Productive Computing

Biometric Fingerprint Reader

Developer's Guide

Revised June 6, 2017

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

Table of Contents

I. Introduction

Description

The Biometric Fingerprint Reader plug-in from Productive Computing offers functions that allow you to incorporate biometric fingerprint authentication, security, and script control options in FileMaker® Pro. The plug-in is intended to be used with Digital Persona U.are.U 4500 Reader device. With this plug-in technology, FileMaker Pro can recognize and authenticate individuals based on who they are, instead of what they know (passwords and PINs) or what they posses (keys and swipe cards). 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.

Product Version History

http://www.productivecomputing.com/biometric/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: http://www.productivecomputing.com/biometric
4) Watch video tutorials: http://www.productivecomputing.com/video/?p=1236

II. Integration Steps

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

1) Installing the Plug-in with the Demo File

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.

3) Copy the actual plug-in and paste it to the Extensions folder which is inside the FileMaker program folder.
   - On Windows this is normally located here: C:\Program Files\FileMaker\FileMaker X\Extensions.

4) Start FileMaker Pro. Confirm that the plug-in has been successfully installed by navigating to "Preferences" in FileMaker, then select the "Plug-ins" tab. There you should see the plug-in listed with a corresponding check box. This indicates that you have successfully installed the plug-in.

2) Troubleshooting Plug-in Installation

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

1) Invalid Bitness of FileMaker

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

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

2) Missing Dependencies

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

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

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

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

3) Duplicate Plug-in Files

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

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

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

3) Installing the Fingerprint Device Libraries

Locate the "PCBiometric.msi" installer in your download which will be located in a folder called "Plug-in and Installer." Click on the "PCBiometric.msi" installer and follow the prompts to run the installer. During this time you will be prompted to select your FileMaker version. Please ensure that you select your correct FileMaker version that you are using such as C:\Program Files\FileMaker\FileMaker Pro X\. The installer may take a moment to run as all the UAREU and companion .dll files are being unpacked. For the curious minds, the following files are unpacked in your selected FileMaker version as shown in Figure 1.0 and 1.1 below. Please do not alter or move these files. To uninstall these files, please uninstall the "PC Biometric" from your Programs area found in the Control Panel. You may also run the installer to remove the necessary files.

Figure 1.0 - UAREUManager.dll file located in the C:\Program Files\FileMaker\FileMaker Pro X\folder

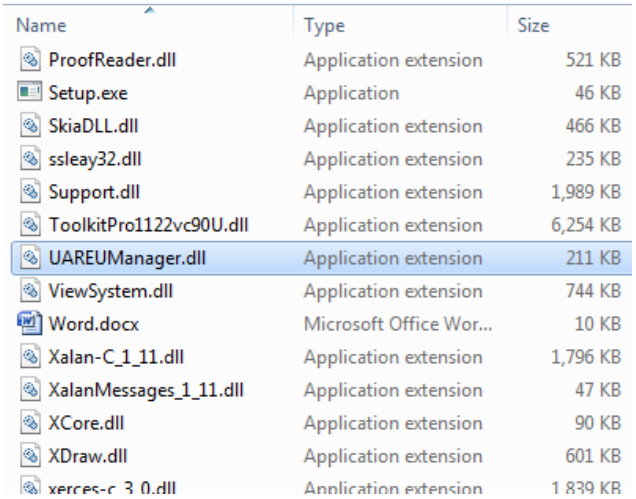
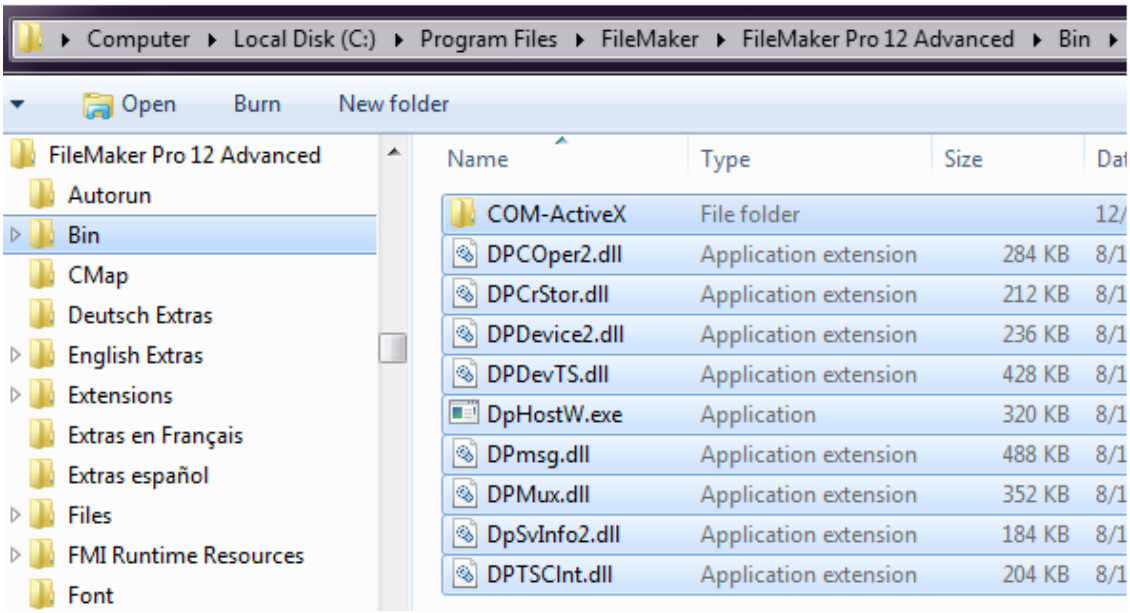


