



Productive
Computing



Exchange
Manipulator SE

Developer's Guide

Table of Contents

I. Introduction	3
II. Integration Steps	4
1) Prerequisites for Plug-in Installation	4
2) Installing the Plug-in with the Installer	5
3) Installing the Plug-in Manually	5
4) Troubleshooting Plug-in Installation.....	6
5) Registering the Plug-in	7
6) FileMaker 16 Plug-in Script Steps	8
7) Server-Side Deployment.....	10
A. Installation.....	10
B. Deployment Requirements	11
C. Registration	11
D. Recommended Script Structure.....	12
E. Server-Side Scheduled Scripts.....	15
F. WebDirect Script	15
8) Talking to Exchange.....	16
A. Authenticate to Exchange.....	16
B. Session Management	17
C. Select a Root Folder.....	18
D. Select a Folder	18
E. Create/Open a Record	19
F. Manipulate the Fields.....	20
G. Save Record or Send Email.....	20
H. Responding to a Meeting Request	22
9) Setting the "From" and "Send On Behalf Of" Fields.....	24
10) Moving or Deleting a Record	26
A. Move a Record	26
B. Delete a Record.....	26
11) Working with Custom Fields.....	27
12) Filtering Records.....	28
13) Error Handling	29
14) Known Issues	31
A. Moving Contact or Event Records.....	31
III. Tips	31
IV. Contact Us.....	31

I. Introduction

Description:

The Exchange Manipulator SE ("Server Edition") plug-in is a powerful tool used to exchange data between FileMaker Pro® and one or more Microsoft® Exchange mailboxes. These operations are accomplished using FileMaker function calls from within FileMaker calculations. These calculations are generally determined from within FileMaker "Set Field", "Set Variable" 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 Exchange fields.

Product Version History:

<https://www.productivecomputing.com/products/exchange-filemaker-integration/>

Intended Audience:

Intermediate to advanced developers or persons with knowledge of FileMaker Pro or FileMaker Pro Advanced, especially in the areas of 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) Familiarize yourself with Microsoft Exchange
- 4) Review our FileMaker demo file
- 5) Watch our video tutorials: <https://help.productivecomputing.com/help/exchange-manipulator-se>

II. Integration Steps

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

1) Prerequisites for Plug-in Installation

32-bit vs 64-bit:

The client-side Exchange Manipulator SE plug-in is available as a 32-bit or 64-bit plug-in. The plug-in bit version that you use depends upon your FileMaker Pro bit version; you would use the 32-bit version of the Exchange Manipulator SE plug-in with a 32-bit version of FileMaker Pro, for example. The server-side version is 64-bit only.

Note: 32-bit applications and 32-bit plug-ins will work on a 64-bit operation system.

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"

This link will direct you to download the Microsoft Visual C++ Redistributable Package. 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.

Installing the .NET Framework 4:

Also included in the package is a download link for the Microsoft© .NET Framework 4 installer.

This link will direct you to download the Microsoft .NET Framework 4 installer package. If you are using an older operating system, such as Windows Vista or Windows XP, you will need to download and run the .NET Framework 4 to install it on your machine. Newer operating systems should have a sufficient version of the .NET Framework already installed.

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 Exchange Manipulator SE installation zip package, simply extract the package and open the resulting folder. Install the Exchange Manipulator SE with the following steps:

- 1) Run the "setup.exe" file
- 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!

*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 FileMaker. 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/16.0/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.

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 the 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.

3. 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.
4. 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.
5. 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 PCEX_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-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 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 PCEX_Register, PCEX_GetOperatingMode, and PCEX_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 [ PCEX_Version( "short" ) = "" or PCEX_Version( "short" ) = "?" ]
Show Custom Dialog [ Title: "Warning"; Message: "Plug-in not installed."; Buttons:
"OK" ]
If [ PCEX_GetOperatingMode ≠ "LIVE" ]
Set Field [Main::gRegResult; PCEX_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

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.

For an example of using plug-in script steps, compare two versions of the same script from the Exchange Manipulator SE demo file: Move Mail.

