



# Developer's Guide

Revised September 3, 2013

# Table of Contents

<b>I. INTRODUCTION .....</b>	<b>3</b>
<b>II. BASIC INTEGRATION STEPS .....</b>	<b>4</b>
1) Install the Plug-in .....	4
2) Register the Plug-in.....	5
3) Install Component for Windows 8.....	6
4) Open the PDF.....	7
5) Get/Set Field Data.....	7
6) Save the PDF (optional) .....	8
7) Close the PDF (recommended).....	8
<b>III. SAMPLE SCRIPTS .....</b>	<b>9</b>
Push data from FileMaker to PDF form fields.....	9
Get PDF form fields Names .....	10
Pull data from PDF form fields into FileMaker .....	11
<b>IV. ADDITIONAL FUNCTIONALITY .....</b>	<b>12</b>
Access Text From a Page or Number of Pages .....	12
Access Document Information (metadata).....	12
Combine, Delete, or Print PDF Files .....	12
<b>V. CONTACT US .....</b>	<b>13</b>

# I. Introduction

## Description

The PDF Manipulator plug-in offers the ability to exchange data between FileMaker® and Adobe® Acrobat. With the plug-in FileMaker users are able to “push” and “pull” information to and from PDF form fields. The FileMaker user can also extract page text, extract or modify PDF metadata, combine several PDF documents into a single PDF, delete and print PDF files. These operations are accomplished by using FileMaker function calls from within FileMaker calculations. These calculations are generally determined from within FileMaker “SetField” or “If” script steps. For a list of available plug-in functions and their functionality please see the accompanying Functions Guide.

## Product Version History

[http://www.productivecomputing.com/pdf-integration/version\\_history](http://www.productivecomputing.com/pdf-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) Review our FileMaker Demo and video tutorials

Demo and video tutorials: <http://www.productivecomputing.com/pdf-integration>

- 4) Familiarize yourself with Adobe Acrobat and how to create PDFs with form fields

## Technical Note

File and full paths are formatted as follows:

**Mac** - Paths take the form of MountedVolume/path. Paths can also be user- relative (e.g., ~/Desktop )

For Example: /Volumes/MacHD/PDF Template.pdf

**Windows** - Paths take the form of DriveLetter:\path or \\ServerName\path

For Example: C:\Documents and Settings\User\Desktop\PDF Template.pdf or \\Server1\Docs\PDF Template.pdf

## II. Basic Integration Steps

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

### 1) Install the Plug-in

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

#### FileMaker 12:

- 1) Open the FileMaker demo file available in the plug-in bundle ([www.productivecomputing.com](http://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.

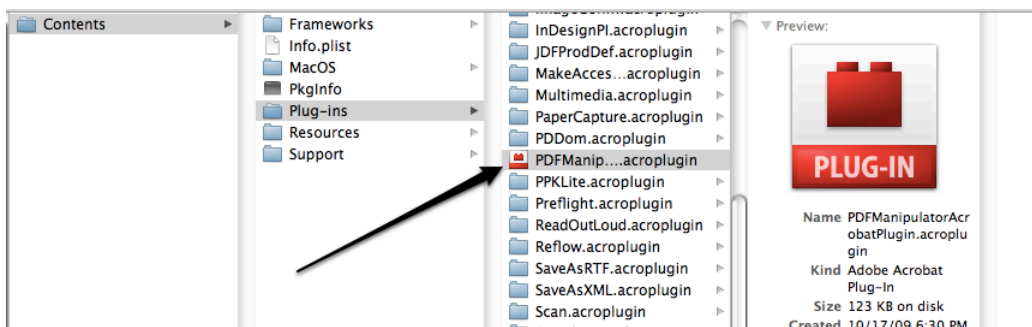
- 1) Quit FileMaker Pro completely.
- 2) Locate the plug-in in your download which will be located in a folder called "Plug-in." On Windows the plug-in will have a ".fmx" extension. On Mac the plug-in will have a ".fmplugin" extension.
- 3) Copy the actual plug-in and paste it to the Extensions folder 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.

#### Installing the Acrobat Plug-in (Mac OSX ONLY):

On Macintosh platforms you will need to install a second plug-in for Acrobat called "PDFManipulatorAcrobatPlug-in.acroplugin." To install this plug-in, do the following:

- 1) Quit Adobe Acrobat.
- 2) Navigate to the Acrobat Application (usually called "Adobe Acrobat Professional.app") in the Applications folder. For example: Applications > Adobe Acrobat 9 Pro > Adobe Acrobat Pro.
- 3) Right Click (or control click) on Adobe Acrobat Pro and select "Show Package Contents."
- 4) A new window will appear. Open the folder in the new window that appears called "Contents" and select the "Plug-ins" folder.
- 5) Copy and paste the Acrobat plug-in provided by Productive Computing directly into this folder called

Contents > Plug-ins as shown in the screen shot below.



