Productive Computing

eSign

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 eSign Signature Capture plug-in from Productive Computing offers functions that support capturing and rendering ESIGN / UETA compliant digital signatures. This plug-in allows FileMaker® Pro to capture signatures and bind them to information stored in your database. If the data ever changes, or if the signature is tampered with, the software renders the signature invalid. Use the eSign plug-in to decrease labor and processing time of your standard forms as well as increasing security for your organization. The current version of the plug-in is intended to be used with Topaz® 1x5 Signature Pads. With this plug-in a FileMaker user can capture and store legally binding signatures right in a FileMaker solution. 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/electronic-signatures/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

Technical Note

Congress enacted the Electronic Signatures in Global and National Commerce Act (ESIGN Act) in June of 2000. The National Conference of Commissioners on Uniform State Laws (NCCUSL) adopted the Uniform Electronic Transactions Act (UETA) in 1999. Both Acts provide a legal framework for electronic transactions. Either Act gives electronic signatures and records the same validity and enforceability as manual signatures and paper-based transactions.

Before implementing electronic signatures and transactions in your organization consult an attorney to assure compliance with any applicable laws regarding electronic signatures and electronic transactions.

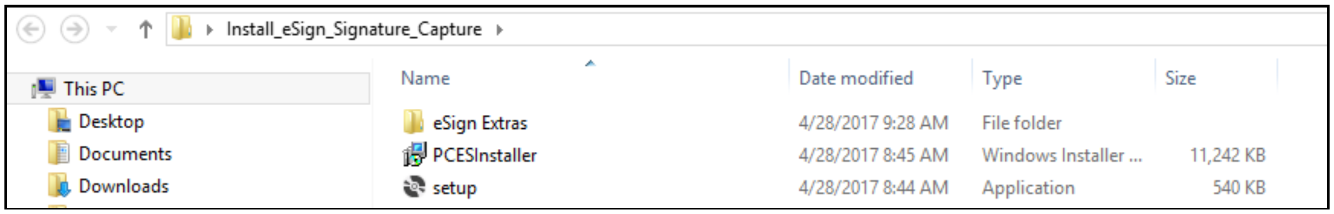
II. Integration Steps

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

1) 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.

Contents of the eSign Signature Capture installation zip package:



Name	Date modified	Type	Size
eSign Extras	4/28/2017 9:28 AM	File folder	
PCESInstaller	4/28/2017 8:45 AM	Windows Installer ...	11,242 KB
setup	4/28/2017 8:44 AM	Application	540 KB

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

1) Run the "setup.exe" file
2) If prompted, install the Visual C++ 2013 Runtime Libraries
3) If you are currently running FileMaker, please close FileMaker so that the plug-in will be installed correctly.
4) Accept the End User License Agreement ("EULA")
5) Select the location to install the plug-in*
6) Confirm the installation
7) If prompted by Windows user account control, allow the Installer to run
8) Your installation is complete!

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

2) 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, and then choose which plug-in to install.
   - "LCD" will install the "PCES_LCD8SigCapt.fmx" plug-in (for LCD Topaz device)
   - "NON-LCD" will install the "PCES_8SigCapt.fmx" plug-in (for Non-LCD Topaz device)

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
   - For LCD Topaz device install the "PCES_LCD8SigCapt.fmx" plug-in.
   - For Non-LCD Topaz device install the "PCES_8SigCapt.fmx" plug-in.

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.

3) 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.

4) Install Component for Windows 8

Installing the Microsoft Visual C++ 2013 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++ 2013 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++ 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 (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.

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

5) Registering the Plug-in

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

1) Confirm that you have access to the internet and open our FileMaker demo file, which can be found in the "FileMaker Demo File" folder in your original download.

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

3) If you are registering a licensed copy, then simply enter your license number in the "LicenseID" field and select the "Register" button. Ensure you have removed the Demo License ID and enter your registration information exactly as it appears in your confirmation email. Your plug-in should now be running in "LIVE" mode. The mode is always noted on the Setup tab of the FileMaker demo.

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

6) FileMaker 16 Plug-in Script Steps

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

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

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

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

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

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

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

Exit Script [Text Result:0]
```

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

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

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

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

Exit Script [Text Result:0]
```

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

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

7) Installing the Topaz Software and Device

Install Topaz Software and Device:

In order to capture signatures you must install the Topaz device. The software included with your Topaz device must be installed for both rendering and capturing signatures. The Topaz device does not need to be connected to the machine to render signatures. Please refer to the documentation that came with your Topaz device to install the software and properly connect the hardware.

You may manually locate your device’s SigPlus software installer online at http://www.sigpluspro.com

Which Topaz Device is Right For You?

- **Topaz T-S460-HSB (Non-LCD device)**: SigLite features all the high-quality biometric and forensic capture techniques of a SignatureGem tablet but with a low-cost touchpad and stylus in place of the active electromagnetic pen and sensor. The touchpad sensor is protected by an optional replaceable overlay for longer life. Requires the user to click an "OK" button to finish the scan process.
- **Topaz T-L462-HSB (LCD device)**: SignatureGem LCD 1X5 includes all the high-quality capture features of a Topaz electronic signature pad with the added feature of an LCD interactive display, allowing users to see "electronic ink" under the pen tip as they sign as well as navigate and display text and graphics. It provides the signor with 'Cancel' and 'OK' buttons on the device itself and does not require interaction by the FileMaker user on the computer.
- **Topaz T-LBK462-HSB (Backlit LCD device)**: SignatureGem LCD 1X5 includes all the high-quality capture features of a Topaz electronic signature pad with the added feature of an LCD interactive display, allowing users to see "electronic ink" under the pen tip as they sign as well as navigate and display text and graphics on a backlit surface. It provides the signor with 'Cancel' and 'OK' buttons on the device itself and does not require interaction by the FileMaker user on the computer.
- **Topaz T-LBK462-BSB (LCD device)**: This SignatureGem LCD 1X5 is required for eSign plug-in use in a terminal services environment. The device includes all the high-quality capture features of a Topaz electronic signature pad including an LCD interactive display, "electronic ink", text and graphic display, and is configured for ease of use in a terminal services environment.