Script 1 – “Move Mail” with calculation (“traditional”) plug-in scripting:

```
Perform Script ["Plug-in Checker"]
Commit Records/Requests [With dialog:Off]

# ...
# Various state checkers go here...
# ...

If [PCEX_OpenMailbox ( Main::gFolder Root ) <> 0]
    # Root mailbox could not be opened
    Show Custom Dialog ["Warning"; PCEX_GetLastError]
    Exit Script [Text Result:]
End If

# Open Folder (sets the internal pointer to the specified folder)
Set Variable [$FolderID; Value: PCEX_OpenFolder ( Mail::Folder )]
If [$FolderID = "!!ERROR!!" or $FolderID = "?"]
    Show Custom Dialog ["Warning"; "Invalid source folder"]
    Exit Script [Text Result:]
End If

# Attempt to open the record:
If [PCEX_OpenRecord( Mail::Entry ID ) <> 0]
    Show Custom Dialog ["Warning"; "No Record found with ID: " & Mail::Entry ID &&
    " OR you are attempting to move a mail item that was deleted."]
    Exit Script [Text Result:]
End If

# Move Mail
# A new Entry ID is generated when a record is moved and must be recaptured
Set Variable [$EntryID; Value: PCEX_MoveRecord ( Main::gFolder Destination )]
If [$EntryID = "!!ERROR!!" or $EntryID = "?"]
    Show Custom Dialog ["Warning"; PCEX_GetLastError]
    Exit Script [Text Result:]
End If

# Update folder field to represent the new location for this email
Set Field [Mail::Entry ID; $EntryID]
Set Field [Mail::Folder; Main::gFolder Destination]
Set Field [Main::gFolder Destination; ""]

Commit Records/Requests [With dialog:Off]
Show Custom Dialog ["Message"; "You have successfully moved this mail to the " &
Mail::Folder & " folder."]
```

Script 2 – “Move Mail” with plug-in script steps:

```
Perform Script ["Plug-in Checker"]
Commit Records/Requests [With dialog:Off]

# ...
# Various state checkers go here...
# ...

PCEX_OpenMailbox [Select; Results:$result; Root Folder Name:Main::gFolder Root]
If [$result <> 0]
    # Root mailbox could not be opened
    Show Custom Dialog ["Warning"; PCEX_GetLastError]
    Exit Script [Text Result:]
End If

# Open Folder (sets the internal pointer to the specified folder)
PCEX_OpenFolder [Select; Results:$FolderID; Folder Path:Mail::Folder]
If [$FolderID = "!!ERROR!!" or $FolderID = "?"]
    Show Custom Dialog ["Warning"; "Invalid source folder"]
    Exit Script [Text Result:]
End If

# Attempt to open the record:
PCEX_OpenRecord [Select; Results:$result; Entry ID:Mail::Entry ID]
If [$result <> 0]
    Show Custom Dialog ["Warning"; "No Record found with ID: " & Mail::Entry ID &&
    " OR you are attempting to move a mail item that was deleted."]
    Exit Script [Text Result:]
End If

# Move Mail
# A new Entry ID is generated when a record is moved and must be recaptured
PCEX_MoveRecord [Select; Results:$EntryID; Folder Path:Main::gFolder Destination]
If [$EntryID = "!!ERROR!!" or $EntryID = "?"]
    Show Custom Dialog ["Warning"; PCEX_GetLastError]
    Exit Script [Text Result:]
End If

# Update folder field to represent the new location for this email
Set Field [Mail::Entry ID; $EntryID]
Set Field [Mail::Folder; Main::gFolder Destination]
Set Field [Main::gFolder Destination; ""]

Commit Records/Requests [With dialog:Off]
Show Custom Dialog ["Message"; "You have successfully moved this mail to the " &
Mail::Folder & " folder."]
```

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; PCEX_OpenRecord as a script step will still open the desired record, and store the value in the \$result global variable (as seen in Script 2), just the same as the Set Variable script step calls PCEX_OpenRecord (which opens the desired 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) Server-Side Deployment

The Exchange Manipulator SE plug-in can run on FileMaker Server within the FileMaker Server Scripting Engine or the Web Publishing Engine. This section will go into detail on how to install the plug-in, how the plug-in handles registration, session management, and functionality.

A. Installation

The server plug-in will need to be installed on the server machine within the FileMaker Server “Extensions” folder. If the desire is to use the plug-in for Perform Script on Server or in server-side scheduled scripts, the plug-in must be installed in the scripting engine’s Extensions folder, located within the “Database Server” folder in FileMaker Server’s install directory. For WebDirect, this Extensions folder would be in the “cwpc” folder (Custom Web Publishing Core) in the Web Publishing Engine. See the paths below for the plug-in installation locations on a Windows server (assuming a default installation path):

Windows:

Database Server:

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

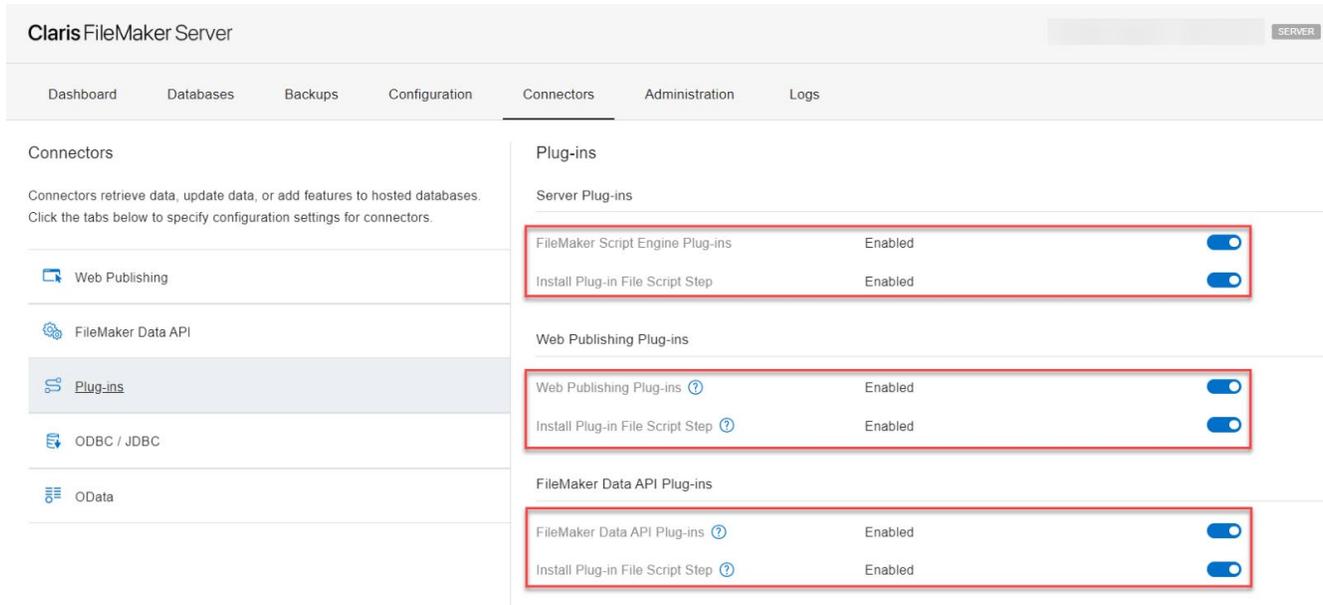
WebDirect & Custom Web Publishing:

C:\Program Files\FileMaker\FileMaker Server\Web Publishing\publishingengine\cwpc\Plugins

The demo file available from our website provides a one-click installation method for installing the Exchange Manipulator SE plug-in into a host FileMaker Server’s Extensions folder; however, it will only install it into the Database Server location above. The demo file must be hosted on FileMaker Server in order to install the plug-in into the Server’s Extensions folder. In order to install the server-side plug-in for use in WebDirect or Custom Web Publishing, the plug-in can be installed by opening the hosted server-side demo file using WebDirect and using the WebDirect Setup tab, or it can be installed manually.

Any manual installation of the plug-in will require the server or web publishing engine to be restarted in order for the plug-in to load.

An important note: In order to allow FileMaker Server to install and use a server-side plug-in from a hosted database, both options in the Connectors > Plug-ins sections must be checked. See screenshot below.



B. Deployment Requirements

In order to utilize the Exchange Manipulator SE server-side plug-in, the plug-in must be installed at the location above on the FileMaker Server machine, and the solution that has scripts designed to utilize the plug-in must be hosted on that FileMaker Server machine. This is easily done by following the instructions for uploading a solution to FileMaker Server, which can be reviewed in the FileMaker Server User's Guide, which comes with every version of FileMaker Server.

C. Registration

Like the client version, the server version of Exchange Manipulator SE must be registered in order to function. In order for the plug-in to function on the server machine, a call to PCEX_Register with valid license information must be called at the beginning of any scripted process. This process should be hard-coded and reference either passed script parameters from a calling script or pre-populated fields within the FileMaker solution. Please see the "Recommended Script Structure" section below for an example of how registration can be utilized.

Depending on the indented deployment of the Exchange Manipulator SE plug-in, scripts utilizing the plug-in may need to be changed.

D. Recommended Script Structure

When a client calls Perform Script on Server, the action reaches out to FileMaker Server with the desired script to allow the server's Scripting Engine to perform the heavy lifting of calculation and data manipulation and return a result back to the caller. For handling plug-in usage, the following is a handy checklist of what is needed to ensure proper plug-in functionality:

- 1) Ensure the script pace starts in the right table occurrence context, with the appropriate record(s) in the found set and available to be accessed.
- 2) Perform a call to PCEX_Register with valid registration information to ensure the plug-in is in the "LIVE" operating mode.
- 3) Perform a call to PCEX_Authenticate with valid Exchange credentials to ensure the plug-in is communicating with the correct Exchange account and is ready to transfer information.

