Integration Guide

pDoc SigCapture SDK

Version 2.2
February 1, 2019

Table of Contents

1 – Introduction

pDoc SigCapture SDK provides capabilities for capturing secured biometric handwritten signatures using Signature Capture devices from Topaz Systems, Signature Capture devices from ePadLink, Windows Tablets, and GemView Tablet Displays on Windows Platforms. The signatures captured using this SDK can be used in conjunction with Topaz pDoc SigEmbed SDK to embed signatures into a PDF document. The SDK provides methods to generate the images of the captured signatures and hence can be used in any application requiring signature images too. The biometric signature data for the captured signature can be obtained in encrypted form and saved in any database for future use.

2 – Overview

- Supports capturing of biometric handwritten signatures using Topaz and ePadLink signature pads.
- Provides API for capturing handwritten signatures using Windows tablets and GemView Tablet Displays.
- Provides APIs to export the captured handwritten signature as an image along with the raw biometric data for future use, like embedding into any document.
- This SDK can be used in conjunction with the Topaz pDoc SigEmbed SDK to sign a signature field in a PDF document.

3 – Key Features

pDoc SigCapture SDK will support the following key features.

- Host Applications can use the standard control provided by the SDK for capturing and displaying signatures or they can provide host defined signature capture area handles.
- Set the signature information
- Set the signature appearance options like ink color
- Set whether smoothing for signatures is enabled or not
- Set the minimum number of signature points required to qualify a signature as valid
- Initiate signature capture
- Clear the captured signature
- Export signature image
- Export the raw biometric signature data in secured form
- Provide the connected Signature Pad information
- Check whether a signature is valid or not
- Export the captured signature data as a Topaz SigPlus compatible .SIG format
- Export the captured signature data as .SIGP format
- Check if a GemView Tablet Display is supported for signature capture
- Check if a GemView Tablet Display is connected to the PC
- Retrieve the monitor number of the GemView Tablet Display
- Specify if custom text is to be displayed in the signature image
- Specify the custom text that is to be displayed in the signature image
- Specify the percentage of the signature image that will be used for displaying for custom text

4 – Operating Systems Supported

The pDoc SigCapture SDK is supported for operating systems Windows 7 and higher, and Windows Tablets.

5 – Signature Capture Devices

The pDoc SigCapture SDK supports capturing signatures using signature capture devices from Topaz Systems and ePadLink. It also supports capturing signatures from Windows tablets and GemView Tablet Displays; to capture signatures from Windows Tablets, you must acquire a special licensed version of the SDK. Contact Topaz Systems (**www.topazsystems.com**) for further information. Signature Capture Device drivers need to be installed separately for the pDoc SigCapture SDK to capture signatures using signature pad devices.

6 – Components of pDoc SigCapture SDK

The main component of pDoc SigCapture SDK is pDoc.SigCapture.dll.

As pDoc SigCapture SDK requires supporting components installed by the device drivers for capturing signatures, the following list of files needs to be available along with the pDoc.SigCapture.dll in the same folder for integrating pDoc SigCapture SDK into any application.

1. AxInterop.EpadInkLib.dll
2. Interop.EpadInkLib.dll
3. AxInterop.GIIEPADLib.dll
4. Interop.GIIEPADLib.dll
5. AxInterop.SIGPLUSLib.dll
6. Interop.SIGPLUSLib.dll
7. SigUtil.dll
8. ePadService.dll (This file is required only if you do not have ePadLink drivers installed in the machine and are using either Topaz Signature Pad or a Windows Tablet to capture the signatures)
9. WintabDotNet.dll
10. Microsoft.Ink.dll
11. pDocSDKConfig.enc

Microsoft Visual Studio 2008 with .Net framework 3.5 is used as the development IDE for developing pDoc.SigCapture.dll and hence .NET framework 3.5 should be available in the end user machine for pDoc SigCapture SDK to work properly.