8) Installing a BSB Device or Tablet PC License

Topaz SigPlus BSB Device Installation:

When using the LCD plug-in with a BSB device, you first must install the necessary software before use. This installer is included with your original download in a folder called "BSB Device Installer." To install the software please select the "sigplusbsb.exe" file and run the application. This must be installed on each machine using a Topaz BSB device.

Tablet PC License Installation:

Please note the Tablet PC version of the plug-in has been temporarily suspended due to compatibility issues (version 1.0.6.0+).

Plug-in versions 1.0.5.1 or below:
When using the plug-in with a Tablet PC, you must install the necessary license. This license is included with your original download in a folder called "Tablet PC License." To install this Tablet PC license please select the "sigplus374tpc.exe" file and run the application. This must be installed on each Tablet PC using the plug-in.

9) Determine Compliance with Local Laws

The eSign Signature Capture plug-in and accompanying Topaz or Tablet PC software meet federal requirements for capturing and rendering legally binding signatures. This means that when used properly the software will produce an electronic signature that meets the criteria for legally binding electronic signatures. The plug-in offers the FileMaker developer the tools to capture legally binding electronic signatures. The plug-in is not intended to be a 'total solution' for a paperless office, but rather to offer some of the tools required to create one. It is the responsibility of the developer to ensure that their electronic document storage solution will meet local requirements regarding electronic signatures, transactions, and documents.

Since the laws regulating electronic signatures, transactions, and documents vary from state to state and country to country, we recommend consulting an attorney regarding the laws in your locality. For instance, one state might allow for signatures on mortgage documents to be in electronic format and another state might require hand written 'wet ink' signatures on documents for the same purpose. Also, the data required to be stored with an electronic signatures differs from state to state. For instance, one state might require that the current date be encrypted with an electronic signature and another state may require the date to be kept separate from the signature entirely.

10) Capturing and Rendering Signatures

Now you are ready to start using the plug-in to capture and render signatures.

Option 1 - Capture Signature and Bind Signature to Data:

This is the recommended usage choice for capturing signatures as signatures are securely bound to specific data. With the plug-in installed, scan the signature and produce the image of the signature using a calculation field that calls the plug-in. This is how our demo works today. If the user changes the bound text, the signature will disappear because the original bound text is no longer validated against the original signature data. The calculation field in FileMaker is continuously checking between the bound text and the signature data using the plug-in. If either the bound data changes or the plug-in is not installed, then the signature will not render. Every user who wishes to "see" the signature must have the plug-in installed.

The one function used to capture the signature is PCES_CaptureSignature(BoundData).
When PCES_CaptureSignature is called, a signature dialog is presented to the user and the signature pad is ready to capture a signature. The signature will be bound to the 'BoundData' parameter passed to the function. If the 'BoundData' is ever changed after the signature is captured or if the signature data itself is ever modified, then the signature will be rendered invalid. In addition the rendering function will not return a proper image of the signature.

The PCES_CaptureSignature function returns an ASCII string of characters that represent a signature. The text of the document record is passed to the capturing function. The returned string will be bound to this text. This technology allows devices to capture signature information that is unique to the person signing. This information is used by analysts to determine ownership of the signature. This signature may be bound to data you have stored in your database. If the data ever changes or if the signature is tampered with, the signature is rendered invalid. Binding captured signatures to data in your FileMaker Pro database has many benefits such as authenticating content, securing unauthorized changes to documents or forms by invalidating signatures, increasing accountability and ensuring non-repudiation of transactions.

Render Signature:

The one function used to render the signature for display or printing purposes is PCES_RenderSigFromString(SigString ; BoundData). The function is passed with the signature string. This is the string returned by the capturing function and the data that is bound to that signature. If either the signature or the data bound to that signature has been modified, the literal 'UNSIGNED' string will be returned. Otherwise a bitmap of the signature will be returned and can be displayed or printed.

Option 2 - Capture Signature without Binding Signature to Data:

This option does not require the plug-in to see the signature after the first initial capture, but is also is less secure and does not bind the signature to data. You can enable multiple signatures on a document or capture a signature and store the signature in a container field.

Using a machine that has the plug-in installed, scan the signature and render the signature with a traditional calculation field that calls the plug-in. Again this is how our demo works today. Then by incorporating an additional script step (set field), you can copy the signature image from the calculation field to a regular FileMaker container field. This permanently stores the "image" of the signature without the need for the plug-in. HOWEVER, using this method does not properly bind the signature to data. In other words, if a user comes along and changes the bound text in the record, the signature will remain visible as if the record was perfectly valid when in fact it may not be.

Some users just want to capture a scanned signature for use on non-secure, non-binding documents. For this use option 2 is perfectly fine. This option does not require the plug-in for users to see a signature because the signature is permanently stored in the container field. However the plug-in is required to make the original signature capture. This option is a simple way to have a user scan and capture a signature. This option does not actively and continuously bind and check text data against signature data.

11) Known Issues

- On the Microsoft® Surface Pro Tablet PC, signatures cannot be captured using the tablet's touch screen interface. Topaz signature pads connected to the tablet still functions as normal, and can be used to capture signatures.

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!