Figure 1.1 - Companion files located in the C:\Program Files\FileMaker\FileMaker Pro X\



You are now ready to connect your Biometric device to your computer.

4) Installing the Microsoft.NET 3.5 Framework

The Biometric Fingerprint Reader requires the installation of the Microsoft.NET 3.5 Framework.

This can be downloaded from Microsoft at the following link:
http://www.microsoft.com/en-us/download/details.aspx?id=21

5) 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.

6) FileMaker 16 Plug-in Script Steps

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

In this example, we use the FM Books Connector plug-in's script steps to demonstrate the difference. The same scripting differences would be found for any of the Productive Computing plug-in product lines.
For an example of using plug-in script steps, compare two versions of the same script from the FM Books Connector demo file: Pull Customer__Existing Session.Script 1 - Pull Customer__Existing Session with calculation ("traditional") plug-in scripting:

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

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

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

Exit Script [Text Result:0]
```

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

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

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

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

Exit Script [Text Result:0]
```

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

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

7) 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.

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.

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 PCFP_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 PCFP_GetOperatingMode and PCFP_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 PCFP_Register, PCFP_GetOperatingMode and PCFP_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 [ PCFP_Version( "short" ) = "" or PCFP_Version( "short" ) = "?" ]
Show Custom Dialog [ Title: "Warning"; Message: "Plug-in not installed."; Buttons: "OK" ]
If [ PCFP_GetOperatingMode ≠ "LIVE" ]
Set Field [Main::gRegResult; PCFP_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 visit our video library (http://www.productivecomputing.com/video/?p=1236) for additional setup information.