7 – Classes

The primary output component of this SDK will be pDoc.SigCapture.dll; following are the classes contained in this library. Properties and methods of each class are explained below their respective classes.

7.1 Capture

Namespace
pDoc.SigCapture

Syntax
public class Capture

7.1.1 Properties

	Name	Syntax	Description
	SigViewer	IntPtr SigViewer { set; }	Specifies the host defined signature capture area handle. The scribblings on the device are displayed on this area. Currently this is available only for devices from ePadLink and for Windows Tablets. Not applicable for Topaz Signature Pads.
	Enable Smoothing	bool EnableSmoothing { set; }	Specifies if signature smoothing is enabled or not.
	MinSigPoints	int MinSigPoints { set; }	Specifies the minimum number of points for signature to be valid.
	IsSignature	bool IsSignature { get; }	Specifies if the captured signature is acceptable or not based on the Minimum Number of Signature Points set. True if the captured signature is valid.
	InkColor	Color InkColor { set; }	Specifies the ink color for the signature
	InkThickness	short InkThickness { set; }	Specifies the ink thickness for signature
	PadName	string PadName { get; }	Returns the name of the connected signature pad.

	PadSerialNumber	string PadSerialNumber { get; }	Returns the Serial Number of the connected signature pad if the pad has one.
	bReqSerialNumber	bool PadSerialNumber { set; }	Specifies if calling GetDeviceInfo method should read the serial number of the connected device. Applicable only for ePadLink Signature Pads. For Topaz Signature Pads, Windows Tablets and GemView Tablet Display devices, the connected signature pad Serial Number is returned even without setting this property.
	ExternalDeviceCheck	bool ExternalDeviceCheck { set; }	Specifies if external signature capture devices should be enabled for signing or not. This property is useful when using a GemView Tablet Display for singing. Setting this to true will avoid checking for external signature pads and speed up the connection opening process with the GemView Tablet Display.
	ErrorMessage	string ErrorMessage { get; }	The message that describes the last error.
	UseTabletDisplay	bool UseTabletDisplay { set; }	If the application has previously detected that a GemView Tablet Display is connected to the PC and to specify that the signature capture should not check the GemView Tablet Display existence again (to speed up the signing experience), set this property to true.
	TabletDisplayName	string TabletDisplayName { set; }	If the UseTabletDisplay property is set to true, the TabletDisplayName property should be set too. When the application has detected the presence of the GemView Tablet Display, it can get the name of the Tablet Display.
	TabletDisplaySerialNumber	string TabletDisplaySerialNumber { set; }	If the UseTabletDisplay property is set to true, the TabletDisplaySerialNumber property should be set too. When the application has detected the presence of the GemView Tablet Display, it can get the Serial Number of the Tablet Display.
	DisplayXLine	bool DisplayXLine { set; }	Specifies if a line with X in the beginning should be displayed in the signature capture or not. Set it to true to display the X and the line. This is applicable for ePadLink pads, GemView Tablet Displays, and Windows Tablets.

	DisplayCustomText	bool DisplayCustomText { set; }	Specifies if display of custom text in the signature image is enabled or not. This property should be set before calling GetSignatureImage method. By default this value is set to False.
	CustomTextPercent	int CustomTextPercent { set; }	If the DisplayCustomText property is set to true, the CustomTextPercent property specifies the percent of signature image that should be used for displaying the custom text. The minimum percent of signature image that is allocated for custom text should be between 20 and 50. Any value less than 20 is treated as 20 and value greater than 50 will be treated as 50. This property should be set before calling GetSignatureImage method. If the percent is set to be between 20 and 50 and if the custom text specified does not fit in the space available, the text would be omitted and only the squiggle is drawn on the image.
	CustomTextLine1	string CustomTextLine1 { set; }	If the DisplayCustomText property is set to true, the CustomTextLine1 property specifies the first line of custom text to be displayed in the signature image. This property should be set before calling GetSignatureImage method.
	CustomTextLine2	string CustomTextLine2 { set; }	If the DisplayCustomText property is set to true, the CustomTextLine2 property specifies the second line of custom text to be displayed in the signature image. This property should be set before calling GetSignatureImage method.