## 2) Register 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.

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.

### 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 PCPF\_Register, PCPF\_GetOperatingMode and PCPF\_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 [ PCPF_Version( "short" ) = "" or PCPF_Version( "short" ) = "?" ]  
Show Custom Dialog [ Title: "Warning"; Message: "Plug-in not installed."; Buttons: "OK" ]  
If [ PCPF_GetOperatingMode ≠ "LIVE" ]  
Set Field [Main::gRegResult; PCPF_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" ]
```

### 3) 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.

## 4) Open the PDF

Now that the plug-in is installed and registered, let's discuss some of the basic steps involved in successfully using the plug-in. Moving information between FileMaker and Acrobat first involves opening the desired PDF file by calling the PCPF\_Open function( FullPath ). The FullPath parameter is the system path to the desired PDF file.

If multiple versions of Acrobat exist on the system, it is best to have the desired version of Acrobat opened before calling the PCPF\_OpenPDF function.

## 5) Get/Set Field Data

After the desired PDF document is opened, then you can either SET or GET the value of the PDF form fields. If you are pushing data from FileMaker to the PDF, then you will need to use the PCPF\_SetPDFFieldValue function. If you are pulling data from the PDF into FileMaker, then you will need to use the PCPF\_GetPDFFieldValue function. Both the PCPF\_SetPDFFieldValue and the PCPF\_GetPDFFieldValue functions require the FieldName or the PDF form field name. If you do not know the name of PDF form field to pass to these functions, then you can retrieve the form field names by calling the PCPF\_GetFieldNames function.

### **Setting Field Data (Pushing)**

The PCPF\_SetPDFFieldValue( FieldName ; FieldValue ) function sets the FileMaker values into the corresponding PDF form fields. The name of the PDF form field is passed to the function along with the FileMaker value to populate the field.

For example:

```
PCPF_SetPDFFieldValue( "Form1.Subform.FirstName" ; "Tom" ) or
```

```
PCPF_SetPDFFieldValue( SomePDFFormFieldName ; SomeFileMakerTable::SomeFileMakerFieldName )
```

### **Getting Field Data (Pulling)**

The PCPF\_GetPDFFieldValue( FieldName ) function retrieves the values of the specified PDF form field. The FieldName is the full name of the PDF form field from which to retrieve a value. The form field name is passed to the function and whatever value is contained in the PDF form field is pulled into FileMaker.

For example:

```
PCPF_GetPDFFieldValue( "First Name" ) or
```

```
PCPF_GetPDFFieldValue( SomePDFFormFieldName )
```

## **6) Save the PDF (optional)**

After you have set the values in the PDF or in other words you have pushed values from FileMaker into the corresponding PDF form fields, you can save the changes to the PDF. You have the option to use PCPF\_SavePDF function to save the changes to the currently opened PDF or you can call the PCPF\_SaveAsPDF( FullPath ) function to save the currently opened PDF to the new path specified.

## **7) Close the PDF (recommended)**

Lastly you will then need to close the PDF using the PCPF\_ClosePDF function. We always recommend closing the PDF once you are finished.



### III. Sample Scripts

Now that you understand the basic integration steps involved for getting or setting PDF form field values, let's have a look at some sample script steps. Please understand that there are many different ways to construct your scripts and these are just a few examples to get you started. For example, you could hard code the FieldValues such as "Tom" or you could reference the FileMaker field names such as Main::FirstName. You can also add loops to populate or retrieve the values depending on your design. The design depends on the experience and creativity of the FileMaker developer.

#### Push data from FileMaker to PDF form fields

##### Sample Script 1:

**\*\*This script assumes you have a PDF with "Form1.Subform.FirstName," "Form1.Subform.LastName," "Form1.Subform.Company," "Form1.Subform.Date," "Form1.Subform.Email" and "Form1.Subform.Phone" form field names and a FileMaker table named "Main" with the "FirstName," "LastName," "Company," "Date," "Email," "Phone" and "Result" fields.\*\***

