IntegriSign ePad Ink SDK (Java)

Developer's Guide

C O N T E N T S

1. Introduction

IntegriSign is a handwritten signature Capture and Verification System, using ePad device. IntegriSign ePad Ink SDK (Java) is bundled with capture components. Signatures are captured and data is presented as a BASE64 string, facilitating the host application to store in any database/file system. Flexible and yet powerful API calls are provided to convert and optimize captured signatures to various image formats like BMP, JPEG, PNG and GIF. It has also got the APIs to leverage the ePad Ink LCD and Bi-directional capability. Using the ePad Ink specific calls one can push the text to the device and capture the user response.

2. Abstract

Desktop signature capture system, developed using IntegriSign Java Developer's Kit, is described. This application allows users to sign and authenticate java forms, with the ability to store and retrieve signatures in the server database as well as the ability to generate image files of the signature. Also has the ability to expose the ePad Ink functionality.

3. Key Features

- Option of converting signature data into image formats (JPEG, GIF, BMP and PNG).
- Allows signature capture through any ePad device.
- Enables the verification of content integrity of signed documents or forms.
- Has the additional APIs to leverage the ePad Ink Bi-directional capabilities.

4. System requirements

Client: Windows 2003/Windows 2008/Windows 2012/Windows 7/Windows 8/Windows 8.1/Windows 10 (all 64-bit) operating systems

JDK 1.4 RUNTIME or higher

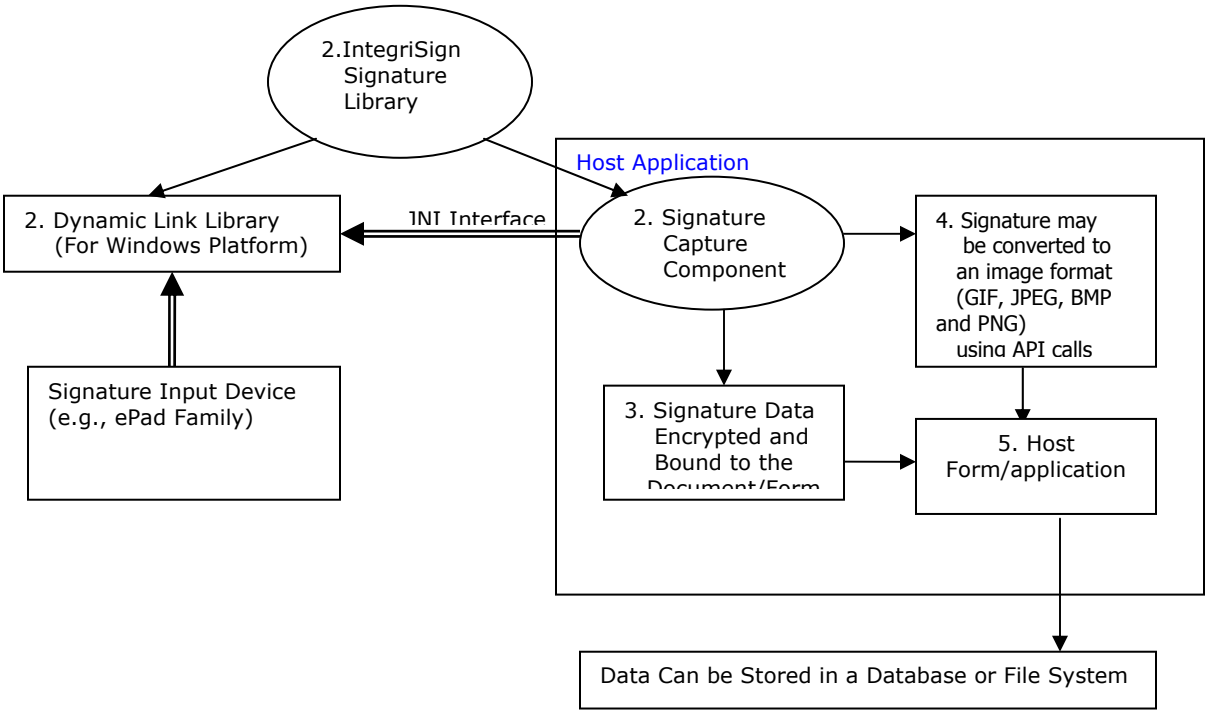
5. Process

(Steps are illustrated in the flow diagram below)

