# ePadLink®

**WebSign SDK (Java)**
**Java (Client) – ASP (Server)**
**Developer's Guide**

# ePadLink®

**WebSign SDK Java Client ASP Server Developer's Guide**

## Table of Contents

WebSign SDK Java Client ASP Server Developer's Guide

# 1.0 – Introduction

IntegriSign is a handwritten signature Capture and Verification System, using any ePad device. WebSign SDK (Java) is bundled with client capture components and server side API. Client components are developed as distribution units, which get installed automatically and are supported on IE. Signatures are captured and data is presented as a BASE64 string, facilitating the host application to transport using HTTP/HTTPS protocols and 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.

# 2.0 – Abstract

An Internet Browser-enabled signature capture system, developed using IntegriSign WebSign Developer's Kit, is described. This application allows users across the organization to sign and authenticate Web 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.

# 3.0– Key Features

1. Allows users to sign from any location.
2. Option of converting signature data into image formats (JPEG, GIF, BMP and PNG).
3. Allows signature capture through ePad device.
4. Enables the verification of content integrity of signed documents or forms.
5. No manual installation or enabling of client desktop is required.
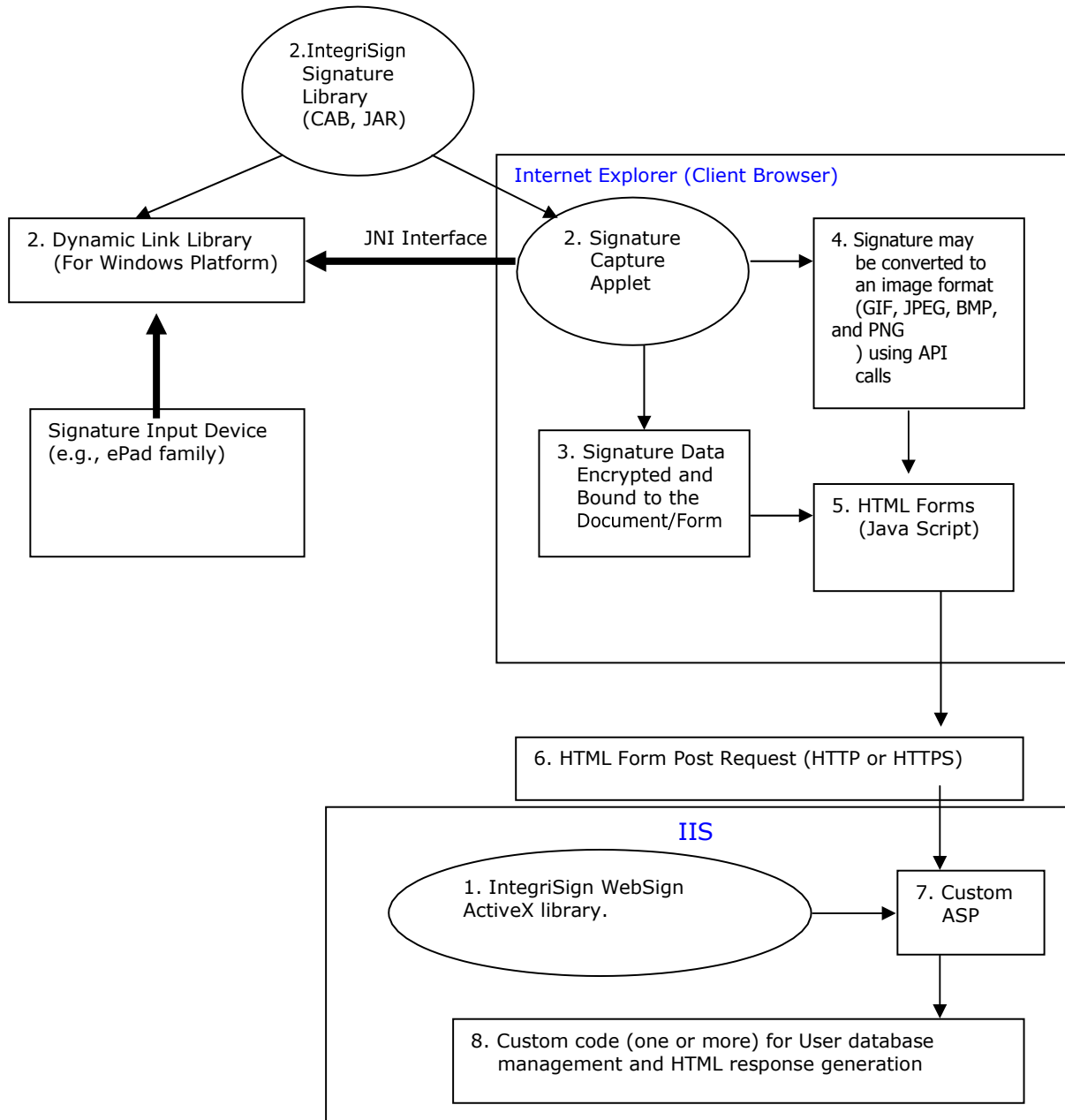
# 4.0 – System Requirements

