAddress::State"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Bill_Postal Code;
Field Name:" BillAddress::PostalCode"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Phone; Field
Name:"Phone"]
PCQB_RsGetFirstFieldValue [Select; Results:main_CUST__Customers::Email; Field
Name:"Email"]

Exit Script [Text Result:0]
```

Using script steps instead of the more traditional methods can make scripting within a solution more direct, as well as help with data entry validation. Some functions accept calculation-style input, while others accept a Boolean "true" or "false" option, and others employ a drop-down list for the developer to choose an option from. As stated earlier, the functionality of the plug-in script step is identical to its functionality as a calculation function; PCQB_RsOpenFirstRecord as a script step will still open the first record in the response, and store the value in the $$Result global variable (as seen in Script 2), just the same as the Set Variable script step calls PCQB_RsOpenFirstRecord (which opens the first response record) and stores the result in the $$Result variable.

For all Productive Computing, Inc., plug-ins that provide plug-in script step functionality, calculation functions will still be provided for use in development. This is to ensure that scripts already integrated with any of our plug-ins will still be viable and functional, and the developer now has the option to utilize the plug-in script steps at their discretion.

7) Talking to Outlook

Talking to Outlook typically requires that you follow these steps.

A. Authenticate to Outlook

When communicating with Outlook you must first specify what Outlook profile will be used. This is accomplished by authenticating to Outlook by calling the PCEM_Authenticate (ProfileName ; ProfilePassword) function. Authenticate must be called once per FileMaker session. If Outlook is open when this function is called, then FileMaker will simply authenticate to the currently opened Outlook profile.

If Outlook is not open when this function is called, then FileMaker will open Outlook in the background and authenticate with the default Outlook profile or it will prompt the user to select one. This depends on your machine settings found in the Windows Control Panel under Mail. Most users do not have multiple Outlook profiles set up on their machine and will benefit from leaving the two parameters (ProfileName ; ProfilePassword) blank. If you do have multiple Outlook profiles set up on your machine, then you can pass the ProfileName and ProfilePassword to access the desired Outlook profile. Please note that not all profiles have passwords associated with them.

If you desire to authenticate with a different Outlook profile, then you must first close FileMaker AND Outlook. This is because FileMaker will hold onto the authenticated Outlook profile until both FileMaker and Outlook processes are terminated.

B. Select a Root Folder

The next step is to specify what "root" folder or mailbox will be used. During the authenticate process the root folder is typically automatically set to the user's default folder. The default folder is typically the first folder named in the Outlook Folder List such as "Personal Folders" or "Mailbox - User Name." Please note that in Outlook 2003 and Outlook 2007, the mailbox name is in the format of "Mailbox - Your Name," but Outlook 2010 and newer now uses your email address as the mailbox name. Please account for this change as needed if your users are using various Outlook versions.

Specify the the root folder to work with by calling PCEM_OpenRootFolder(RootFolderName) and passing the name of the desired root folder to be opened. Once the Outlook root folder has been selected then any subfolder of the root folder can be accessed such as Mail folders, Calendar folders, Contact Folders, etc. Please note that since Outlook refers to user mailboxes and modules such as Mail, Contacts, Calendar, Tasks, Notes and Journal modules as folders, then so does our documentation and plug-in.