See below for an example of a usage of Perform Script on Server, using scripts from the PCEX_Demo_Serverside demo file:

Script 1: Send Mail (Short-Method) [Server]

```
# Check the status of the solution. This file should be hosted.
Perform Script ["Check Solution Status"]

# Perform the script on the server
Perform Script on Server [Wait for completion; "SVR - Send Mail (Short)"; Parameter:
Mail::ID_Mail_pk]

# Get the result and check for error
Set Variable [$result; Value: Get(ScriptResult)]
If [$result = 0]
    Show Custom Dialog ["Success"; "Successfully sent the email via the Short
Method."]
Else
    Show Custom Dialog ["Error"; "There was an error while sending the email
message: " & "¶" & $result]
End If
```

Script 2: SVR – Send Mail (Short)

```
# Verify the plug-in is registered.
Perform Script ["SVR - Register Plug-in"]
If [Get(ScriptResult) <> 0]
    # There was an error with registration. Return the result back to the client.
    Exit Script [Text Result:Get(ScriptResult)]
End If

# Verify the plug-in is authorized.
Perform Script ["SVR - Authenticate"]
If [Get(ScriptResult) <> 0]
    # There was an error with authentication. Return the result back to the
client.
    Exit Script [Text Result:Get(ScriptResult)]
End If

Set Variable [$mail; Value: Get(ScriptParameter)]
Go to Layout ["Mail" (Mail); Animation:None]

# Find Criteria: Mail::ID_Mail_pk == $mail
Perform Find [Restore]
If [Get(LastError) < 0]
    # Could not locate the mail record to send.
    Exit Script [Text Result:"Could not locate the mail record to send."]
End If

# Open the root Mailbox; this may or may not be the same mailbox as the authenticated
user.
If [PCEX_OpenMailbox( Main::gFolder Root ) <> 0]
    # Could not open the root mailbox. Return the error from the plug-in.
    Exit Script [Text Result:PCEX_GetLastError( "Text" )]
End If

# Send the Mail (For simplicity, we will send with no attachments; check demo for
attachment example)
Set Field [Mail::Result; PCEX_SendMail( Mail::To ; Mail::Cc ; Mail::Bcc ;
Mail::Subject ; Case ( Mail::Body Format = "Rich Text" ; GetAsCSS( Mail::Body ) ;
Mail::Body ) ; "" ; Case( Mail::Body Format = "HTML" or Mail::Body Format = "Rich
Text" ; "Y" ; "N" ) )]

If [Mail::Result = "!!ERROR!!" or Mail::Result = "?"]
    # Error encountered while sending the email. Return the error from the plug-in.
    Exit Script [Text Result:PCEX_GetLastError( "Text" )]
End If

# Update mail state fields (optional)
Set Field [Mail::Modified; Get ( CurrentTimestamp )]
Set Field [Mail::Sent; Get ( not IsEmpty( Mail::Sent ) ; Mail::Sent ; Get (
CurrentTimestamp ) )]

# At the end of the script, since we would have exited if there was an issue, return
0.
Exit Script [Text Result:0]
```

Script 3: SVR – Register Plug-in

```
# Register the plug-in using the license information stored in the solution
Set Variable [$result; Value: PCEX_Register( Main::RegistrationServer;
Main::RegistrationPort; Main::RegistrationPage; Main::RegistrationLicenseID )]
If [$result <> 0]
    Exit Script [Text Result:$result]
Else
    Exit Script [Text Result:0]
End If
```

Script 4: SVR – Authenticate

```
# Verify that the parameter values are present; the authentication process requires
the Email field at minimum.
If [IsEmpty( Main::gProfile Email )]
    Exit Script [Text Result:"Error: The Profile Email field is not populated.
This field is required at minimum."]
End If

# Attempt to Authenticate to Exchange:
Set Field [Main::gAuthentication Result; PCEX_Authenticate( Main::gProfile Email ;
Main::gProfile User Name ; Main::gProfile Password ; Main::gProfile Domain )]

If [Main::gAuthentication Result = "!!ERROR!!"]
    Exit Script [Text Result:PCEX_GetLastError( "Text" )]
Else
    Exit Script [Text Result:0]
End If
```

Script #1 is considered the "Driver" script. This kind of script is usually tied to a button control or some other interactive object, and its purpose is to ensure that the server-side script has what it needs from the client, and then calls the "Worker" script using Perform Script on Server. In a WebDirect environment, the "Driver" script can simply call the "Worker" script directly, or it can call Perform Script on Server if the developer decides to make the server scripting engine's plug-in perform the work.

Script #2 is considered the "Worker" script. This script is the entire scope of the script session and is the "scripted process" mentioned in the earlier sections. At the beginning of the Worker script, a call to register the server-side plug-in (Script #3) and authenticate with Exchange (Script #4) must be made successfully before any further progress can be made. The actual work scripting is identical between the client and server styles, with the notable exception being that there can be no dialog windows used when running in the server session.

Once the Worker script has completed all its script steps, control will return back to the Driver script and the server-side plug-in will release the registration and authentication information that it has and return to the original "neutral" state it was in before the call to the Worker script. You can call multiple Worker scripts *sequentially*, and each call will register the plug-in, authenticate, and perform its task before returning to the Driver script.

E. Server-Side Scheduled Scripts

Scheduled scripts would follow similar adjustments to the order of scripting as scripts called by Perform Script on Server. The “Driver” script, in this case, could be largely ignored, but the script space still needs to be in the correct context, PCEX_Register must still be called, and PCEX_Authenticate must use valid Exchange credentials before any work can commence.