**Client:**     Internet Explorer browser 7.0 and above with JVM build supporting JNI (Java Plug-in 1.4 onwards is recommended)
**OS:**     Windows 7 and above
**Server:**     Internet information Server (IIS 5.0 onwards) with ActiveX support

# 5.0 – Process

Steps are illustrated in the flow diagram below. Processes 2-5 occur at the client end (IE). Processes 1,7,8 occur at the server side. Process 6 is transmission from client to server.

2.IntegriSign Signature Library (CAB, JAR)

2. Dynamic Link Library (For Windows Platform)

JNI Interface

Signature Input Device (e.g., ePad family)

Internet Explorer (Client Browser)

2. Signature Capture Applet

4. Signature may be converted to an image format (GIF, JPEG, BMP, and PNG ) using API calls

3. Signature Data Encrypted and Bound to the Document/Form

5. HTML Forms (Java Script)

6. HTML Form Post Request (HTTP or HTTPS)

IIS

1. IntegriSign WebSign ActiveX library.

7. Custom ASP

8. Custom code (one or more) for User database management and HTML response generation

1. **Installation:** The IntegriSign WebSign server Library along with the custom servlet is loaded onto the server. The downloadable distribution unit (CAB) is also loaded onto the server and a reference is given to it from the HTML forms to be downloaded.

2. **Client setup:** The first time a user downloads an HTML form for signing from a particular machine, the distribution component (CAB) is downloaded onto the client and installed. The CAB file consists of the IntegriSign WebSign client library, signature capture applet and the Windows DLLs. No subsequent download is required.

3. **Signature capture and processing**: The signature capture applet calls the Windows DLLs through the Windows JNI (Java Native Interface) and gets the signature data from the input device. The raw data captured from the input device is processed and encrypted using the IntegriSign WebSign 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 transmitted as a part of HTML form data and can be stored as a string field in any database.

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 at client.

5. **Assign signature to HTML form:** The signature data (BASE64 string) is assigned to one of the fields of an HTML form and is transmitted along with the rest of the form data.

6. **Transmission:** The signature data along with the rest of the HTML form data is transmitted to the server using HTML Form Post request (HTTP/HTTPS).

7. **Processing data:** The custom ASP (using the WebSign API) decrypts/decodes and processes the received data with the help of the IntegriSign ActiveX library. The processing involves the verification of document integrity and decoding the signature image (JPEG, GIF, BMP and PNG) data.

8. **Database:** Custom code handles the user database management and the HTML response generation.

## 6.0– Running IntegriSign WebSign Enabled Samples

1. Copy the capbind.htm from samples folder into the HTML directory of the web server.
2. Copy all the cab and jar files from SDK folder to the directory where capbind.htm page is copied. If the cab and jar files are published into a different directory than the htm page specify the correct URL in htm page wherever those files are referred.
3. Edit the capbind.htm and change the action part of the Form Submit to suit your IIS directory structure (and also URL links wherever applicable).
4. Copy the esWebSign.dll from SDK folder into one of the folders onto web server.
5. Register the dll. Click Start → Run and type regsvr32 <dll_path>\esWebSign.dll and then click OK.
6. Copy the EsUtil.dll from the SDK folder into the system folder of the web server machine.
7. Edit the capbind.asp from SDK folder to modify the file paths (where WebSign generated images are stored). Copy it onto IIS web server.
8. Access the capbind.htm from the browser.

## 7.0 – Enabling Web Pages with WebSign

Follow these steps to enable HTML pages with IntegriSign WebSign Applet.