The special version of pDoc SigCapture SDK for Windows Tablets does not support GemView Tablet Displays and hence the following properties (described above) are not applicable.

- bReqSerialNumber
- ExternalDeviceCheck
- UseTabletDisplay
- TabletDisplayName
- TabletDisplaySerialNumber

7.1.2 Methods

7.1.2.1 SetSignatureInfo

Method Description
Sets the signature information, which will be persisted into the raw biometric signature data.

Syntax
void SetSignatureInfo(string FirstName, string LastName, string EMailID, string Location);

Parameters

Parameter Name	Type	Description
FirstName	String	First Name
LastName	String	Last Name
EmailID	String	Email ID
Location	String	Geographical Location

7.1.2.2 StartSign

Method Description
Initiates the signature capture

Syntax
bool StartSign();

Return Value
Returns “True” if initiation of the signature capture is successful and “False” if failed.

7.1.2.3 ClearSign

Method Description
Clears the currently capture signature

Syntax
void ClearSign();

7.1.2.4 CloseConnection

Method Description
Closes the connection with the connected signature capture device and ends the signature capture

Syntax
void CloseConnection();

7.1.2.5 GetSignatureImage

Method Description
Returns the image (BMP, JPG or PNG) of the captured signature in the form of Base64 string

Syntax
string GetSignatureImage(int ImageFormat, Color BkColor, int iWidth, int iHeight, bool bCenter, bool bTransparency, bool bMaxUpScaleFactor, float fUpScalePercent);

Parameters

Parameter Name	Type	Description
ImageFormat	Int	Image Format (i.e. BMP, JPEG or PNG) 0 – BMP 1 – JPEG 2 - PNG
BkColor	Color	Background color for the image
iWidth	int	Width of the required image
iHeight	Int	Height of the required image
bCenter	Boolean	Specifies if the signature should be centered in the image
bTransparency	Boolean	Specifies if the signature image should be transparent. Applicable only for PNG.
bMaxUpScaleFactor	Boolean	Specifies if the Maximum Upscale Factor has to be applied while scaling up signature for display.
fUpScalePercent	Float	Specifies the percentage of the target image to be used while generating the image if the size of the signature squiggle captured is smaller than the specified image size

Return Value
Base64 encoded string containing the image binary data

7.1.2.6 GetSignatureData

Method Description
Returns the encrypted raw biometric signature data in the form of Base64 string

Syntax
string GetSignatureData();

Return Value
Base64 encoded string containing the encrypted biometric data.

7.1.2.7 GetSigString

Method Description
Returns the signature data as string that is compatible with Topaz's SigPlus SigString format

Syntax
string GetSigString();

Return Value
String containing the signature data.

7.1.2.8 GetDeviceInfo

Method Description
Retrieves the information of the connected signature capture device. Before calling the PadName and PadSerialNumber properties, this method needs to be invoked.

Syntax
```
void GetDeviceInfo();
```

7.1.2.9 SaveAsSIGFile

Method Description
Saves the captured signature into a Topaz SigPlus compatible Signature (.SIG) file. One needs to call GetSignatureData method before saving the signature as a SIG file.

Syntax
```
bool SaveAsSIGFile(string strSigFileName);
```

Return Value
Boolean indicating if the File was generated successfully or not.

7.1.2.10 SaveAsSIGPFile

Method Description
Saves the captured signature into a .SIGP file. One needs to call GetSignatureData method before saving the signature as a SIGP file. In addition to the information in the SIG file, this file contains information about pressure for each point and also the device that is used to capture the signature.

Syntax
```
bool SaveAsSIGPFile(string strSigPFileName);
```

Return Value
Boolean indicating if the File was generated successfully or not.