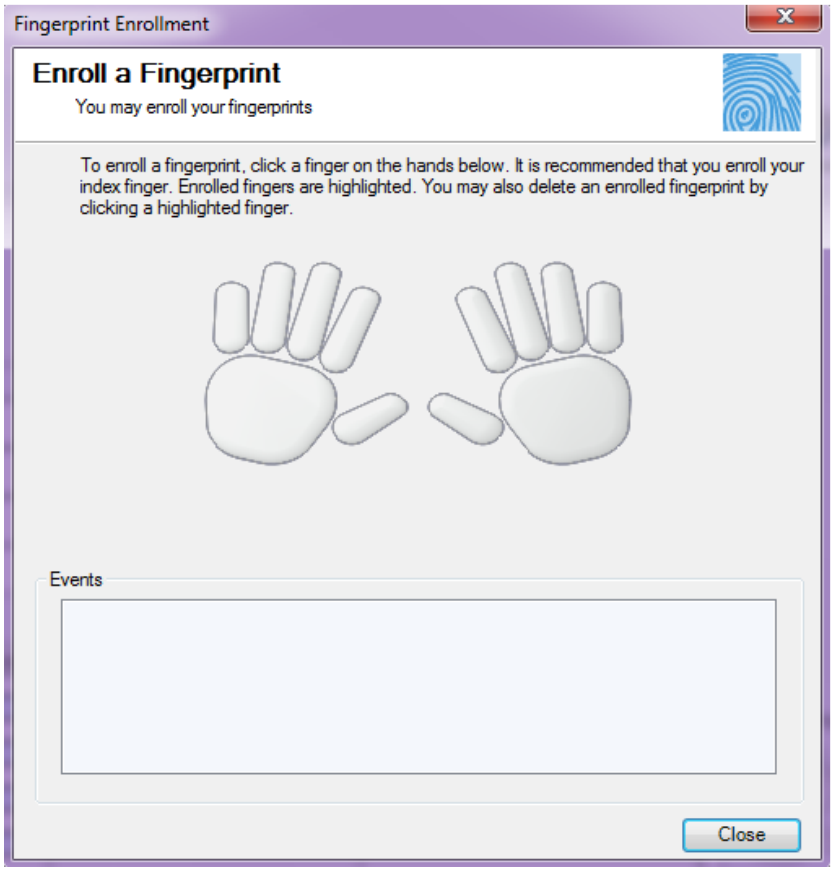
8) Initialization

You must initialize once per FileMaker session. The function used to initialize is the PCFP_Initialize(MaxFingerprints) function. This function initializes the biometric libraries required by the plug-in and the Digital Persona device manager. The MaxFingerprints parameter allows you to specify the number of fingerprints you are allowed to store in the device manager at a given time. If unsure what you should set this parameter to, then "500" is a good starting point. You can always adjust this number later if needed.

9) Enroll a User

After initialization then we are ready to start enrolling users. The PCFP_EnrollUser(UserID) function is used to enroll users. After you pass the name of the user being enrolled such as "Kendrick" or "007," then you will see the dialog prompting you to enroll fingerprints as shown in Figure 2.0. Please note that it takes about 10 seconds to display the Enrollment dialog for the first time in a FileMaker session. After the user is enrolled the scripts import the information from the plug-in and a record is created for each finger, for each user.