1. Place the Following code into the HTML page (which will enable the web page with IntegriSign WebSign). Publish the cab, jar onto the web server. If it is published into a directory other than html directory change the *cabbase* PARAM tag accordingly.

```
<OBJECT classid=clsid:E634B267-B8E7-406C-A308-988636B7D7E1
NAME=websignsup width=0 height=0 codebase=websignsup.cab#Version=11,2,0,1>
        <param name=useslibrary value=websignsup>
        <param name="useslibrarycodebase" value=websignsup.cab>
        <param name="useslibraryversion" value=11,2,0,1>
        </OBJECT><BR>
        <OBJECT classid=clsid:8AD9C840-044E-11D1-B3E9-00805F499D93
NAME=Sign width=125 height=65>
        <PARAM NAME=code VALUE=integrisign.webclient.WebSign>
        <PARAM NAME=archive VALUE=websignsunjvm.jar>
        <PARAM NAME=scriptable VALUE=true>
        <PARAM NAME=cache_option VALUE=Plugin>
        <PARAM NAME=cache_archive VALUE=websignsunjvm.jar>
        <PARAM NAME=cache_version VALUE=10.5.0.1>
        <PARAM NAME=MAYSCRIPT VALUE=true>
        <PARAM NAME=borderstyle VALUE=1> Java Plugin not enabled
        </OBJECT>
```

2. Write the JavaScript's necessary to invoke the Signing.

3. Copy the esWebSign.dll on some folder on the server machine.

4. Register the dll. To register the dll. Click Start → Run and type regsvr32 <dll_path>\esWebSign.dll and then click OK.

5. Copy the Esutil.dll into the system directory of the web server machine.

6. Write ASP to handle the HTML forms.

7. Access the esWebSign.dll for signature related tasks(Base64 decoding , content integrity verification etc..)

# 8.0 – WebSign (Java) API Model

Implementation routines needed

- Client Side
  WebSignSup.cab websignsunjvm.jar
  This cab file contains the necessary dlls and the WebSign applet and its supporting files.

- Server Side
  IntegriSign WebSign ActiveX library IntegriSign package is collection of Server API.

# 9.0 – Client API

**WebSign**

public class **WebSign** extends **Applet**

*Description*

> WebSign applet is responsible for capturing the Signature dynamics, and processes the signature data and paints the signature on the web form. This applet internally uses IntegriSign WebSign 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 assigned to an HTML field and submit it to the web server. This class is also capable of generating different image formats (BMP, GIF, JPEG) 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 and Address. WebSign applet can be customized to add other host functionalities. WebSign applet is also capable of generating MD5 hash of the content to be authenticated optionally.

> This applet class is shipped as a Downloaded CAB (websignSup.cab) file along with the supporting classes and dlls. The Certificate issued by VeriSign class-3 commercial publisher CA Certificate signs this CAB file.

## Method Index

- 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 data as string.

  *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 data

  as String

  *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.

    Note: This function is only supported when running under Internet Explorer with Sun Plug-in installed and enabled.

- 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 String **getInputDeviceInfo()**

    *Arguments*

    None

    *Return Value*

    This method returns the Signature Input device details, such as manufacturer name, device ID, device version as a concatenated String.

NOTE: All the get methods should be called after the act of signing is finished.

- 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.

- public void **enableWhiteSpaceRemoval**(boolean state)

  By default the value of this property is set to TRUE. When this property is set to TRUE, the pad removes the white space around the signature drawn on the pad and sends only the rectangle that contains the signature information to the PC. Thus, when the PC displays the signature on a Signature Field, the signature typically appears larger and more legible (compared to no white space removal), especially in the usual case where the Signature Field in the document is small. When this setting is FALSE, the signature pad sends to the PC the entire image of the drawing surface of the pad without removing white space.

  *Arguments*

  boolean : To enable or disable white space removal.

  *Return Value*

  This method returns void.

- public void **enableMaxEnlargementFactor**(boolean state)

  By default the value of this property is set to TRUE. This property lets the user to specify if the Maximum Enlargement Feature should be enabled or not. The Maximum Enlargement feature is used only in cases where the size of signature on the signature pad (after white space removal) is smaller than the size of the Signature control in the application. When the signature drawn on the signature pad is smaller than the size of the Signature Field in the document, IntegriSign will enlarge the signature to fill the Signature Field. However, to limit the enlargement the user can specify a limit to the enlargement as described in the next property.

  *Arguments*

  boolean : To enable or disable signature enlargement feature.

  *Return Value*

  This method returns void.

- public void **setMaxEnlargementFactor**(float value)

  The input variable type for this property is float and the value can range between 1.0 to 2.0. If the **enableMaxEnlargementFactor** property explained above is set to TRUE and this value is not set or set beyond the range 1.0 to 2.0, then a default value of 1.25 is considered.

  *Arguments*
  float : Value of the enlargement factor.