For example: PCEM_OpenRootFolder opens what is commonly known as a "mailbox" and PCEM_OpenFolder opens what is commonly known as a "module" such as the Calendar module. Outlook refers to both the mailboxes and modules as folders. Everything in Outlook is a folder even if it has the appearance of a mailbox or Calendar, Tasks, and any other various icons that Outlook uses.

C. Select an Outlook Folder

Now that we have authenticated to Outlook and specified a "root" folder, the next step is to navigate to the desired folder. In order to access any records in Outlook the user must first open the desired folder that contains the desired records. This is accomplished by calling PCEM_OpenFolder(FolderPath) and specifying the desired folder by passing the FolderPath parameter to the OpenFolder function. When the FolderPath is prefixed with a '/' then the FolderPath is relative to the current root folder. In other words the '/' indicates a subfolder.

For example, the following call opens the folder named "NewItems" located in the "Inbox" Folder:
- PCEM_OpenFolder("/Inbox/NewItems")

Another example shows the following call opens the folder named "Holiday" located in the "Calendar" Folder:
- PCEM_OpenFolder("/Calendar/Holiday")

When the FolderPath is NOT prefixed with a '/', then the FolderPath is relative to the currently opened folder. For example, you could also open the Holiday subfolder mentioned above by following two calls:
- PCEM_OpenFolder("/Calendar")
- PCEM_OpenFolder("Holiday")

It is up to the developer to "know'" their location within the Outlook Tree and to know what types of records are contained in the currently opened folder. Outlook has various items types which are: Calendar Items, Contact Items, Mail and Post Items, Note Items, Task Items, and Journal Items.

Note: If in the course of setting field data, saving an e-mail record, or sending an e-mail you receive a dialog in Outlook asking to allow a program to send on behalf of an email account, ensure that the current machine has an anti-virus program installed and updated to the latest definitions. This is a normal security response from Outlook when it detects that there is no anti-virus software installed on the system, or that software is not up to date.

D. Create/Open a Record

After navigating to the desired folder, you can now create new records to be placed in the folder or access the records within that folder.

Create a New Record:

There is one method for creating a new record which is by calling the PCEM_NewRecord(optModule) function. Anytime the PCEM_NewRecord function is called the plug-in creates a new record and holds it in memory. The record does not actually exist in Outlook until the record is properly saved, which is addressed later in this document. Note that this function closes the currently opened record without saving any changes. It is up to you to properly save the record before beginning another command.

Open an Existing Record:

There are two methods for accessing existing Outlook records which are directly or indirectly.

Directly:

If the user knows the OutlookID of the desired record, then they can open that record directly by calling the PCEM_OpenRecord(OutlookID) function and passing the OutlookID to the function.

Note that this function closes the currently opened record without saving any changes. It is up to you to properly save the record before beginning another command. Saving records is discussed later in this document.

Indirectly:
The functions PCEM_GetFirstRecord and PCEM_GetNextRecord offer the user the ability to iterate through the records contained in the currently opened folder.

For example:

```
/* Opens the first Record in the Current Folder */
SetField( someField ; PCEM_GetFirstRecord )
Loop
Exit Loop If[ someField = 'End' or someField < 0]
/* Access - Modify the fields */
…
/* Opens the next record */
New Record/Request
SetField( someField ; PCEM_GetNextRecord )
End Loop
…
```

Note that these functions make either the First or Next record the currently opened record available. They also close the currently opened record without saving any changes. It is up to you to properly save the record before beginning another command. Saving records is discussed later in this document.

E. Manipulate the Fields

Once a record is created or opened then you can access the fields contained within that record. You can either set or retrieve the contents of the available Outlook fields for that record using PCEM_SetFieldData or PCEM_GetFieldData. A list of all available Outlook fields can be found in the "Functions Guide."

The PCEM_SetFieldData(FieldName ; FieldValue) function is used to populate fields in the new or currently opened Outlook record. See the "Functions Guide" for list of all available FieldNames.

The PCEM_GetFieldData(FieldName) function is used to extract data from the currently opened Outlook record. See the "Functions Guide" for list of all available FieldNames.

F. Save Record or Send Email

After creating or modifying records use the PCEM_SaveRecord(optionalParam) function to save the record to the current Outlook folder and return the Outlook ID.

If the current record is a new email message (Mail item) then the PCEM_SaveRecord function will cause Outlook to send the email message. Sending an email using SaveRecord is known as the "long method" and requires the following functions (PCEM_NewRecord, PCEM_SetFieldData, PCEM_Add Attachment, PCEM_SaveRecord). You can also send email using the "short method" in a single script step using PCEM_SendMail(To ; Cc ; Bcc ; Subject ; Body ; Attachments ; optUseHTML). The "short method" is easier to program. All parameters except Subject, Body, and OptUseHTML can contain multiple comma separated values.
The only time you need to create an email using the "long method" is if you need to obtain the Outlook ID for the email being sent, set additional fields such as SendOnBehalfOf, From, Categories, Flag Icon, etc. or if you need to set the optionalParam of the PCEM_SaveRecord function. The optionalParam can be set to "Appointment" or "DontSend." The literal string "Appointment" sends a meeting request to attendees if you have a calendar/appointment type email. The literal string "DontSend" delays the sending of the email if you desire to open the email in Outlook before the email is sent. Please reference PCEM_SaveRecord function in the "Functions Guide" for further details.

In the examples below an email will be sent with three attachments using the various methods. Please refer to the Functions Guide and our FileMaker demo file for further details and live examples.

Example 1: Sending an email using the "short method" by hardcoding parameter values:
- PCEM_SendMail("joe@somecompany.com"" ; "" ; "" ; "New Release" ; "A new product was just released allowing for ..... please contact us" ; "C:\Pricing.pdf, C:\Overview.pdf, F:\SpecSheet.pdf" ; "N")

Example 2: Sending an email using the "short method" by using field names:
- PCEM_SendMail(Main::gEmail To ; Main::gEmail Cc ; Main::gEmail Bcc ; Main::gEmail Subject ; FullPath1 & "," & FullPath2 & "," & FullPath3 ; Main::gUseHTML)
The FullPath1, FullPath2, and FullPath3 represent the field that stores the full file path.

Example 3: Sending an email using the "long method" by hardcoding parameter values:
- PCEM_OpenRootFolder("Mailbox - User")
- PCEM_OpenFolder("/Inbox")
- PCEM_NewRecord("Mail")
- PCEM_SetFieldData("To" ; "joe@somecompany.com")
- PCEM_SetFieldData("From" ; "me@mycompany.com")
- PCEM_SetFieldData("Subject" ; "New Release")
- PCEM_SetFieldData("Body" ; "A new product was just released allowing for ..... please contact us")
- PCEM_SetFieldData("Categories" ; work)
- PCEM_AddAttachment("C:\Pricing.pdf" ; "Pricing")
- PCEM_AddAttachment("C:\Overview.pdf" ; "Overview")
- PCEM_AddAttachment("F:\SpecSheet.pdf" ; "SpecSheet")
- PCEM_SaveRecord

Example 4: Sending an email using the "long method" by using field names:

- PCEM_OpenRootFolder(Main::gFolder Root)
- PCEM_OpenFolder(Main::gFolder Mail)
- PCEM_NewRecord("Mail")
- Set Variable [$Result; Value:PCEM_SetFieldData("To" ; Mail::To)]
- Set Variable [$Result; Value:PCEM_SetFieldData("From" ; Mail::From)]
- Set Variable [$Result; Value:PCEM_SetFieldData("Subject" ; Mail::Subject)]
- Set Variable [$Result; Value:PCEM_SetFieldData("Body" ; Mail::Body)]
- Set Variable [$Result; Value:PCEM_SetFieldData("Categories" ; Mail::Categories)]
- PCEM_AddAttachment(FullPath1 ; FileName1)
- PCEM_AddAttachment(FullPath2 ; FileName2)
- PCEM_AddAttachment(FullPath3 ; FileName3)
- PCEM_SaveRecord

G. Responding to a Meeting Request

The Outlook Manipulator has the ability to send a meeting response to the organizer of a meeting. To accept a meeting invitation, perform the following steps:

1. Open the Meeting event using PCEM_OpenRecord

2. Set the field "Meeting Response" with the value "Accepted"

3. Save the record with a call to PCEM_SaveRecord("ApptResponse")

The use of the parameter "ApptResponse" will create a meeting response message in Outlook for the meeting that is opened, populate it with the required meeting response information based off of the setting of the "Meeting Response" field, and then submit the message through Outlook to be sent to the organizer of the meeting with the desired response.

Note: At this time, the plug-in can only send a meeting acceptance message. It cannot decline a meeting, tentatively accept a meeting, nor propose a new meeting time to the organizer. This has been noted for further enhancements for a later version of the plug-in.

Example script demonstrating the acceptance of a meeting invitation:

```
# Open the default Calendar
Set Variable [ $result ; Value:PCEM_OpenDefaultFolder( "Calendar" ) ]
# Error handling…
#
# Open the record
Set Variable [ $result ; Value:PCEM_OpenRecord( $$OutlookID ) ]
# Error handling…
#
# Set the Meeting Response field with the value of "Accepted"
Set Variable [ $result ; Value:PCEM_SetFieldData( "Meeting Response" ; "Accepted" ) ]
#
# Save the record and send the response
Set Variable [ $OutlookID ; Value:PCEM_SaveRecord( "ApptResponse" ) ]
# Error Handling…
```

8) Setting the “From” and “Send On Behalf Of” Fields

If you would like to send an email from or on behalf of another user then please read this entire section. If you simply desire to send an email from your default Outlook account, then you can skip over this section.

In order to send an email from or on behalf of another user you will need to use the long method to send mail and set the “From” or “Send On Behalf Of” field. Before you can successfully set these fields and send email from or on behalf of another user you will need to ensure that you have proper access rights in either Exchange or Outlook. This is a security designed by Microsoft that protects from having an unauthorized user send mail from or on your behalf.

Send As Set Up:

Sending email from another user allows one user to send an email as though it came directly from another user. The recipient will not be notified that the email was composed by someone other than the stated sender. "Send As" permission can only be granted by the Exchange System Administrator. There are various ways to grant "Send As" from a different Exchange account and this is just one of them. Please note that this is a caveat of Microsoft Exchange. If Microsoft Exchange is not configured to grant user rights to properly "Send As" another address, then neither can our plug-in. The following instructions were taken from Exchange 2003.

1) Log onto the server running Exchange.
2) Open Active Directory Users and Computers.
3) Ensure that the "Advanced Feature is checked" under the "View" menu.
4) Locate the user's account that you want to be able to send as. Open the account properties.
5) Select the "Security" tab.
6) Select "Add" under "Group or user names" and add the user (users or group) that is to be granted permission to send-as this account.
7) For each account added ensure that the account under "Group or user names" and in the "Permissions for ..." window grant the account "Send As" permission.
8) Select "OK" to close the account properties.
9) Log into Outlook and display the 'From" field.
10) Attempt to simply send an email out of Outlook as the specified user you just granted.
11) If you receive these errors, then this typically indicates that the security was not setup properly or has not taken effect.
"Your message did not reach some or all of the intended recipients.
The following recipient(s) could not be reached: ...
You do not have permission to send to this recipient. For assistance, contact your system administrator"

Or the following error (if you are not in cached mode):
"You do not have sufficient permission to perform this operation on this object. See your system administrator.”

Ensure that the security has taken effect by re-starting the Microsoft Exchange System Attendant Service on the server. The Microsoft Exchange System Attendant service functions to proxy Active Directory requests and to regulate internal Exchange Server functions.

12) Then let's also re-start Outlook on the client machine. You may also need to restart the client machine, but typically you can just restart Outlook.
13) Log into Outlook again and attempt to send an email as the specified user. Once you can send an email using the "Send As" feature in Outlook, then you are ready to start using our plug-in.

You may want to consult Microsoft or Exchange experts for how to grant a user access to "Send As" another user.