1. **Installation:** The IntegriSign Desktop Library along with the support dlls loaded onto the client machine.
2. **Client setup:** JNI calls are used for signature input from ePad devices.
3. **Signature capture and processing**: The signature capture component calls the Windows DLLs through the Windows JNI (Java Native Interface) and gets the signature data from the input device. The input device is any ePad. The raw data captured from the input device is processed and encrypted using the IntegriSign client library and generated in a BASE64 string format, along with content hash generated by an MD5 hashing algorithm, which binds the signature to the content. This makes it possible for the signature data to be stored in any database or transmitted through any kind of network/protocol.

4. **Conversion of signature as an Image:** The developer has the option of converting the signature data into an image format (JPEG, GIF, BMP and PNG) using API calls.

5. **Assign signature to form:** The signature data (BASE64 string) is assigned to one of the fields of Host application.
6. **Storage:** The signature data along with the rest of the application data is stored in database server.



6. Running IntegriSign enabled Sample

Running the simple capture application

1. Copy the 'IntegriSign' folder/package from SDK folder to a directory where class path is set to JVM.
2. Copy 'eSJDeskPad.dll', 'JMsgBox.dll', 'ESUtil.dll' to windows system folder (e.g. c:\windows\system or c:\winnt\system32).
3. Run the DeskSignTest.class from the 'Samples/SimpleCapture' folder.

Running the ePad Ink application

1. Copy the 'IntegriSign' folder/package from SDK folder to a directory where class path is set to JVM.
2. Copy 'eSJDeskPad.dll', 'JMsgBox.dll', 'ESUtil.dll' to windows system folder (e.g. c:\windows\system or c:\winnt\system32).
3. Also copy the ePadInkNative.dll to the windows system folder.

4. Copy the ePadInkCtrl.dll onto any of the folders and register the dll.
5. Run the ePadInkTest.java from 'Samples/ePadInk' folder.

The Sample Capture application demonstrates

- Signature capture (Signature can be captured after entering userid and message. userid will be used to store the images with that name. user name and Message are used for hashing.)

- Content binding to the signature. (The data entered in the user name and message field is bound to the signature. Clicking VerifyIntegrity button after the act of signing can check content integrity status. Data can be altered to cross check the hash).

- Generating GIF,JPEG,BMP and PNG images from the captured signatures.(When SaveImages button is pressed image files will be written to the application directory with the userid entered as file name.

- Opening and displaying signature from already persisted signature data.(After the act of signing click GetSignData button to get the data, then click clearsign to clear the signature. Finally press openSign button to redisplay the signature. After opening the signatures verifyintegrity can be checked.

- Clearing the signature (By clicking clearsign button clears the signature on the form.

- GetSignData (This demonstrates how to get the encrypted Base64 signature string)

The ePad Ink sample demonstrates

- Demonstrates the ePad Ink Bi-Directional capability by sending a sample agreement text to the device. Signature is captured at the end based on the user input. During the process of displaying the text on the device the component locks the application, during that time if the application is required to be closed then one needs to click on the **X** displayed on the top right corner of the device
- Simple Signature capture

JAVA Run time 1.2 or higher is required to run these samples.

7. Customizing applications to use IntegriSign

Follow these steps to Customize applications to use IntegriSign.

1. Import integrisign package into the host application (drag the integrisign bean on to application if integrisign Bean is added to the IDE).

2. Create DeskSign object and add it to the host application, in case import integrisign is used.

3. Host application must implement the **integrisign.IdocInfo** interface.

4. Refer to the sample applications shipped for more help on how to implement the all the functions of **IdocInfo interface.**

5. In case data binding is not required host application must implement the **IdocInfo** interface and provide null implementation to all the methods.

8. API Model

8.1 integrisign package

integrisign package is collection API calls for capture of the signature.