Figure 2.0 - Initial Fingerprint Enrollment screen

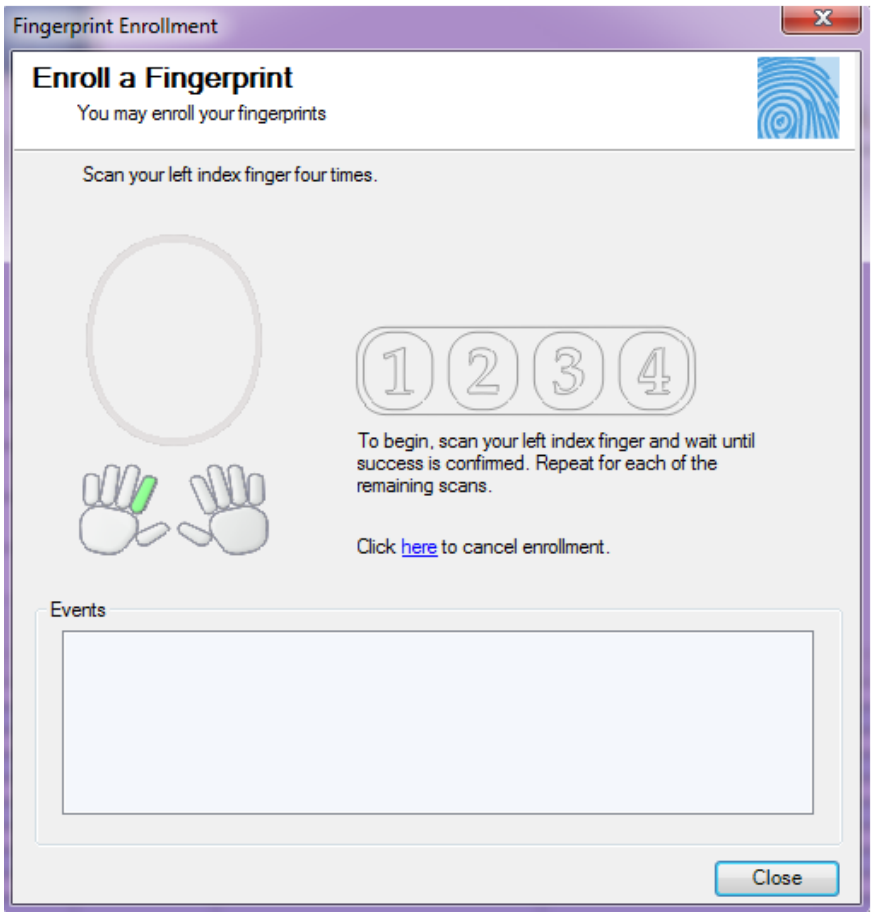


Select the finger you wish to enroll (ie. left index finger).

You will be prompted to scan your left index four times as shown in Figure 2.1.

Figure 2.1 - Fingerprint Enrollment Screen for scanning specific fingers

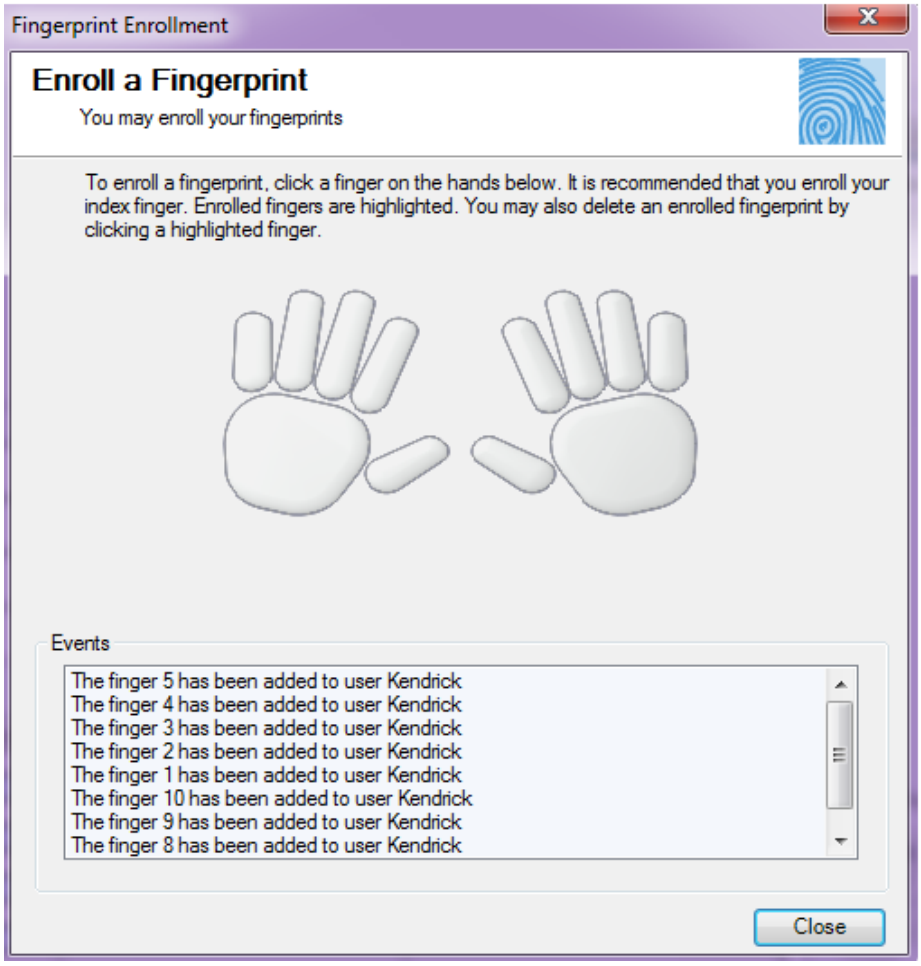
Simply place your left index finger on the DigitalPersona device four times. Afterwards you will receive confirmation as shown in Figure 2.2 that the left index (or finger 7) has been successfully added for the user. Then select the next finger to add for this user.



Note: It is recommended to enroll all ten fingers the initial time a user is enrolled. It is more complicated to add another finger to an existing user.