F. WebDirect Script

At a glance, scripting in WebDirect and scripting in FileMaker Pro are identical when it comes to plug-in calls and how they are structured. The only critical difference between WebDirect scripting and standard FileMaker Pro scripting is the requirement of authenticating with an Exchange account. All other features are the same.

When working within a WebDirect session, it’s safe to assume that the session will be similar to that of a FileMaker Pro session. This means that the call to PCEX_Authenticate must take place at least once each time the user is connected to the solution via WebDirect. A developer may wish to include a call to PCEX_Authenticate with each “workflow” similar to server-side scheduled scripts or in a Perform Script on Server session. Similarly, a WebDirect implementation can also make use of Perform Script on Server instead of using script steps and function calls as a client application; in this case, the plug-in used by the Database Engine is the one performing the work, and not the one used by the Web Publishing Engine. It is up to the developer to determine the best setup for their solution when it comes to where to host the server-side version of the plug-in.

8) Talking to Exchange

Talking to Exchange typically requires that you follow these steps.

A. Authenticate to Exchange

If your organization is using an Exchange On-premise server (meaning they are not using an Exchange Online server), authentication should be carried out through the "PCEX_Authenticate" function, which requires the use of the desired account's email, username, password, and optionally the domain and Autodiscover URL. If working on a network that is housing the Exchange server (such as an on-premise version of Exchange 2013 or 2016), the username, password and domain fields can be left blank when making this function call, as the plug-in will use the credentials of the currently logged-in user.

If your organization is using Exchange Online or Office 365, authentication should be carried out through the "PCEX_BeginSession" / "PCEX_Authorize" workflow. The solution will need to specify the App ID and Tenant ID of the Exchange Manipulator app registration in the Exchange Online's Azure AD portal, which will need to be set up as per the instructions below. Once the call to PCEX_BeginSession is made with those values, the user will be prompted to choose the Microsoft account they should authenticate as in a separate webpage and will be brought to a splash page that will contain session information to be copied and pasted back into the solution, such as into a custom dialog text field, to submit to the plug-in and thus complete authentication.

The Exchange Manipulator communicates with the Microsoft Graph API utilizing a connection set within the organization's Azure Active Directory settings. This setting provides an App ID and Tenant ID relating to the Exchange Manipulator's app registration within the Azure AD portal. The App ID and Tenant ID are fed into the PCEX_BeginSession function, which will bring up an external browser window or tab prompting the user to log into their Microsoft account associated with the organization, and guide them through the OAuth 2.0 authentication process before landing them on a splash page that contains the session information to be copied and pasted into a subsequent call to PCEX_Authorize via the solution (such as through a custom dialog or layout field). This will complete authentication, and the plug-in will be able to transfer data to and from Exchange as the authenticated user.

Below are the steps for setting up an app registration to allow the Exchange Manipulator to properly communicate with Exchange Online services:

- 1) Sign into the Microsoft Azure portal for the Exchange Online server with a user that has sufficient privileges to manage app registrations (such as an "admin"-type user)
- 2) If you have access to multiple tenants, use the "Directories + subscriptions" filter to switch to the tenant you want to register the Exchange Manipulator
- 3) Search for and select "Azure Active Directory"
- 4) Under "Manage", choose "App Registrations", then "New Registration"
- 5) Call the app "Exchange Manipulator"
- 6) In the "Supported Account Types" section, select "Accounts in my organizational directory and personal Microsoft accounts"
- 7) In "Redirect URI", add a new "Public client/native" redirect URI with the following value:
 - a. <https://www.productivecomputing.com/authorize/exchange-manip>
- 8) Select "Register"

In “Overview” of the app registration, make note of the Application (client) ID and the Directory (tenant) ID. These will be used for accessing the system and should be stored safely and securely within your FileMaker solution to be given to the PCEX_BeginSession function.

B. Session Management

After connecting to an Exchange Online service via the BeginSession/Authorize workflow, sessions have a limited duration of time. This is due to the nature of the connections, which use OAuth 2.0 as the authentication model, in which access tokens are limited to between 60 and 90 minutes of validity, randomly determined by the Exchange server at time of authentication. After that time expires, the access tokens are considered ‘invalid’, and the session will no longer be able to communicate with the Exchange server. However, the Exchange Manipulator is able to automatically refresh the access token any time it needs to submit a request to Exchange Online and will continually keep track of the up-to-date access token, as well as its requisite refresh token, to ensure the user has a clean, uninterrupted experience with communication.

When preserving connections between FileMaker sessions (e.g. when closing FileMaker down and reopening it the following day), session information can be saved and loaded using the PCEX_SaveSessionInfo and PCEX_LoadSessionInfo functions. After calling PCEX_SaveSessionInfo, an encoded string containing everything the plug-in requires to recreate the session and re-establish connection with Exchange Online is returned to FileMaker. This string should be stored in a secure, local field, such as a field located in a “Main”, “Global” or “Preferences” table, protected by FileMaker security. Loading the session is as simple as passing this string to PCEX_LoadSessionInfo, which will decode, parse and validate the connection information.