Here are a few helpful links:
http://support.microsoft.com/kb/895949
http://support.microsoft.com/default.aspx?scid=kb;EN-US;907434
http://support.microsoft.com/search/default.aspx?qid=219454&query=send+as+another+user&catalog=LCID%3D1033&mode=r

Please also note that after the emails have been sent using the "Send As" feature, then the sent emails will appear in the Sent Items folder for the default mailbox in Outlook. It is up to the developer to develop a process to move these messages to the appropriate users' Sent Items folder if necessary. This is a functionality of Microsoft Exchange and Outlook. Since Microsoft stores all sent email messages in the Outlook user's Sent Items folder, then so does our plug-in. If this feature becomes available by Microsoft, then we can add the functionality into our plug-in. Please feel free to contact Microsoft and request this feature.

Send On Behalf Of Set Up:

"Send on Behalf of" allows the recipient to be notified both who the author was and on whose behalf the email was sent. "Send on Behalf of" can be granted by the Outlook user and is relatively simple to set up.

1) Start Outlook.
2) Select Tools, then Delegates from the menus at the top.
3) Select the Add button and add the desired delegate. A delegate can send items on your behalf.
4) Select the desired delegate permissions.
5) Select Apply and then OK.

Please ensure that you can "Send As" or "Send on Behalf Of" directly from Outlook before attempting to set the "From" and "Send On Behalf Of" fields using our plug-in.

9) Moving or Deleting a Record

A. Move a Record

After a record has been opened, call the PCEM_MoveRecord(FolderPath) to move the record from the current folder to another folder. After calling the MoveRecord function, the folder identified by the FolderPath becomes the currently opened folder. If you are going to move multiple items, then a call to OpenRootFolder and/or OpenFolder must be called to return to the previously opened folder.

The following caveats must be recognized:
- When you move records within the same mailbox, the leading "/" character for the destination folder should be included. For example: /folderpath or /Inbox/Test
- When you move records to a different mailbox such as Public Folders or another user's mailbox, the leading "/" character for the destination folder should be omitted. For example: mailbox/folderpath or Public Folders/All Public Folders/Test

B. Delete a Record

In order to delete records you can use the PCEM_DeleteRecord(OutlookID ; Permanent), PCEM_DeleteCurrentRecord(bPermanent) or PCEM_DeleteAllRecords(ExcludePrivate ; ExcludeDistList) functions. The DeleteRecord function will delete a record in the currently opened folder. By using the optional Permanent parameter the record can be permanently deleted or sent to the "Deleted Items" folder in Outlook for archiving. The DeleteCurrentRecord function will delete the currently opened record and maintain the index of items. The DeleteAllRecords command will delete all records from a given module. This command should be used with extreme caution as it will remove ALL items permanently and will not save a copy of the items in the "Deleted Items" folder in Outlook. Please reference the "Functions Guide" for additional information about these functions.

10) Working with Custom Fields

Outlook has custom fields or "user-defined" fields. The PCEM_SetCustomFieldData, PCEM_GetCustomFieldData, and PCEM_CreateCustomField functions are used when working with Outlook custom fields. Let's explore some sample uses of these three functions.

PCEM_SetCustomFieldData(FieldName ; Data ; optType) sets the value of a custom field in a new record or an opened record. For example, if you wanted to create a new contact record with a custom field and push this record from FileMaker to Outlook, then you would follow these steps:

```
...
PCEM_NewRecord( "Contacts" )
PCEM_SetFieldData( "First Name" ; "Brett" )
PCEM_SetFieldData( "Last Name" ; "Wilcox" )
PCEM_SetFieldData( "Company" ; "ABC Company" )
PCEM_SetCustomFieldData( "Pets" ; "Kujo" ; "Text" )
PCEM_SaveRecord
...
```

This PCEM_SetCustomFieldData step will actually create and set the custom field in Outlook in a single step.

PCEM_GetCustomFieldData(FieldName) function gets the value of the custom field from Outlook into FileMaker. If you wanted to pull a record with a custom field from Outlook into FileMaker, then you would use these steps:

```
...
PCEM_GetFirstRecord
PCEM_GetFieldData( "First Name" )
PCEM_GetFieldData( "Last Name" )
PCEM_GetFieldData( "Company" )
PCEM_GetCustomFieldData( "Pets" )
...
```

PCEM_CreateCustomField(Name ; Type) function creates a custom field and adds the new custom field to the currently opened record. You must specify the Name of the custom field and the Type. This function may be useful if you just need to create the actual custom field name in Outlook.

```
...
PCEM_OpenRecord( Contacts::Outlook ID )
PCEM_CreateCustomField( "Pets" ; "Text" )
PCEM_SaveRecord
...
```

11) Filtering Records

The plug-in uses the "Restrict Method" in order to filter records. The PCEM_FilterByFilterString and PCEM_FilterByLastModified functions are available to restrict records available to the plug-in that meet the specified criteria.

The **PCEM_FilterByFilterString**(FilterString) function is designed to restrict records that meet the criteria specified in the FilterString parameter.

The **PCEM_FilterByLastModified**(Timestamp) function is designed to restrict records that are in the current folder and have a modified timestamp later than that passed into the function.

Filter strings should be constructed in a very specific manner known as the "Restrict Method." The "Restrict Method" is explained in detail on the following Microsoft websites:
http://msdn.microsoft.com/en-us/library/bb220369.aspx
http://msdn2.microsoft.com/en-us/library/aa210275(office.11).aspx

The field names to be used in a filter string are not the same field names that are used by the plug-in. The field names available in a filter string are those that are available to VBA (VisualBasic for Applications) and not necessarily those available to the plug-in. The available field names are outlined in the "Restrict Method" documentation provided by Microsoft.

We constructed a filter string for the PCEM_FilterByLastModified function, because this function works on one Outlook field (the LastModificationTime, which is available to all Outlook objects), and we know how to convert a FileMaker timestamp to something that the "Restrict Method" accepts. We chose this field as it seemed to be the most common filter that anyone would apply to a folder. To do the same for the rest of the fields in each of the Outlook Objects would be impossible. That is why the choice was made to simply pass the user supplied filter string directly to the "Restrict Method" from the PCEM_FilterByFilterString function.

Because we pass filter string directly to the "Restrict Method", the filter string MUST be a string that the "Restrict Method" understands. It must contain field names that the method will work with and the values must be formatted in the way the "Restrict Method" requires.

For example, the "To" field works as it is a valid MailItem Property for VBA clients, but the "From 1" field is not a valid field for VBA clients and can cause a '?' to be returned to FileMaker.

Review the "Reference" section of this link http://msdn.microsoft.com/en-us/library/aa221877.aspx to see which properties are available to which Outlook objects and which properties are NOT supported by the "Restrict Method."

12) Error Handling

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 call PCEM_GetLastError function for a detailed description of the exact error.

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: " & PCEM_GetLastError ]
End If
```

The PCEM_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 PCEM_GetLastError("Number"), then an error number will be returned. If format parameter is empty such as PCEM_GetLastError or PCEM_GetLastError(""), then an English error description will be returned.

Please find a list of return codes and descriptions below for your reference.

Code	Description
0	SUCCESS
-1	Plug-in not registered
-2	Registration failed
-3	Invalid Number of Parameters
-4	Invalid Parameter value(s)
-10	Expired Registration
-200	Library failed to load
-300	The specified file cannot be found
-301	The specified file cannot be found
-302	The specified folder cannot be accessed
-400	System Error
-1000	Outlook Logon Failed
-1001	Outlook not started or Authenticate not called
-1002	This Version of Outlook is not supported
-1003	Cannot Find Root Folder
-1011	Missing Field Name Parameter
-1012	Save Path Missing
-1013	Missing Path to Attachment
-1015	Folder Not Found
-1016	Invalid Path or File Name
-1017	Malformed or invalid root folder name
-1018	Root folder not selected

Code	Description
-1019	Custom Field does not exist
-1020	Save called without calling New Record first
-1021	Invalid Module Name called
-1022	Invalid field name passed
-1023	Item operation called without opening/creating a record
-1024	Invalid Outlook ID
-1025	Unable to Save record
-1026	Must navigate to a folder before operation
-1027	Outlook Entry ID not found
-1028	Unable to find records
-1029	Unable to locate destination folder
-1030	GetFirstRootFolder not called
-1031	Unable to access root folders
-1032	Unable to access folder
-1033	Folder not created or already exists
-1034	Invalid ExcludeDistList argument
-1035	Invalid ExcludePrivate argument
-1036	Item type does not match destination folder
-1038	Unable to locate record
-1040	Invalid attachment number
-1041	Item is not supported in the version of the plug-in
-1087	Field is read-only
-1088	No attachments found
-1089	Invalid value for field
-1090	Item does not support custom fields
-1091	Unable to retrieve or invalid value for the custom field
-1092	Unable to access or invalid custom field type
-1093	User properties unavailable for item
-1094	Invalid type for custom field
-1095	MAPI Subsystem error
-1096	Invalid Parameter
-1097	Function Unavailable
-1098	Error accessing shared folder
-1099	Unable to restrict items
-1100	Invalid Folder Type
-1200	Invalid Folder Name

13) Known Issues

- **Outlook 2013 Task Handling:**

  In Microsoft Outlook 2013, Microsoft has changed the way task creation is handled by third party applications (plug-ins for Outlook/Exchange). This known issue is currently under investigation and has been partially fixed in the Outlook 2016 release.