8.1.1 integrisign.IdocInfo

public interface **IdocInfo**

Description

This interface defines methods that needs to be implemented by the host application to bind the content to the signature that is being captured. Incase no hashing is required all the methods should be implemented as shown in the sample

```
public void feedGrabber (integrisign.IGrabber ig){


}
public byte getVersion(){
        return 1;
}
public String getDocID(){
        return "";
}
```

8.1.2 integrisign.desktop.DeskSign

public class **DeskSign** extends **Jcomponent** implements **integrisign.IGrabber**

Description

DeskSign component is responsible for capturing the Signature dynamics, and processes the signature data and paints the signature on the host form. This component internally uses IntegriSign SDK components for capture and has methods for signing and also to get the Signature info as a BASE64 format string by which the sign info can be easily stored in a database or transmitted via different networks. This class is also capable of generating different image formats (BMP, GIF, JPEG and PNG) from the captured signature. Captured and stored signatures can be opened with this class. Once the signature is

captured or opened using existing signature info, user details like Name, Designation, Organization, Address can be queried if they are set.

DeskSign component is also capable of generating MD5 hash of the content to be authenticated optionally. The host Application should implement **integrisign.IdocInfo** interface. Even if no hashing is required the above interface should be implemented and feedGrabber() method can be null Implemented.

This Component class is shipped as a part of SDK along with supporting windows dlls.

The component background can be set to be transparent using the setOpaque method. This function sets the transparency if the opaqueness is set to false.

8.1.3 Methods

- public void **clear ()**

  This method clears the current signature on the component and reinitializes the component.

  Arguments

  None

  Return value

  This method returns void.

- public String **getString** ()

  This method is for getting the signature info as encrypted BASE64 String. This method should be called if the isSigned() method returns true.

  Arguments

  None

  Return value

  Returns the signature information as encrypted BASE64 String.

- public boolean **isSigned** ()

  This method returns the status of signing as a boolean.

  Arguments

None

Return value

Returns boolean value depending on whether the form is signed or not. Returns true if the Signature is done otherwise returns false.

- Public void **openSign1** (String signstring)

  This method facilitates to open and display previously captured and stored signatures.

  Arguments

  signstring: The raw signature information as a BASE64 String.

  Return value

  This method returns void.

- Public String **getGifString (**string signstr, int width, int height**);**

  This method returns the gif information as BASE64 String from the raw signature.

  Arguments

  signstr:Signature string in BASE64 Format.

  width: Width of the GIF image to be returned.

  height: Height of the GIF image to be returned.

  Return value

  This method returns Gif Image data as a BASE64 String, which can again be converted to byte array using Base64 class.

- Public String **getJpegString (**string signstr, int width, int height, int quality**);**

  This method returns the jpeg bytes as an array.

  Arguments

  signstr: Signature string in BASE64 Format.

  width: Width of the JPEG image to be returned.

height: Height of the JPEG image to be returned.

Quality: Quality of the JPEG image to be returned on a scale of 1 to 100 (100 being best quality)

Return value

This method returns JPEG Image data as a BASE64 String, which can again be converted to byte array using Base64 class.

- Public String **getBMPString (**string signstr, int width, int height**);**

  This method returns the BMP bytes as an array.

  Arguments

  signstr: Signature string in BASE64 Format.

  width: Width of the BMP image to be returned.

  height: Height of the BMP image to be returned.

  Return value

  This method returns BMP Image data as a BASE64 String, which can again be converted to byte array using Base64 class.

- Public String **getPNGString (**string signstr, int width, int height, Boolean transparency**);**

  This method returns the PNG image information as BASE64 String from the raw signature.

  Arguments

  signstr: Signature string in BASE64 Format.

  width: Width of the PNG image to be returned.

  height: Height of the PNG image to be returned.

  Transparency: Image background transparency. If set to true, the image generated will have the transparent background.

  Return value

  This method returns PNG Image data as a BASE64 String, which can again be converted to byte array using Base64 class.

- public String **getSignerName ()**

  Arguments

  None

  Return Value

  This method returns the Signer Name bound to the signature.