```
#Opens the desired PDF
Set Field[ Main::Result ; PCPF_OpenPDF ( "Some PDF file path" ) ]
#Sets the specified form field with values from the corresponding FileMaker field
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.FirstName" ; Main::FirstName ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.LastName" ; Main::LastName ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Company" ; Main::Company ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Date" ; Main::Date ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Email" ; Main::Email ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Phone" ; Main::Phone ) ]
Set Field[ Main::Result ; PCPF_SaveAsPDF( "Some other PDF file path" ) ]
Set Field[ Main::Result ; PCPF_ClosePDF ]
```

##### Sample Script 2:

**\*\*This script assumes that you have a PDF with "Form1.Subform.FirstName," "Form1.Subform.LastName," "Form1.Subform.Company," "Form1.Subform.Date," "Form1.Subform.Email" and "Form1.Subform.Phone" form field names and that you are hard coding the FieldValues.\*\***

```
#Opens the desired PDF
Set Field[ Main::Result ; PCPF_OpenPDF ( "Some PDF file path" ) ]
#Sets the specified form field with the hard coded field values
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.FirstName" ; "Tom" ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.LastName" ; "Smith" ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Company" ; "ABC Company" ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Date" ; "10/31/2009" ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Email" ; "tom@abc.co.com" ) ]
Set Field[ Main::Result ; PCPF_SetPDFFieldValue ( "Form1.Subform.Phone" ; "760-510-1200" ) ]
Set Field[ Main::Result ; PCPF_SaveAsPDF ( "Some other PDF file path" ) ]
Set Field[ Main::Result ; PCPF_ClosePDF ]
```

## Get PDF form fields Names

Sample script 1 and 2 assume that you know the PDF form field names. If you do not know the PDF form field names, then you will need to use the PCPF\_GetFieldNames function to obtain the PDF form field names. Let's look at a basic and more advanced method of obtaining the PDF form field names.

The basic method will simply pull a list of all form field names into a single text field for future reference. The advanced method uses the same principle except that after the field list is gathered, then we parse through the list and add a record in FileMaker for every individual field name. In addition we also pull the value for each field from the form with this script. Let's take a look at these two basic and advanced samples.

### Sample Script (Basic):

**\*\*The script that best demonstrates this concept in our demo is named: "Extract Fields From PDF." We encourage you to perform this script while running debugger in the demo for a better understanding of this concept as documentation alone may not suffice.\*\***

```
#Opens the desired PDF
Set Field [ Main::Result; PCPF_OpenPDF( "Some PDF file path" ) ]
#Gets a list of the PDF form fields
Set Field [ Main::gPDF Field Listing; PCPF_GetFieldNames( 0 ) ]
```

### Sample Script (Advanced):

**\*\* The script that best represents this concept in our demo is named: "Extract From PDF." We encourage you to perform this script while running debugger in the demo for a better understanding of this concept as documentation alone may not suffice.\*\***

```
Go to Layout [ "Form Field Names" (Form Field Names) ]
Show All Records
Delete All Records
New Record/Request
#Opens the desired PDF
Set Field [ Main::Result; PCPF_OpenPDF( "Some PDF file path" ) ]
#Get List of Field Names
Set Field [ Form Field Names::gListOfNames; PCPF_GetFieldNames( 0 ) ]
Set Field [ Form Field Names::gCount; ValueCount ( Form Field Names::gListOfNames ) ]
Set Field [ Form Field Names::gCounter; 1 ]
#Enter loop, create a new record, and get field name
Loop
Set Field [ Form Field Names::Name; Substitute (Trim( MiddleValues ( Form Field Names::gListOfNames;
Form Field Names::
gCounter; 1)) ; ¶; "" ) ]
Set Field [ Form Field Names::Path Current File; Main::Path Currently Open File ]
Set Field [ Form Field Names::Value; PCPF_GetPDFFieldValue( Form Field Names::Name ) ]
Set Field [ Form Field Names::gCounter; Form Field Names::gCounter + 1 ]
Exit Loop If [ Form Field Names::gCounter > Form Field Names::gCount ]
New Record/Request
End Loop
#Exit loop when you reach the last field on the list
End If
Go to Layout [ original layout ]
```

## Pull data from PDF form fields into FileMaker

Now that you know how to push from FileMaker to PDF form fields and how to obtain the exact PDF form field names, let's look at some sample scripts of how to pull data from PDF form fields into FileMaker.

### Sample Script 5:

**\*\*This script assumes you have a FileMaker table named "Main" with the "FirstName," "LastName," "Company," "Date," "Email," "Phone" and "Result" fields and that you have a PDF with "Form1.Subform.FirstName," "Form1.Subform.LastName," "Form1.Subform.Company," "Form1.Subform.Date," "Form1.Subform.Email" and "Form1.Subform.Phone" form field names.\*\***