  Outlook 2013: If pushing a task record from FileMaker to Outlook 2013, and the task has a set reminder time, a dialog message will appear in Outlook indicating that the reminder for the task will not appear because the item is in a folder that doesn't support reminders. The task will still be created and set with the reminder in the background; however, the reminder is set as "Inactive" and the reminder will not actually trigger. This issue only affects Outlook/Exchange 2013. Adding tasks without reminders functions normally. All other versions of Outlook work as expected either with or without reminders.

  Outlook 2016: If pushing a task record from FileMaker into Outlook 2016, and the task has a set reminder time, a dialog message will appear in Outlook indicating that the reminder for the task will not appear because the item is in a folder that doesn't support reminders. If the user indicates "yes", the task will be created and the reminder will trigger a popup reminder notification. If the user indicates "no", an error will be returned and the task will not be pushed. This issue is still under investigation.

- **Outlook Authentication – "Unknown MAPI Error" / "Interface Not Registered":**

  In certain cases, Outlook Manipulator may return an error "80040155" ("Unknown MAPI Error" or "Interface Not Registered") result when calling PCEM_Authenticate(username ; password). This issue is most prominent in Outlook 2013, and specifically from using a "Click-To-Run" installer for Outlook 2013 or Outlook 365.

  The following fix has been provided for the plug-in (version 6.0.1.0 or later):

  - Ensure that FileMaker and Outlook are both closed.
  - Right-click on the FileMaker application and select "Run as Administrator." (In Windows 7 or 8, you may need to right-click on FileMaker within the first pop-up box to see "Run as Administrator")
  - Allow FileMaker to run with the administrator elevated privileges.
  - Once FileMaker has completed loading, close and reopen FileMaker as normal.
  - Optionally, you may reopen Outlook if your solution requires Outlook to be open.
  - This fix will make a modification to the Windows Registry to ensure that the communication interface the plug-in uses to talk to Outlook is properly set up. The changes will only work if FileMaker runs in "administrator" mode. Once FileMaker has been run with these elevated permissions for the first time, the fix will be set and any further calls to PCEM_Authenticate, whether FileMaker is running in standard or administrator mode, will have the proper communication interface set up for it.

- **FileMaker 13:**

  When performing operations in FileMaker 13, some unexpected behavior in regards to dialog boxes may occur. The following is a list of observed behavior that occurs only in FileMaker 13:

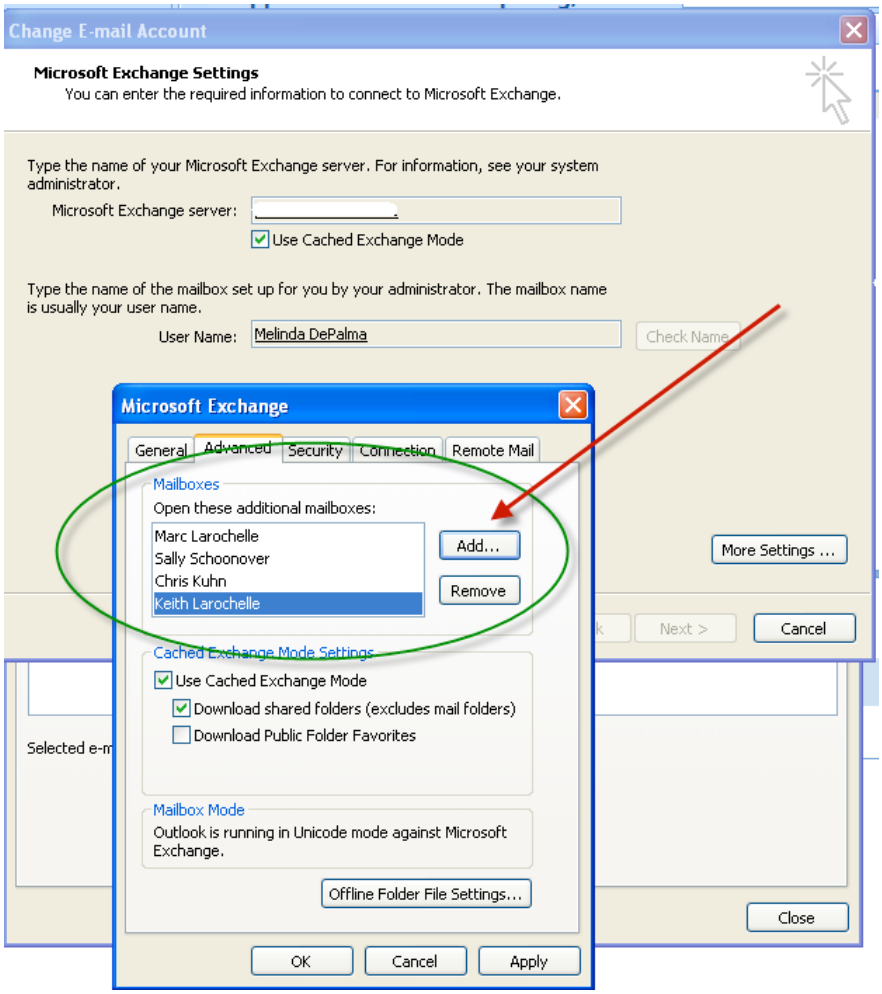
  - When using "PCEM_Display" after opening a record and passing a parameter value of "1" or blank, the record will appear in an Outlook modal window, as expected. After a short period of time, FileMaker will display a "Server Busy" status window that will persist until the Outlook modal window is closed. This behavior does not occur if PCEM_Display is called with a parameter value of "0".
  - If a dialog window appears in Outlook as a result of a function call from FileMaker, after a short period of time, a "Server Busy" status window will appear and will persist until the dialog window in Outlook is closed.

- **Outlook 2013 Release:**
  - Journal has been disabled as of the Outlook 2013 release. Please see link for other discontinued features and changes in the Outlook 2013 release: https://support.office.com/en-us/article/Discontinued-features-and-changes-in-Outlook-2013-6fad16ab-b50b-4900-81b9-249c71f3027b

III. Configure Multiple Mailbox Access

The Outlook Manipulator plug-in can access all mailboxes in Outlook and/or all folders for any Exchange user including public folders. Actual installation on the Exchange server is not required as the plug-in talks directly to Outlook. We recommend that you do not install the plug-in on the same machine running Exchange Server. The plug-in is installed on a single client or server machine, which has Outlook and FileMaker Pro installed. An "admin" profile is then created in Outlook. Within this "admin" Outlook profile, you open up all desired mailboxes as shown in the screenshot below.

NOTE: When using Office 2016, Exchange accounts need to be added using the Auto Discovery feature.



If the Outlook profile has access to all the Mailboxes in an organization, then the current FileMaker session will also. This plug-in is generally used in organizations where a dedicated machine such as a "robot" or "dummy" machine will be constantly running to update FileMaker and Exchange records. Native FileMaker OnTimer script triggers or third-party products such as ScriptFire or Activator can be used to periodically run scripts and update data through the plug-in. The plug-in installed on the dummy or robot machine will remove the need to have it installed on each and every client machine that will need to communicate with Outlook. Developers should be aware of the pros and cons for implementing the plug-in on multiple client machines vs. a single robot machine and design their implementation accordingly.

IV. Tips

- This "Developer's Guide" is designed to give you a basic understanding of how Outlook and FileMaker "talk" to each other. There are various other functions available by the plug-in depending on your needs. Please reference the "Functions Guide" for a detailed list of all available functions.

- The Outlook ID is a unique identifier created by Outlook. This ID is unique across all folders (Mail, Contacts, Calendar, Tasks, Notes, and Journal) and is unique to every Outlook user. In other words, you can store the Outlook ID to identify records in a multi-user environment. This may make programming easier if you are creating a data merge/sync process in a multi-user environment. Remember the Outlook ID is often longer than 120 characters.

- The plug-in cannot edit or maintain the list of categories. This is done exclusively in Outlook. If you add a brand new category in Outlook, you must force the Authenticate call before a record can be assigned to that category created in Outlook.

V. 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!