Productive Computing

Developer's Guide

Revised January 17, 2014

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

Table of Contents

I. Introduction

Description

The eMail Marketing Connector plug-in is a tool used to bidirectionally exchange data between FileMaker® and Constant Contact®. Now you can import, add, update and remove contacts or lists between FileMaker and Constant Contact allowing you to effectively manage your eMail Marketing. These operations are accomplished using FileMaker function calls from within FileMaker calculations. These calculations are generally determined from within FileMaker "SetField," "SetVariable" or "If" script steps. For a list all plug-in functions and available fields please see the accompanying Functions Guide document.

Product Version History

http://www.productivecomputing.com/email-marketing/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) Reverse engineer our FileMaker demo and review video tutorials
Demo and video tutorials: http://www.productivecomputing.com/email-marketing
4) Familiarize yourself with Constant Contact: http://www.constantcontact.com

Error Handling:

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

II. Integration Steps

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

1) Installing the Plug-in

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

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. On Mac the plug-in will have a ".fmplugin" extension.

3) Copy the actual plug-in and paste it to the Extensions folder which is inside the FileMaker program folder.
   - On Windows this is normally located here: C:\Program Files\FileMaker\FileMaker X\Extensions
   - On Mac this is normally located here: Volume/Applications/FileMaker X/Extensions (Volume is the name of the mounted volume)

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.

2) Install Component for Windows 8

Installing the Microsoft Visual C++ 2008 Redistributable Package on Windows 8:

Included in the package is a download link for all users of Windows 8.
Name of link is: "Download Microsoft Visual C++ 2008 Redistributable Package (x86) (Windows 8 Install)"

This link will direct you to download the Microsoft Visual C++ Redistributable Package (x86). Windows 8 does not have a Visual C++ 2008 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 (ie. 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 8 need to install the 64-bit ("x64") version of the redistributable package, which is also available from Microsoft.

3) 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 PCXP_GetOperatingMode function.

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

Why do I need to Register?

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

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

How do I hard code the registration process?

You can hard code the registration process inside a simple "Plug-in Checker" script. The "Plug-in Checker" script should be called at the beginning of any script using a plug-in function and uses the PCXP_Register, PCXP_GetOperatingMode and PCXP_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 [ PCXP_Version( "short" ) = "" or PCXP_Version( "short" ) = "?" ]
Show Custom Dialog [ Title: "Warning"; Message: "Plug-in not installed."; Buttons: "OK" ]
If [ PCXP_GetOperatingMode ≠ "LIVE" ]
Set Field [Main::gRegResult; PCXP_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" ]
```

Please also watch our setup video found here: http://www.productivecomputing.com/video/?p=1387 for additional setup information.

4) Authenticate with Constant Contact

Before you can authenticate with a Constant Contact account, you must first sign up for an account with Constant Contact. Please navigate to http://www.constantcontact.com/index.jsp and click on "Sign Up" to create a new account. You can sign up for a free trial account or sign up for a live account. Either account type will work with the eMail Marketing Connector.

Now that you have your Constant Contact account name and password, you can use these credentials to authenticate with your Constant Contact account. When communicating with Constant Contact you must first authenticate to your Constant Contact account. This is accomplished by calling the PCXP_Authenticate(UserName ; Password) function and passing your Constant Contact user name and password to the corresponding parameters. For example: PCXP_Authenticate("Melinda" ; "somePassword"). If you desire to switch between Constant Contact accounts, then you must first simply authenticate with your different Constant Contact account before calling any additional plug-in functions. All plug-in function calls are applied to the currently authenticated Constant Contact account.

5) Lists

Constant Contact has lists and contacts. In this section we will discuss how to deal with lists. A list in Constant Contact is an electronic list containing specific subscriber contact information that allow for widespread distribution of information. Each list has a unique name and unique list ID number. The list name can be seen in your Constant Contact account in your web browser, however the list ID is typically not exposed in your web browser interface. Alternatively the list ID AND list name are both available in FileMaker by using our plug-in. Let's further explore how to use the plug-in functions to see how we can push and pull lists between FileMaker Pro and Constant Contact.

Create a New List

The functions involved to create a new list in Constant Contact from FileMaker are PCXP_NewList(Name), PCXP_SetListField(FieldName ; Value) and PCXP_SaveList. Let's have a look at an example to better illustrate how to create a new list.

These are the script steps to create a new list titled "Newsletter":

```
...
Set Variable [$$Result; Value:PCXP_NewList( "Newsletter" )]
Set Variable [$$Result; Value:PCXP_SaveList]
...
```

These are the script steps to create a new list titled "Sales" with the OptInDefault set to true:

```
...
Set Variable [$$Result; Value:PCXP_NewList( "Sales" )]
Set Variable [$$Result; Value:PCXP_SetListField( "OptInDefault" ; "true" )]
Set Variable [$$Result; Value:PCXP_SaveList]
...
```

The main field you will be setting in a list is the list name, which you set when calling the PCXP_NewList(Name) function. However, in the particular case where you need to set additional fields in a list, please use the PCXP_SetListField function and refer to the "Available Constant Contact Fields" section at the end of the accompanying Functions Guide. We will discuss how to add contacts to your new list under the contacts section of this document, so please keep reading!

Update a List

Once you have an existing list, then you may desire to update fields associated with that list. This can be accomplished by getting all your lists, iterating through your lists and setting the fields for the desired list. You would use these functions in order to update the list name: PCXP_GetAllLists, PCXP_ GetFirstList, PCXP_ GetNextList, PCXP_ SetListField(FieldName ; Value) and PCXP_SaveList. Once you get a list using the PCXP_ GetFirstList or PCXP_ GetNextList, then that list is open for editing, therefore you can change the list fields (including the list name) with a subsequent call to PCXP_SetListField(FieldName ; Value). Afterwards ensure you always call PCXP_SaveList to save any changes you made to that list. Please refer to the "Available Constant Contact Fields" section at the end of the accompanying Functions Guide for a list of all available fields.

Delete a List

In order to permanently delete a list you will call the PCXP_DeleteList(ListID) function and pass the unique list ID. For example: PCXP_DeleteList("22"). Please note that this operation will not delete any of the contacts associated with the list you are deleting. Only the list will be deleted. There are two caveats, which are that the 3 built in Constant Contact lists, which are "active," "donotmail" and "removed" cannot be deleted. In addition if you attempt to delete your default list, then the default list will NOT be deleted, but rather the contacts will be removed from the default list. After you delete a list, then we also recommend calling the PCXP_GetAllLists function in order to reindex the lists stored in the plug-in memory. This is a crucial step, especially if you plan on subsequent calls to PCXP_GetFirstList and PCXP_GetNextList as you will want an accurate count of all existing lists after any deletions. Please use the PCXP_DeleteList function with caution in order to avoid any undesirable effects.

Import Lists

In order to import all lists the following functions should be used: PCXP_GetAllLists, PCXP_GetFirstList, PCXP_GetListField(FieldName) and PCXP_GetNextList. Basically you would get all your lists, iterate through all your lists and get all the desired field values for each list. For example:

```
...
#Get All Lists
Set Variable [$$Result; Value:PCXP_GetAllLists]
#Get First List
Set Variable [$$Result; Value:PCXP_GetFirstList]
Loop
  Set Field [Lists::Name; PCXP_GetListField("Name")
  Set Field [Lists::OptInDefault; PCXP_GetListField("OptInDefault")
  Set Field [Lists::SortOrder; PCXP_GetListField("SortOrder")
  Set Field [Lists::ContactCount; PCXP_GetListField("ContactCount")
  #Gets Next List
  Set Variable [$$Result; Value:PCXP_GetNextList]
Exit Loop If [$$Result = "END" ]
...
```

6) Contacts

Constant Contact has lists and contacts. In this section we will discuss how to deal with contacts. A contact in Constant Contact is a subscriber, who has a unique email address and additional basic contact information such as name, phone, etc.

Push a New Contact

In order to add a new contact to Constant Contact from FileMaker Pro you should call these functions: PCXP_NewContact, PCXP_SetContactField(FieldName ; Value) and PCXP_SaveContact. Please refer to the "Available Constant Contact Fields" section at the end of the accompanying Functions Guide for a list of all available contact fields.

For example:

...
#Create New Contact
Set Variable [$$Result; Value:PCXP_NewContact("joe@abc.test.com" ; "3"]
#Set Fields for Contact
Set Variable [$$Result; Value:PCXP_SetContactField("FirstName" ; Contacts Global::gFirstName)]
Set Variable [$$Result; Value:PCXP_SetContactField("LastName" ; Contacts Global::gLastName)]
Set Variable [$$Result; Value:PCXP_SetContactField("HomePhone" ; Contacts Global::gHomePhone)]
Set Variable [$$Result; Value:PCXP_SetContactField("Note" ; Contacts Global::gNote)]
Set Field [Contacts Global::gContact_ID; PCXP_SaveContact]
...

Update Contacts

In order to update information for an existing contact in Constant Contact from FileMaker Pro you should call these functions: PCXP_OpenContact(IDorEmail), PCXP_SetContactField(FieldName ; Value) and PCXP_SaveContact.

For example:

...
#Open Existing Contact
Set Variable [$$Result; Value:PCXP_OpenContact("joe@abc.test.com"]
#Set Fields for Contact
Set Variable [$$Result; Value:PCXP_SetContactField("FirstName" ; Contacts Global::gFirstName)]
Set Variable [$$Result; Value:PCXP_SetContactField("LastName" ; Contacts Global::gLastName)]
Set Variable [$$Result; Value:PCXP_SetContactField("HomePhone" ; Contacts Global::gHomePhone)]
Set Variable [$$Result; Value:PCXP_SetContactField("Note" ; Contacts Global::gNote)]
Set Field [Contacts Global::gContact_ID; PCXP_SaveContact]
...

Remove Contacts

There are two options for removing contacts. The first option is to opt a contact out of a mailing list. The second option is to remove a contact from a list. Let's have a closer look at these two options.

Opt Out Option

This option will set the contacts's status so that they are opted out from any mailing list. In order to achieve this you should call the following functions: PCXP_OpenContact(IDorEmail), PCXP_OptOutContact and PCXP_SaveContact.

Remove From List Option

This option will remove the currently selected contact from the provided list. In order to achieve this you should call the following functions: PCXP_OpenContact(IDorEmail), PCXP_RemoveFromList(ListID) and PCXP_SaveContact.

Import Contacts

You may desire to import all your contacts from Constant Contact into FileMaker. This can be accomplished by calling PCXP_GetAllContacts, PCXP_GetFirstContact, PCXP_GetContactField and PCXP_GetNextContact.

For example:

```
...
#Get All Contacts
Set Field [Main::gLatestUpdate; PCXP_GetAllContacts]
#Get First Contact
Set Variable [$$Result; Value:PCXP_GetFirstContact]
Loop
   New Record/Request
   Set Variable [$$Result; Value:PCXP_OpenContact( $$Result )]
   Set Field [Contacts::EmailAddress; PCXP_GetContactField("EmailAddress")]
   Set Field [Contacts::FirstName; PCXP_GetContactField("FirstName")]
   Set Field [Contacts::LastName; PCXP_GetListField("LastName")]
   #Get Next Contact
   Set Variable [$$Result; Value:PCXP_GetNextContact]
Exit Loop If [$$Result = "END" ]
...
```

If you want to get the contacts from a specific list, then you would introduce the PCXP_GetContactsFromList function. This function retrieves all the contacts from the currently selected list. A previous call to PCXP_GetFirstList or PCXP_GetNextList is required to select the desired list.

Import Updated Contacts

The plug-in offers the ability to just import the updated contacts. This can be quite time consuming and efficient when you just desire to import any contacts that have been updated or changed. The PCXP_GetUpdates(SinceDate ; ListTypeOrID) function retrieves all of the contacts that have been updated since the previous date passed to the SinceDate parameter.

For example:

```
...
#Get all active contacts updated since a previous date
Set Variable [$$Result; Value:PCXP_GetUpdates( Main::gLatestUpdate ; “active” )]
#Get First Contact
Set Variable [$$Result; Value:PCXP_GetFirstContact]
Loop
   New Record/Request
   Set Variable [$$Result; Value:PCXP_OpenContact( $$Result )]
   Set Field [Contacts::EmailAddress; PCXP_GetContactField(“EmailAddress”)]
   Set Field [Contacts::FirstName; PCXP_GetContactField(“FirstName”)]
   Set Field [Contacts::LastName; PCXP_GetListField(“LastName”)]
   #Get Next Contact
   Set Variable [$$Result; Value:PCXP_GetNextContact]
Exit Loop If [$$Result = “END” ]
...
```

7) Handling Errors

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

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

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

The PCXP_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 PCXP_GetLastError("Number"), then an error number will be returned. If format parameter is empty such as PCXP_GetLastError or PCXP_GetLastError("Text"), then an English error description will be returned. The error numbers and their meanings can be found below.

Error Number	Error Text
0	Success
-1	Plug-in not registered or session expired
-3	Invalid # of Parameters
-4	Invalid Parameter value(s)
-10	Failed Registration
-12001	Must open or create a new contact.
-12002	Constant Contact Error: "additional text provided by Constant Contact"
-12003	Must authenticate first.
-12004	Invalid XML:
-12005	Must retrieve contacts first.
-12006	No contacts in list.
-12007	No match for key.
-12008	No match for list ID or type.
-12009	Must retrieve lists first.
-12010	There are no lists in this account.
-12011	Must open or create a new list.
-12012	No such contact.

8) Known Issues

As the eMail Marketing Connector communicates with Constant Contact through HTTP requests, certain characters have the potential of causing unexpected or undesirable errors or effects during transmission. When adding or updating a Contact or Contact List using the eMail Marketing Connector, it is a good practice to ensure that there are no special characters in the data values of the Contact or Contact List fields being passed. An example of special characters would be the use of Smart Quotes (‘, ”) in a FileMaker data field, which can be disabled through the File Options menu item in the File menu.

III. Contact Us

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

Phone: 760-510-1200
Email: support@productivecomputing.com
Forum: www.productivecomputing.com/forum

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