```
#Opens the desired PDF
Set Field[ Main::Result ; PCPF_OpenPDF (PDF file path) ]
Set Field[ Main::FirstName ; PCPF_GetPDFFieldValue ( "Form1.Subform.FirstName" ) ]
Set Field[ Main::LastName ; PCPF_GetPDFFieldValue ( "Form1.Subform.LastName" ) ]
Set Field[ Main::Company ; PCPF_GetPDFFieldValue ( "Form1.Subform.Company" ) ]
Set Field[ Main::Date ; PCPF_GetPDFFieldValue ( "Form1.Subform.Date" ) ]
Set Field[ Main::Email ; PCPF_GetPDFFieldValue ( "Form1.Subform.Email" ) ]
Set Field[ Main::Phone ; PCPF_GetPDFFieldValue ( "Form1.Subform.Phone" ) ]
Set Field[ Main::Result ; PCPF_ClosePDF ]
```

### Sample Script 6:

**\*\*This script assumes you do not yet know the exact PDF form field names.\*\***

```
#Opens the desired PDF
Set Field[ Main::Result ; PCPF_OpenPDF (PDF file path) ]
#Gets the PDF form field names. You can then enter the exact form field names obtained into the
GetPDFFieldValue function as shown below
Set Field[ Form Field Names::gListOfNames ; PCPF_GetFieldNames( 0 ) ]
Set Field[ Main::FirstName ; PCPF_GetPDFFieldValue ( "SomeFormFieldName 1" ) ]
Set Field[ Main::LastName ; PCPF_GetPDFFieldValue ( "SomeFormFieldName 2" ) ]
Set Field[ Main::Company ; PCPF_GetPDFFieldValue ( "SomeFormFieldName 3" ) ]
Set Field[ Main::Date ; PCPF_GetPDFFieldValue ( "SomeFormFieldName 4" ) ]
Set Field[ Main::Email ; PCPF_GetPDFFieldValue ( "SomeFormFieldName 5" ) ]
Set Field[ Main::Phone ; PCPF_GetPDFFieldValue ( "SomeFormFieldName 6" ) ]
Set Field[ Main::Result ; PCPF_ClosePDF ]
```

## IV. Additional Functionality

### Access Text From a Page or Number of Pages

The plug-in can also access all of the text contained in a PDF document on a page by page basis. To obtain all of the text content of the currently opened PDF document call the PCPF\_GetTextFromPage function for each page in the document.

To access the number of PDF pages in the currently opened PDF document, then call the PCPF\_GetNumPages function.

### Access Document Information (metadata)

The plug-in can also access various metadata or document information contained within the PDF file. With a PDF document opened call either PCFP\_SetPDFInfo or PCPF\_GetPDFInfo depending if you desire to set or get the metadata. A list of the available metadata or information fields are:

Title, Author, Subject, Keywords, Creator, Producer, CreationDate, ModDate, or Trapped

### Combine, Delete, or Print PDF Files

The plug-in can also combine or append PDF files using the PCPF\_AppendPDF( FilePaths ) function. Before appending a PDF please ensure that the PDF is first opened and afterwards ensure that you save and close the PDF. In addition, the plug-in can delete a PDF file by calling the PCPF\_DeletePDF function and also print a PDF by calling PCPF\_PrintPDF function.

While combining PDF files together with PCPF\_AppendPDF( FilePaths ), the opened PDF file will show additional pages being appended. This is standard Adobe Acrobat behavior. Appended pages will not be saved to the opened PDF unless the PCPF\_SavePDF function is called without a specified file path. Our demo file demonstrates saving the changed PDF with the appended PDF files to a new file, through the use of PCPF\_SavePDF( FullPath ).

The accompanying FileMaker demo file also demonstrates many of these available plug-in functions in a FileMaker file. Additional Windows only functions are documented in the accompanying "Functions Guide." Please see the accompanying "Functions Guide" for a detailed description of all functions. We recommend reading all documentation, reverse engineering the scripts in the demo file and watching all videos to get a better understanding of how to use the plug-in functions.

## 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](mailto:support@productivecomputing.com)

Forum: [www.productivecomputing.com/forum](http://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](http://www.productivecomputing.com/rfq). We are ready to assist and look forward to hearing from you!