Figure 2.2 - Fingerprint Enrollment screen showing all fingers that have been successfully added.

When finished select the "Close" button, and you have now successfully enrolled a user and his/her desired fingerprints.



10) Finger Index Key

Now that you have enrolled a user and their fingerprints, there may be a time when you want to know what fingers you have enrolled for a user. The PCFP_GetFingersForUser(UserID) function will retrieve a string of numbers representing each finger enrolled for the given UserID.

As shown in Figure 3.0 below, the following numbers represent each finger.

- 1 represents the right thumb, 2 represents the right index finger, 3 represents the right middle finger, 4 represents the right ring finger, and 5 represents the right pinky.
- 6 represents the left thumb, 7 represents the left index finger, 8 represents the left middle finger, 9 represents the left index finger and 10 represents the left pinky.

Figure 3.0 - Finger Index Key

11) Delete a User

After you have enrolled a user, you may have a need to delete the user. The two functions available to delete users are PCFP_DeleteAllUsers and PCFP_DeleteUser(UserID). This gives you the option to delete all users or delete the user specified. Please note that when a user is deleted, so are all their corresponding fingerprints. In order to use these functions, the device manager must be initialized and the user specified must exist in the device manager.

12) Storing and Uploading Fingerprint Templates

Once you have enrolled users and their fingerprints, then the fingerprint templates can be found in the dynamic linked library (dll) for the device or in FileMaker. Depending on your solution it may be necessary to retrieve fingerprint templates from the dll into FileMaker OR to add fingerprint templates from FileMaker to the dll.

The PCFP_ImportUserFromDevice(UserID ; FingerPosition) function is used to retrieve fingerprint templates from the dll into FileMaker. Once FileMaker is turned off, then the fingerprint templates are removed from the dll. Each time you enroll a user, then it would be wise to import the fingerprint template and store the fingerprint templates in FileMaker.

The PCFP_AddFingerprintToUser(UserID ; FingerprintData ; FingerPosition) function adds the fingerprint template from FileMaker to the dll. When FileMaker is turned off, then the dll is erased and it may be necessary to add all the fingerprint templates back to the dll for use.

In order to better understand how to use these two functions, please refer to the training videos and the FileMaker Demo file as these functions are best understood in action.

13) Identify a User

Once you have successfully enrolled a user and their fingerprints, then you can identify the user giving the user access to any area of functions desired. The PCFP_Identify(optPrintCount) function is used to identify a user and the optPrintCount parameter determines how many fingerprints are required to identify a user. It takes about 10 seconds to display the Identification dialog for the first time in a FileMaker session. For example, let's say that you are using the biometric plug-in to clock employees in and out of your time clock solution. Once the employees (and their fingerprints) have been enrolled, then you can use the PCFP_Identify function to identify the employee and successfully clock the employee in or out in your solution. Since fingerprints are unique to the user, then this is a much more reliable and accurate means of identifying a user thus eliminating fraudulent activity. Perhaps you have a highly secure FileMaker solution and you want to require the user to scan 3 fingers in order to provide them access into the system, then you would pass PCFP_Identify("3") to require the user to scan 3 fingers before being granted access.

14) 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 PCFP_GetLastError("Text") function for a detailed description of what the exact error was or PCFP_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: " & PCFP_GetLastError( "Text" ) ]
End If
```

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

Value	Meaning
0	Success
-1	Plug-in not registered or session expired
-3	Invalid # of Parameters
-4	Invalid Parameter value(s)
-10	Failed Registration
-9001	Process timeout
-9002	No active passport
-9003	User does not exist in system
-9004	Unable to create id from name
-9005	SDK Error:
-9006	App Data Error:
-9007	System error:
-9008	User has fingerprint at that position
-9009	This template already exists in the database
-9012	That user already exists
-9013	You must initialize
-9014	User canceled enrollment
-9015	An instance of the UAREUManager.dll was not created

15) Known Issues

- Due to a fundamental difference in how data is handled between FileMaker and the U.ARE.U SDK, User IDs cannot be longer than 15 characters. Providing a User ID longer than 15 characters will now return an error; previously, it would return a trailing line of "junk characters."

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!