

WebSign SDK (Java)

Java(client) - Java(Server)

Developer's Guide

CONTENTS

| | |
|---|-----------|
| 1. Introduction | 1 |
| 2. Abstract..... | 1 |
| 3. Key Features | 1 |
| 4. System requirements | 1 |
| 5. Process..... | 1 |
| 6. Running WebSign enabled samples | 3 |
| 7. Enabling web pages with IntegriSign WebSign | 4 |
| 8. WebSign SDK (Java) API Model..... | 5 |
| 8.1 Client API | 5 |
| 8.2 Server API | 13 |
| 8.2.1 VerifyIntegrity | 13 |
| 8.2.2 Base64Format | 14 |
| 8.2.3 SaveImage..... | 15 |
| 9. Troubleshooting | 16 |

1. Introduction

IntegriSign is a handwritten signature Capture and Verification System, using 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 browser. 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. 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. Key Features

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

4. System requirements

Client: Internet Explorer browser 5.0 and above with JVM build supporting JNI.
(Java Plug-in 1.4 onwards is recommended)

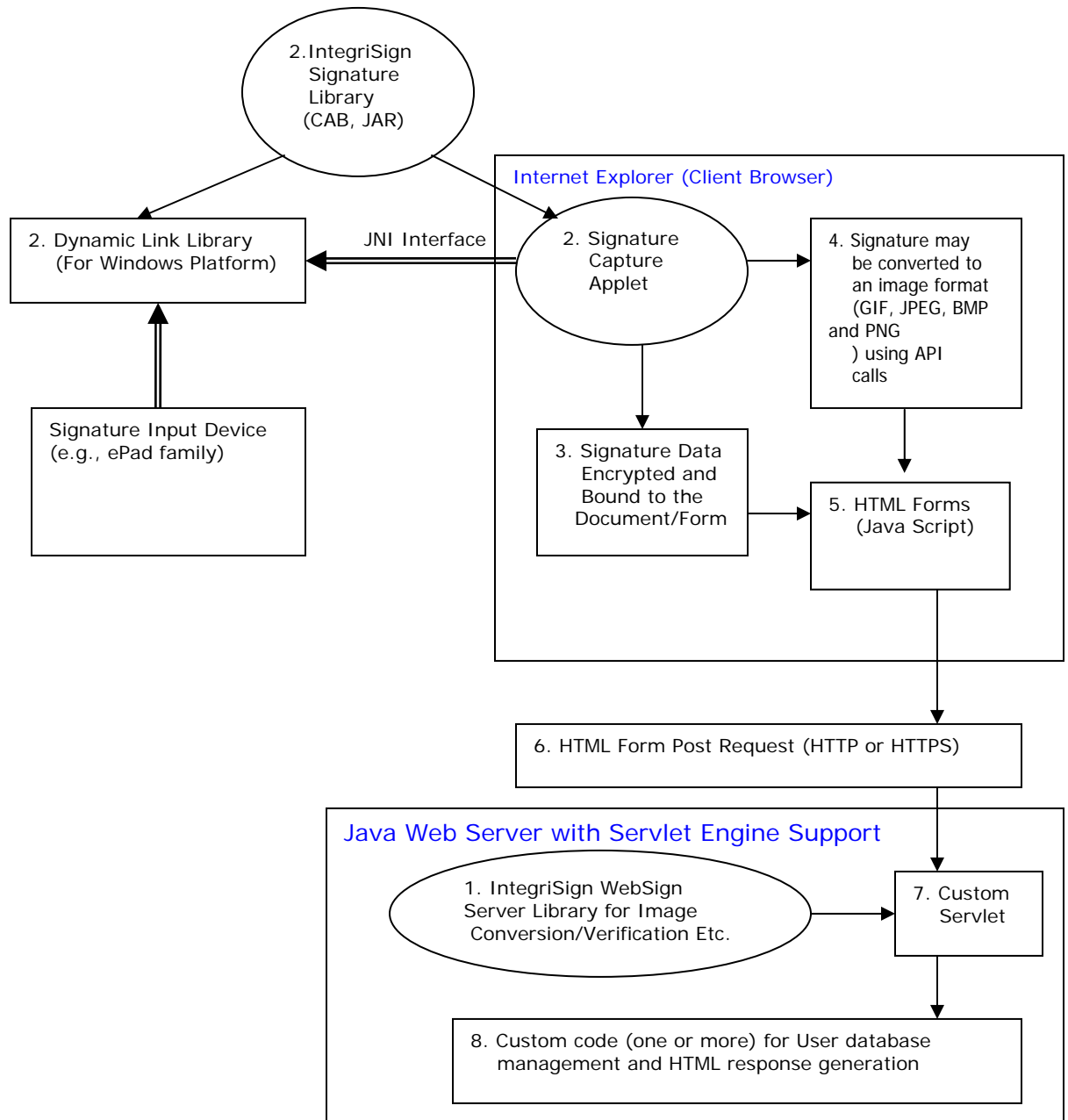
OS: Windows 95/98/Me/2000 (Service Pack 2)/XP/Windows Vista
Windows NT (Service Pack 4)

Server: Any Java enabled Web server.