As mentioned before, once a connection is established, the plug-in will continually refresh its access token as necessary. With this in mind, the developer is highly advised to make use of PCEX_SaveSessionInfo to ensure that the most up to date connection information is available. This is especially crucial when using Exchange Manipulator in a server-side environment. When working with loading session information, if in a client-side context, if the act of loading a session fails, then the next logical step should be to launch the authentication process anew with a call to PCEX_BeginSession, and follow through to establish a brand new, clean session. This might be necessary, as refresh tokens tend to have a lifetime of their own and could very well last an upwards of six months before refresh attempts will cease to work and a new session is required.

When running Exchange Manipulator on a server, the user should ensure the connection is established by authorizing using the client plug-in, and the solution should save this session immediately to the secure table and field. Then, during the process of the Perform Script on Server workflow, or when a FileMaker Server’s scheduled script runs, the script should load the session from the secure table and field before performing any work exchanging data with Exchange Online.

Here are the general recommended times when to call PCEX_SaveSessionInfo:

- 1) After calling PCEX_Authorize
- 2) At least once every 60 minutes
- 3) After calling PCEX_LoadSessionInfo (as the access token may have required a refresh in the interim)

Not all cases must be handled in the developer’s solution; it all depends on how often the plug-in is used to communicate with Exchange, and the nature of the deployment. For example, a simple client-only deployment of the solution could very well never call PCEX_SaveSessionInfo, where each applicable user gets its own instance of the plug-in and only needs to track its own connection information to Exchange Online, and log in each morning to set up the session for the entirety of the day.

C. Select a Root Folder

The next step is to specify what "root" folder or mailbox will be used. The root folder name is the email address of the Exchange mailbox account that is present in your Exchange environment. By default, the PCEX_Authorize script function will "open" the root folder associated with the authenticated account. If a different root folder is desired, the PCEX_OpenMailbox function will attempt to access the specified root folder using the permissions of the authenticated user. Once the root folder is opened, any subfolder can be accessed, such as Mail folders, Calendar folders, Contact folders, etc.

For example: PCEX_OpenMailbox opens what is commonly known as a "mailbox" and PCEX_OpenFolder opens what is commonly known as a "module" such as the Calendar module. Exchange refers to both the mailboxes and modules as folders.

D. Select a Folder

Now that we have authenticated to Exchange and specified a "root" folder, the next step is to navigate to the desired folder. In order to access any records in an Exchange account, the user must first open the desired folder that contains the desired records. This is accomplished by calling PCEX_OpenFolder(FolderPath) and specifying the 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:

```
PCEX_OpenFolder( "/Inbox/NewItems" )
```

Another example shows the following call opens the folder named "Holiday" located in the "Calendar" Folder:

```
PCEX_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:

```
- PCEX_OpenFolder( "/Calendar" )
```

```
- PCEX_OpenFolder( "Holiday" )
```

It is up to the developer to "know" their location within the folder tree and to know what types of records are contained in the currently opened folder. These records are: Calendar Items, Contact Items, Mail and Post Items, Notes Items, and Task Items.

E. 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 `PCEX_NewRecord(optModule)` function. Any time the `PCEX_NewRecord` function is called the plug-in creates a new record and holds it in memory. The record does not actually exist in Exchange 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 records which are directly or indirectly.

Directly:

If the user knows the Entry ID of the desired record, then they can open that record directly by calling the `PCEX_OpenRecord(EntryID)` function and passing the Entry ID to the function. Similar to the Outlook Manipulator plug-in, the "Entry ID" of a record is the record's unique identifier, indicating where the record exists and how to access it within the Exchange environment.

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.

Another important note is that the Outlook Manipulator's Outlook ID and the Exchange Manipulator SE's Entry ID are not the same and will not be compatible with each other.

Indirectly:

The functions `PCEX_GetFirstRecord` and `PCEX_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 */
Set Field( someField ; PCEX_GetFirstRecord )
Loop
    Exit Loop If[ someField = 'End' or someField < 0]
    /* Access - Modify the fields */
    ...
    /* Opens the next record */
    New Record/Request
    Set Field( someField ; PCEX_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.

F. 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 fields for that record using PCEX_SetFieldData or PCEX_GetFieldData. A list of all available fields can be found in the "Functions Guide."

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

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

The Exchange Manipulator SE plug-in is also capable of working with custom fields or fields attached to the mail, appointment, contact, task, or note record that are not part of the typical set of fields. These custom fields are linked on a per-record basis and can be set or retrieved by using the PCEX_SetCustomFieldData and PCEX_GetCustomFieldData functions.

For more information, see section 11 "Working with Custom Fields" below.

G. Save Record or Send Email

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

If the current record is a new email message (Mail item) then the PCEX_SaveRecord function will cause Exchange to send the email message. Sending an email using SaveRecord is known as the "long method" and requires the following functions (PCEX_NewRecord, PCEX_SetFieldData, PCEX_AddAttachment, PCEX_SaveRecord). You can also send email using the "short method" in a single script step using PCEX_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 Entry 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 PCEX_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 by storing it in the user's Drafts folder. From there, the user can open up the email using an email application such as Microsoft Outlook, Mac Mail, or Mozilla Thunderbird to further edit and then send it. Please reference PCEX_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:

- PCEX_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:

- PCEX_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:

- PCEX_OpenMailbox("SomeUser@thecompany.com")
- PCEX_OpenFolder("Inbox")
- PCEX_NewRecord("Mail")
- PCEX_SetFieldData("To" ; "joe@somecompany.com")
- PCEX_SetFieldData("From" ; "me@mycompany.com")
- PCEX_SetFieldData("Subject" ; "New Release")
- PCEX_SetFieldData("Body" ; "A new product was just released allowing for please contact us")
- PCEX_SetFieldData("Categories" ; work)
- PCEX_AddAttachment("C:\Pricing.pdf" ; "Pricing")
- PCEX_AddAttachment("C:\Overview.pdf" ; "Overview")
- PCEX_AddAttachment("F:\SpecSheet.pdf" ; "SpecSheet")
- PCEX_SaveRecord

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

- PCEX_OpenMailbox(Main::gFolder Root)
- PCEX_OpenFolder(Main::gFolder Mail)
- PCEX_NewRecord("Mail")
- Set Variable [\$Result ; Value:PCEX_SetFieldData("To" ; Mail::To)]
- Set Variable [\$Result ; Value:PCEX_SetFieldData("From" ; Mail::From)]
- Set Variable [\$Result ; Value:PCEX_SetFieldData("Subject" ; Mail::Subject)]
- Set Variable [\$Result ; Value:PCEX_SetFieldData("Body" ; Mail::Body)]
- Set Variable [\$Result ; Value:PCEX_SetFieldData("Categories" ; Mail::Categories)]
- PCEX_AddAttachment(\$FullPath1 ; \$FileName1)
- PCEX_AddAttachment(\$FullPath2 ; \$FileName2)
- PCEX_AddAttachment(\$FullPath3 ; \$FileName3)
- PCEX_SaveRecord

H. Responding to a Meeting Request

The Exchange Manipulator SE has the ability to send a meeting response to the organizer of a meeting. To respond to a meeting request, the following steps should be performed:

- 1) Open the Meeting event using PCEX_OpenRecord
- 2) Set the field "Meeting Response" with either "Accepted", "Declined", or "Tentative"
- 3) Save the record with a call to PCEX_SaveRecord("ApptResponse")

The use of the parameter "ApptResponse" will create a meeting response message in Exchange 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 Exchange to be sent to the organizer of the meeting with the desired response.

Note: When declining a meeting invitation, the Exchange environment will automatically delete that event from the calendar folder. Because of this, **the call to PCEX_SaveRecord("ApptResponse") will return a 0 when successfully declining a meeting**; it is up to the developer to ensure that the FileMaker script appropriately clears out the Entry ID in FileMaker for that record when it is declined.

Example script demonstrating the acceptance of a meeting invitation:

```
# Open the default Calendar
Set Variable [ $result ; Value:PCEX_OpenDefaultFolder( "Calendar" ) ]
# Error handling...
#
# Open the record
Set Variable [ $result ; Value:PCEX_OpenRecord( $EntryID ) ]
# Error handling...
#
# Set the Meeting Response field with the value of "Accepted"
Set Variable [ $result ; Value:PCEX_SetFieldData( "Meeting Response" ; "Accepted" ) ]
#
# Save the record and send the response
Set Variable [ $EntryID ; Value:PCEX_SaveRecord( "ApptResponse" ) ]
# Error Handling...
```

Example script demonstrating the declining of a meeting invitation:

```
# Open the default Calendar
Set Variable [ $result ; Value:PCEX_OpenDefaultFolder( "Calendar" ) ]
# Error handling...
#
# Open the record
Set Variable [ $result ; Value:PCEX_OpenRecord( $EntryID) ]
# Error handling...
#
# Set the Meeting Response field with the value of "Accepted"
Set Variable [ $result ; Value:PCEX_SetFieldData( "Meeting Response" ; "Declined" ) ]
#
# Save the record and send the response
Set Variable [ $EntryID ; Value:PCEX_SaveRecord( "ApptResponse" ) ]
If [ $EntryID = 0 ]
    # Clear the Entry ID field; the record has been deleted from Exchange.
    Set Field [ Event::EntryID ; Value: "" ]
Else
    # Error handling...
End If
```

9) 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 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 Exchange. 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) For verification of the "Send As" feature, we recommend using Microsoft Outlook. Make sure both the user you wish to send as and the user that is allowed to send as are added to 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.

When sending as another user, because of the way the Exchange Manipulator SE connects to the other user's account, the email message will appear in that user's Sent Items folder. This is different from the original Outlook Manipulator plug-in, which had impersonated emails appear in the original user's Sent Items folder, as opposed to the impersonated user's Sent Items folder.

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.

Again, 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.

10) Moving or Deleting a Record

A. Move a Record

After a record has been opened, call the `PCEX_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 `OpenFolder` must be called to return to the previously opened folder.

The following caveats must be recognized:

- 1) When you move records within the same mailbox, the leading "/" character for the destination folder should be included. For example: `/folderpath` or `/Inbox/Test`
- 2) As of version 1.0.0.0, the Exchange Manipulator SE plug-in is unable to move records between mailboxes or to/from the Public Folders group.

B. Delete a Record

In order to delete records you can use the `PCEX_DeleteRecord(EntryID ; Permanent)`, `PCEX_DeleteCurrentRecord(bPermanent)` or `PCEX_DeleteAllRecords(ExcludePrivate ; ExcludeDistList)` functions. The `DeleteRecord` function will delete a specific record indicated by its Entry ID, from any folder in the current mailbox tree. By using the optional `Permanent` parameter the record can be permanently deleted or sent to the mailbox's "Deleted Items" folder 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. Please reference the "Functions Guide" for additional information about these functions.

11) Working with Custom Fields

Records in Exchange also contain custom fields, or “user-defined” fields, in addition to the standard fields. The PCEX_SetCustomFieldData and PCEX_GetCustomFieldData functions are used when working with these custom fields. Let’s explore some sample uses of these functions.

PCEX_SetCustomFieldData(FieldName ; FieldData ; FieldType) 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 Exchange, then you would follow these steps:

```
...
PCEX_NewRecord( "Contacts" )
PCEX_SetFieldData( "First Name" ; "Brett" )
PCEX_SetFieldData( "Last Name" ; "Wilcox" )
PCEX_SetFieldData( "Company" ; "ABC Company" )
PCEX_SetCustomFieldData( "Pets" ; "Molly" ; "Text" )
PCEX_SaveRecord
```

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

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

```
...
PCEX_GetFirstRecord
PCEX_GetFieldData( "First Name" )
PCEX_GetFieldData( "Last Name" )
PCEX_GetFieldData( "Company" )
PCEX_GetCustomFieldData( "Pets" ; "Text" )
...
```

For more information on what field types are available, please refer to the “Functions Guide”.

12) Filtering Records

The **PCEX_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.

When filtering records by their last modified time, the procedure should be performed as listed in the following pseudocode:

```
PCEX_OpenMailbox( $DesiredMailbox )
PCEX_OpenFolder( $DesiredFolder )
PCEX_FilterByLastModified( $Timestamp )
Set variable $count = PCEX_GetRecordCount
If $count > 0
    Process Records until "End"
End If
```

Please note filtering by other fields beyond the last modified timestamp is not available.

13) 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 PCEX_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: " & PCEX_GetLastError ]
End If
```

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

The list below includes return codes that the plug-in directly reports on. For some errors, such as the error codes -15003, -400 and -998, the error text can vary due to different causes. Please refer to the result of PCEX_GetLastError("Text") for more information when encountering those particular error codes.

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
-998	Unknown Exception
-999	COM Exception
-15000	Exchange Logon Failed
-15001	Authentication information missing; please authenticate to Exchange first

Code	Description
-15002	This Version is not supported
-15003	Error Encountered from Exchange
-15004	Authenticated user has insufficient Exchange permissions
-15005	Exchange service has not yet been authenticated
-15006	This function is not supported in the current version of the Exchange Manipulator SE
-15007	Operation called without opening/creating a record or folder
-15008	Missing parameter value
-15009	Error encountered during authentication
-15011	Invalid record or folder type
-15012	Invalid path or file name

14) Known Issues

A. Moving Contact or Event Records

As of publication of this guide, the Graph API does not support moving contact records or event records between folders. This is a known issue on Microsoft's end, and the Graph API development community has made them aware of interest in adding these features into the API.

"In version 3.0.0.0, Exchange Manipulator's integration method switched from the Exchange Web Services SDK to the Microsoft Graph API. Unfortunately, the Graph API does not currently support the moving of contacts or calendar events between folders. This is a known issue and is being monitored for future enhancement."

III. Tips

This "Developer's Guide" is designed to give you a basic understanding of how Exchange 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 Entry ID is a unique identifier created by Exchange. This ID is unique across all folders (Mail, Contacts, Calendar, Tasks, and Notes) and is unique to every Exchange user. In other words, you can store the Entry 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 Entry ID is often longer than 120 characters.

The plug-in cannot edit or maintain the list of categories. This is done exclusively in an email application such as Outlook.

IV. 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

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!

**This document was updated on November 15, 2023*