7.2 TabletDisplay

Namespace
pDoc.SigCapture

Syntax
```
public class TabletDisplay
```

7.2.1 Methods

7.2.1.1 IsTabletDisplaySupported

Method Description
Checks whether using GemView Tablet Display for signature capture is supported or not in this version of the SDK.

Syntax
```
bool IsTabletDisplaySupported();
```

Return Value
Returns “True” if GemView Tablet Display is supported and “False” if not supported.

7.2.1.2 IsTabletDisplayConnected

Method Description
Checks whether GemView Tablet Display is connected to the PC or not.

Syntax
```
bool IsTabletDisplayConnected();
```

Return Value
Returns “True” if GemView Tablet Display is connected and “False” if not connected.

7.2.1.3 GetTabletDisplayMonitorNumber

Method Description
Retrieves the monitor number of the GemView Tablet Display (if connected) as set in the Tablet Display Manager.

Syntax
```
int GetTabletDisplayMonitorNumber();
```

Return Value
Returns the monitor number.

8 – Integration with Host Application

8.1 Using Standard Capture Control

This section explains how the pDoc SigCapture SDK can be integrated with a host application. This section assumes that pDoc SigCapture SDK along with the required signature capture device drivers are already installed. Microsoft Visual Studio 2008 is used as the reference IDE for explaining the integration.

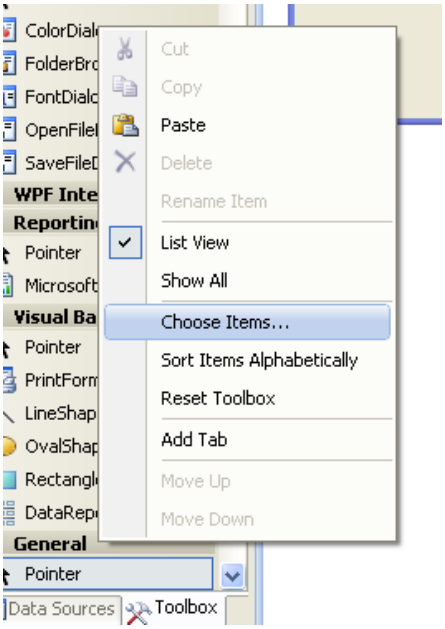
1. Open Microsoft Visual Studio 2008.

2. Click on File → New Project and under Visual C#, select Windows Forms Application.

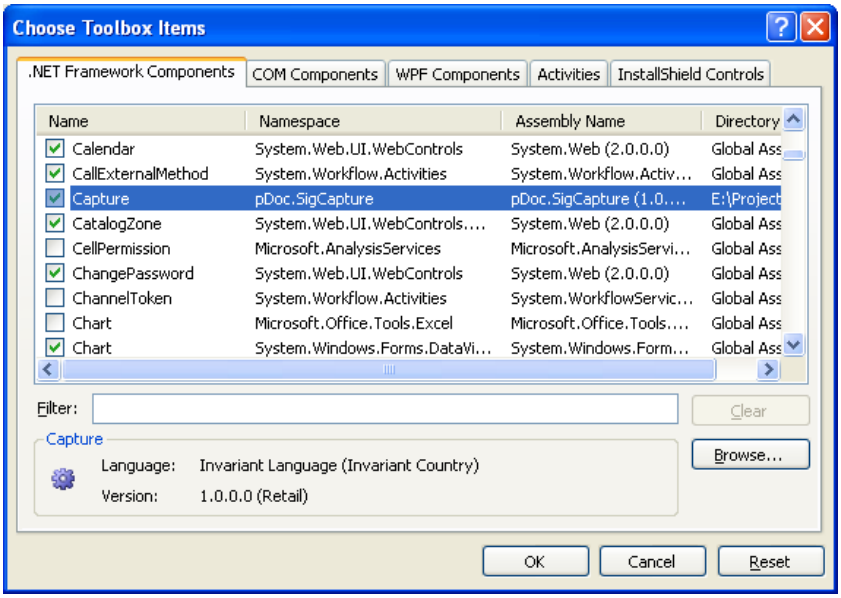
3. Give a name for the project (e.g. SigCaptureSDKSample).