5. Process

(Steps are illustrated in the flow diagram below)

Processes 2-5 occur at the client end (IE or Netscape browser). Processes 1,7,8 occur at the server side. Process 6 is transmission from client to server.



1. **Installation:** IntegriSign WebSign server Library along with the custom servlet is loaded onto the server. The downloadable distribution unit (CAB or JAR file) 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 or JAR file) 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 input device may be any ePad 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 from either client or server side.
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 servlet (developed using the Java SDK) decrypts and processes the received data with the help of the IntegriSign WebSign server library. The processing involves the verification of document integrity and if necessary the conversion of signature data into an image format (JPEG, GIF, BMP and PNG). The developer has the option of generating the image files at the client end or the at server end.
8. **Database:** Custom code (developed using Java SDK) handles the user database management and the HTML response generation.

6. Running WebSign enabled samples

1. Copy the capbind.htm from samples folder into the HTML directory of the java webserver.
2. Copy all the cab, and jar files from SDK folder to the directory where capbind.htm page is copied. If the cab file is published into a different directory than the htm page specify the URL in the cabbase parameter of applet tag.
3. Edit the capbind.htm and change the action part of the Form Submit to suit your java webserver directory structure (and also URL links wherever applicable).
4. Edit the ProcessIntegrisignDemo.java from SDK folder to modify the file paths in the file streams (current code writes the files to the default directory where write permissions are given). Follow the comments inside the java file.
5. Compile the ProcessIntegrisignDemo.java and copy into the servlets directory of the java web server.

6. Copy the integrisign package from SDK folder to a directory where classpath is set to web server JVM. If it already exists overwrite the existing one.

7. Start web server.

8. Start IE browser and access the capbind.htm page.

7. Enabling web pages with IntegriSign WebSign

Follow these steps to enable HTML pages with IntegriSign 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=9,5,2,0>
  <param name=useslibrary value=websignsup>
  <param name="useslibrarycodebase" value=websignsup.cab>
  <param name="useslibraryversion" value=9,5,2,0>
</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=7.5.0.3>
  <PARAM NAME=MAYSCRIPT VALUE=true>
  <PARAM NAME=borderstyle VALUE=1>
  Java Plugin not enabled
</OBJECT>
```

2. The JavaScript function GetHashData() should be implemented for the purpose of content binding. If no content binding is required at least null implementation (the method should return empty string).

```
function getHashData() {
    //In this example uid,uname field values are sent to Hash
    //Similar way any number of form field values can be sent to hash
    var hashstr;

    hashstr=window.document.integrisignfrm.uid.value+window.document
    .integrisignfrm.uname.value+window.document.integrisignfrm.crdno.v
    alue;
    // If no hashing is required return empty string
    return hashstr;
}
```

3. Write the JavaScript's code necessary to invoke the act of signing.
4. Copy the integrisign package into the Java web server.

5. Write Servlets or JSP to handle the HTML forms.
6. Access the integrisign package for signature related tasks like Base64 decoding , content integrity verification etc.

8. WebSign SDK (Java) API Model

Implementation routines needed

- Client Side
WebSignSp.cab
websignsunjvm.jar

This cab file contains the necessary dlls and the WebSign applet and its supporting files.

- Server Side
integrisign package

integrisign package is collection of Server API.

8.1 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 (websign.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.

- 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.

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

A. Whether signature options can be enabled/disabled?

Set the showOptions to true/false depending on whether to 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** (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: 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.

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

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: 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.

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: 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.

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

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: 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

reserved1 : reserved parameter

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

reserved2: 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

8.2 Server API

Classes

8.2.1 VerifyIntegrity

Constructor Index

public **VerifyIntegrity ()**

Method Index

- public **int verifyDocument** (**string** signstring, **string** hashstring)

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 GetHashCode () is implemented in the client script.

Arguments

signstring: This is the signature string submitted by client for authentication as a BASE64 String.

hashstring: This is a string argument containing the information needed to be checked for integrity.

Return Value

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

- 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

8.2.2 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.

8.2.3 SaveImage

public **class** **SaveImage**

This class is intended for generating the signature images from the raw base 64 signature string. It has got apis to specify the width and height of the image to be generated. Currently JPEG and GIF image generation is supported.

- Public byte[] **getBMPIImageBytes** (**String** signstr, **int** w,**int** h)

This method generates a BMP image of specified width and height from the supplied raw signature data.

Arguments

signstr: Raw signature data as Base64 string.
w: Width of the image to be generated.
h: Height of the image to be generated.

Return value

This Method returns a Jpeg image bytes as a byte array.

- Public byte[] **getGifImageBytes** (**String** signstr, **int** w,**int** h)

This method generates a Gif image of specified width and height from the supplied raw signature data.

Arguments

signstr: Raw signature data as Base64 string.
w: Width of the image to be generated.
h: Height of the image to be generated.

Return value

This Method returns a Gif image bytes as a byte array.

- Public byte[] **getPngImageBytes** (**String** signstr, **int** w,**int** h, 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 bytes as a byte array.

- public void setSignThickness(byte signthickness)

This function sets the thickness to be used while drawing the signature trace in the generated images This can be set prior to 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.

9. 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.