  *Return Value*

  This method returns void.

▪ public void **setEnableLCDButtons**(boolean show)

 Shows or hides the OK/CLEAR/CANCEL buttons on an LCD device. The default value is set to TRUE. Applicable only for LCD type of device.

 *Arguments*

 boolean: True or False to show or hide the buttons.

 *Return Value*

 This method returns void.

## SignNowEx 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:
  -  signature options can be enabled/disabled for end user?
Set the showOptions to true/false to show or hide options button.

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** (string name, string designation, string organisation, string address, string footer, boolean showOptions) throws ValidationException

 *Arguments*

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

 designation: Designation of the Signer.

 organization: Organization of the Signer.

 address: Address of the Signer.

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

 showOptions: Enables or disables Options button on signature capture dialog.

 *Return Value*

 This method returns void.

- public void **signNowEx** (string ID, string name, boolean showOptions) throws ValidationException

    *Arguments*

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

    name: Name of the Signer. Cannot be empty.

    showOptions: Enables or disables Options button on signature capture dialog.

    *Return Value*

    This method returns void.

- public void **signNowEx** (string ID, string name, string desg, string org, string Addr, string Footer, boolean showOptions) 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: Organization of the Signer.

    address: Address of the Signer.

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

    showOptions: Enables or disables Options button on signature capture dialog.

    *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) throws ValidationException

    *Arguments*

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

    designation: Designation of the Signer.

    organization: Organization 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: Enables or disables Options button on signature capture dialog.

    breserved2: reserved parameter.

    *Return Value*

    This method returns void.


- public void **signNow** (string ID, string name, boolean breserved1, boolean showOptions, boolean breserved2) 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: Enables or disables Options button on signature capture dialog.

    breserved2: reserved parameter.

    *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) 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: Organization 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: Enables or disables Options button on signature capture dialog.

    breserved2: reserved parameter.

    *Return Value*

    This method returns void.

- public int **verifyDocument()**

    *Arguments*

    None

    *Return Value*

    This method returns the content integrity status as integer.
    - 0 - Contents have not changed
    - 1 - Contents have changed
    - 2 - Hash generated with different version
    - 3 - Photographing information is not stored
    - 4 - signature string is empty

# 10.0 – Server API

ProgID of the Server Side ActiveX library is esWebSign.WebSign. This Activex dll is used for the

1. Content integrity verification upon the form submit.
2. Decoding Base64 data.
3. Decoding and writing the Image data into specified location as file.

## Method Index

- *VerifyDocument*

    This function returns the data integrity status as integer when signature string and the form data submitted are passed as arguments.

    *Arguments*

    FormString : Form data submitted by client concatenated as single string. The order of concatenation should be same as the one implemented in GetHashData() JavaScript function in the client html page.

    SignStr: IntegriSign signature string captured at the client end

    *Return value*

    This method returns the form data tamper status as integer

    0= Form contents have not changed

    1= Form contents have changed

    2= Signature has been created with a different Photographing version

    3= Form Content Information is not stored

    4= Internal Error. Error Re-Photographing the Form content

- *DecodeBase64String*

    This method is used for decoding the Base64 Encoded String into a byte array.

    *Arguments*

    Basestr : Any Base64 String.

    *Return Value*

    Returns the decoded data as Byte Array.

- *CreateImageFile*

    This method takes the signature Image data(BMP OR JPEG OR GIF) as base 64 string and writes the image bytes into a file at the specified location and name and extension.

    *Arguments*

    ImageFilename: Name of the image file name to be created including file path. (the extension of the file name should be according to the type ImageString (BMP,JPEG,GIF)

    ImageString: Image data(either BMP or JPEG or GIF) retrieved from HTML Form as Base64 String.

    e.g. . If BMP Image data is passed as argument the file extension should be .bmp. Return value

    Returns the file creation status as boolean

    NOTE: Proper distinction should be made in making a decision to call either *DecodeBase64String* or *CreateImageFile methods*. When Image data is required as byte array call DecodeBase64String and when image needs to be written into a file call CreateImageFile method.

# 11.0 – Troubleshooting

**Problem:** When IntegriSign WebSign function is invoked, "Unsatisfied Link error" or "Signpad.dll cannot be loaded" is occurs in IE Browser.

This error occurs when one of the supporting libraries SignPad.dll required by WebSign applet is not properly loaded on to the system. This is installed via WebSignSup.cab. Please make sure to check your security settings so that the ActiveX control installation via signed distribution units is enabled.