- public String **getSignerDesignation ()**

  Arguments

  None

  Return Value

  This method returns the Signer designation bound to the signature, if it was set.

- public String **getSignerOrganisation ()**

  Arguments

  None

  Return Value

  This method returns the Signer Organisation bound to the signature, if it was set.

- public String **getSignerAddress ()**

  Arguments

  None

  Return Value

  This method returns the Signer Address bound to the signature, if it was set.

- public void **setSignThickness**(byte signthickness)

  This function sets the thickness to be used while drawing the signature trace on the component as well as during the image generation. This can be set prior to signNow and image generation calls. The thickness can be in the range of 1 – 9. Any value beyond these will be set to the default value of 2.

Arguments

Byte : The pen thickness to be used while drawing the signature. The value should be in the range of 1 -9.

Return Value

This method returns void.

SignNow Description

The signNowEx method is overloaded, depending on the information needs to bind to the Signature object, one of the signNowEx methods can be chosen. When SignNowEx is called a SignPad appears for capturing the signature. User can choose a different signature color from options button, depending on the privileges set to him in signNowEx method.

These methods allow the developer to choose different options like

A. Whether signature options can be enabled/disabled?

Set the showOptions to true/false depending on whether to the allow user to choose options or not.

NOTE: It is recommended to use the signNowEx method instead of the signNow method. signNow methods are maintained to support the legacy applications.

- public void **signNowEx** (integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface and pass empty bytes to grabBytes () method called inside the **feedGrabber ()** method of IdocInfo.

  Return Value

  This method returns void.

  Note: In case of this method all the Signer Details are set to none.

- public void **signNowEx** (string name, string designation, string organisation, string address, string footer, showOptions, integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  name: Name of the Signer. It cannot be empty.

designation: Designation of the Signer.

organization: Organisation of the Signer.

address: Address of the Signer.

footer: Any other information of the Signer like phone no., e-mail id etc…

showOptions: Whether the Options button needs to be enabled or not, to set the signing options like choosing the color.

idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

Return Value

This method returns void.

- public void **signNowEx** (string ID, string name, boolean showOptions, integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  ID: ID of the signer specific to implementation. Cannot be empty.

  name: Name of the Signer. Cannot be empty.

  showOptions: Whether the Options button needs to be enabled or not, to set the signing options like choosing the color.

  idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

  Return Value

  This method returns void.

- public void **signNowEx** (string ID, string name, string desg, string org, string Addr, string Footer, boolean showOptions, integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  ID: ID of the signer specific to implementation. Cannot be empty.

  name: Name of the Signer. Cannot be empty.

  designation: Designation of the Signer.

organization: Organisation of the Signer.

address: Address of the Signer.

footer: Any other information of the Signer like phone no., e-mail id etc…

showOptions: Whether the Options button needs to be enabled or not, to set the signing options like choosing the color.

idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

Return Value

This method returns void.

- public void **signNow** (string name, string designation, string organisation, string address, string footer, string reserved, boolean breserved1, boolean showOptions, boolean breserved2, integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  name: Name of the Signer. It cannot be empty.

  designation: Designation of the Signer.

  organization: Organisation of the Signer.

  address: Address of the Signer.

  footer: Any other information of the Signer like phone no., e-mail id etc…

  reserved: reserved parameter

  breserved1 : reserved parameter

  showOptions: Whether the Options button needs to be enabled or not, to set the signing options like choosing the color.

  bresered2: reserved parameter.

  idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

  Return Value

  This method returns void.

- public void **signNow** (string ID, string name, boolean breserved1, boolean showOptions, boolean breserved2, integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  ID: ID of the signer specific to implementation. Cannot be empty.

  name: Name of the Signer. Cannot be empty.

  breserved1 : reserved parameter

  showOptions: Whether the Options button needs to be enabled or not, to set the signing options like choosing the color.

  breserved2: reserved parameter.

  idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

  Return Value

  This method returns void.