4. Change the name of the Form1 to Sample (i.e. Sample.cs).
5. If the application has to capture user information, one can insert labels and text boxes to capture the relevant information.

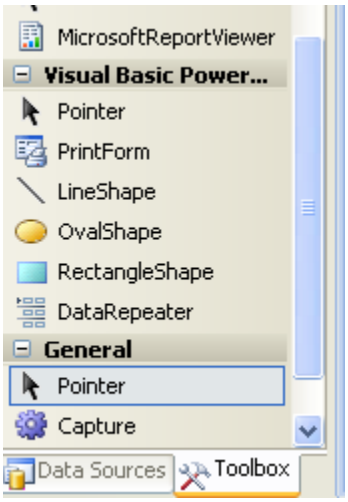
6. Right click on the “General” section of the Toolbox and select “Choose Items” as shown below.



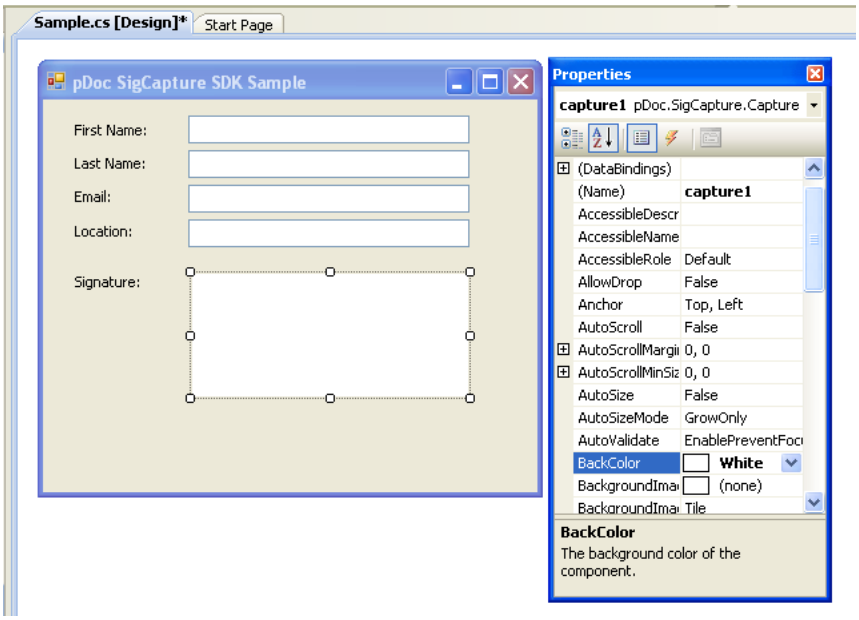
7. In the opened “Choose Toolbox Items” window, under “.NET Framework Components”, click on “Browse” and select the “pDoc.SigCapture.dll” provided to you as part of the SDK and then make sure that the component “Capture” is in selected mode as shown below.



8. Click on the “OK” button. The Capture Control will now be displayed in the “General” section of the “Toolbox” as shown below.