- public void **signNow** (string ID, string name, string desg, string org, string Addr, string Footer, string reserved, boolean breserved1, boolean showOptions, boolean breserved2, integrisign.IdocInfo idinfo) throws ValidationException

  Arguments

  ID: ID of the signer specific to implementation. Cannot be empty.

  name: Name of the Signer. Cannot be empty.

  designation: Designation of the Signer.

  organization: Organisation of the Signer.

  address: Address of the Signer.

  footer: Any other information of the Signer like phone no., e-mail id etc…

  reserved: reserved parameter

  breserved1 : reserved parameter

  showOptions: Whether the Options button needs to be enabled or not, to set the signing options like choosing the color.

breserved2: reserved parameter.

idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

Return Value

This method returns void.

- public int **verifyDocument** (integrisign.IdocInfo idinfo)

  This method checks the integrity of the content that is bound to the signature object and returns the status as an integer.

  Note: This method should be called when GetHashData () is implemented in the client script.

  Arguments

  idinfo: The host application must implement the **integrisign.IdocInfo** interface as shown in the sample. If nothing has to be bound to signature implement the interface grabBytes() method can be null implemented.

  Return Value

  This method returns an integer specifying whether the file is tampered with or not

  0 ----- Success, contents not changed

  1 ----- Failure, contents have changed

  2 ----- Hash generated with different version

  3 ----- Photographing information is not stored

  4 ----- No sign string available

  THE FOLLOWING THREE FUNCTIONS CAN ONLY BE USED WITH THE ePad INK DEVICE.

- Public int **openConnection**()

  Activates ePad-Ink and establishes the connection between the application and the ePad-Ink device.

  Arguments
  None

  Return value

Returns 0 if opening Connection with the device fails and 1 if connection is successful.

- Public int **closeConnection**()

  Closes the connection between the application and the Ink Pad device.

  Arguments
  None

  Return value
  Returns 0 if opening Connection with the device fails and 1 if connection is successful.

- Public int **showMessage**(String message, int BUTTON_STYLE)

  This method allows you to create screens on the ePad Ink device based on predefined templates where you can specify type of the button text to be displayed. This function returns the user event as integer. This can be used by the application for necessary action.

  When this function is invoked a screen is displayed on the device and the user can't access the host application. Incase if the user wants to cancel the transaction and access the host application the user has to click on the **X** button displayed on the top right corner of the ePad Ink Device.

  Arguments

  Message –Text to be displayed on the ePad-Ink device as String.

  BUTTON_STYLE – Type of button to be displayed along with the text on the device as Integer.

  Available Button Types:

  BT_OK = 0,
  BT_OK_CANCEL = 1,
  BT_OK_CLEAR = 2,
  BT_OK_CANCEL_CLEAR = 3,
  BT_YES_NO = 4,
  BT_YES_NO_CANCEL = 5,
  BT_NEXT = 6,
  BT_PREV_NEXT = 7,
  BT_PREV_NEXT_CLEAR = 8,
  BT_ACCEPT_DECLINE = 9,
  BT_NONE = 10

  Return value

  Returns the user event (i.e. the button click as integer)

  the EventIDs returned when user clicks on a button.

  OK =1
  CANCEL = 2

CLEAR	= 3
YES	= 4
NO	= 5
PREV	= 6
NEXT	= 7
ACCEPT	= 8
DECLINE	= 9
CLOSE (**X**)*	= 99

* If the user chooses to close the connection with the device by clicking on the X on the top right corner of the device then an event id 99 is returned.

8.1.5 integrisign.desktop.Base64Format

public class **Base64Format**

This class is mainly for the purpose of encoding and decoding the information into a string from a byte array and vice versa.

Constructor Index

Public **Base64Format** ()

Method Index

- Public String **encode64** (byte[] barray)

  This method encodes the passed byte array into a BASE64 String.

  Arguments

  barray: any array of bytes

  Return value

  This Method returns a BASE64 String.

- Public byte[] **decode64** (string str)

  This method decodes the passed BASE64 String into a byte array.

  Arguments

  str: any BASE64 String

  Return value

  This Method returns a decoded byte[] corresponding to the string passed.