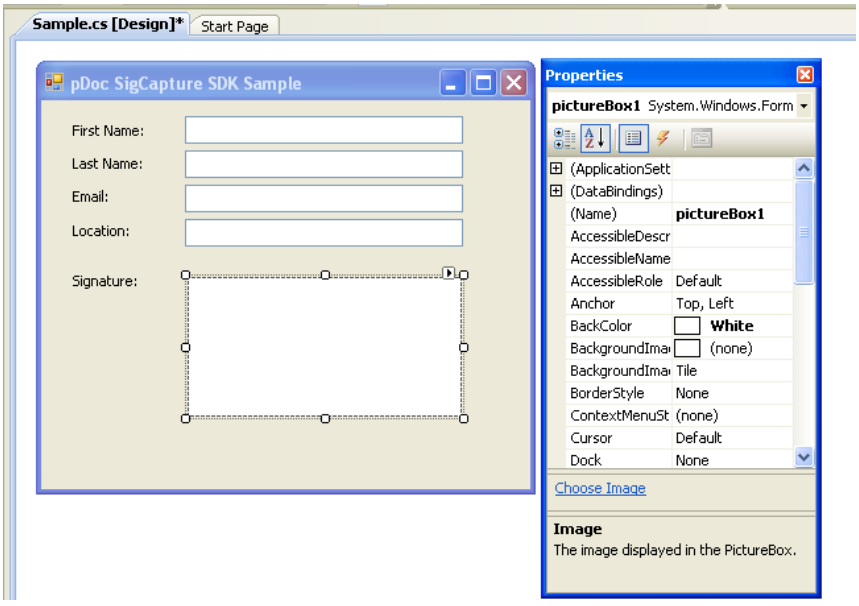


9. Select Capture and Drag it onto the Form. Set the BackColor of the Capture object to White if needed. Now the Form looks like that shown below.

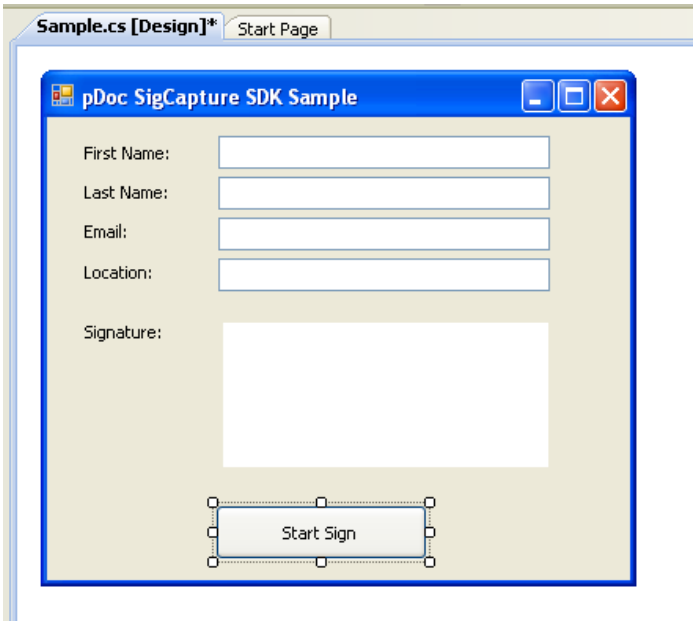


8.2 Using Host Provided Control

1. Alternatively, instead of a standard SDK capture component, host provided control (Picture Box or any other native .NET Framework Control for) can also be used to capture a signature and display the same (applicable only to applications using ePadLink signature capture devices and Windows Tablets). The form below is a variant of the above form containing a .NET Picture Box in the place of the Capture SDK Control. In this case, the scribbling on the signature capture device is reflected in the Picture Box.



2. Now add a button named “Start Sign” to the application.



If using the Capture Control from the pDoc SigCapture SDK, the button code will be as below.

```
private void btnStartSign_Click(object sender, EventArgs e)
{
    capture1.SetSignatureInfo(txtFirstName.Text, txtLastName.Text, txtEmail.Text,
txtLocation.Text);
    capture1.StartSign();
}
```

If using a Picture Box control from the .NET Framework, the button code will be as below.

```
private void btnStartSign_Click(object sender, EventArgs e)
{
    capture1.SetSignatureInfo(txtFirstName.Text, txtLastName.Text, txtEmail.Text,
txtLocation.Text);
    capture1.SigViewer = picturebox1.Handle;
    capture1.StartSign();
}
```

For the usage of other methods and properties of pDoc SigCapture SDK, refer to the Sample Application (along with the source code) provided along with the SDK.