# Signature Library - Java API

## Introduction

The Signature Library includes a set of ActiveX components which provide the functionality for capturing and displaying signatures. This document provides the information a Java developer needs to use the components, to be read in parallel with the Signature Library COM API.

# Class: SigCtl

The class extends java.awt.Canvas to display an embedded signature object and provides a base class for DynamicCapture.
Full qualification: com.florentis.signature.SigCtl

## Summary

| Method |
|---|
| aboutBox |
| close |
| Property |
| signature |
| licence |

## Methods

### aboutBox()

The method displays an About Box for the control. The dialog box will display version, licensing and contact information

| *public static native void aboutBox()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

### close()

The method "Closes" the object, releasing the underlying COM object (and thus freeing resources).

| *public native void close()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

## Properties

| Property | Type | Description |
|---|---|---|
| signature | SigObj | The read/write property contains a SigObj class |
| licence | String | The read/write property contains a licence string |

# Class: DynamicCapture

The class provides the signature capture functionality
Full qualification: com.florentis.signature.DynamicCapture

## Summary

| Method |
|---|
| capture |
| close |
| Property |
| licence |

## Methods

### capture()

The method calls signature capture.

| *public static native int capture( SigCtl sigCtl, String who, String why, Hash what, Key key )* | |
|---|---|
| *Parameters* | |
| sigCtl | Required SigCtl |
| who | Required signatory name |
| why | Required reason for signing |
| what | Optional Hash object (may be null) |
| key | Optional Key object (may be null) |
| *Return Value: int* dynamicCaptReturn | |

| | |
|---|---|
| 0 | Signature captured successfully. |
| 1 | Signature not captured because user cancelled |
| 100 | No capture service available |
| 101 | Pad Error |
| 200 | Error - unable to parse document contents |

### close()

The method "Closes" the object, releasing the underlying COM object (and thus freeing resources).

| |
|---|
| *public native void close()* |
| *Parameters:none* |
| *Return Value: none* |

## Properties

| **Property** | **Type** | **Description** |
|---|---|---|
| licence | String | The read/write property contains a licence string |

## Class: eSeal

The class provides the eSeal capture functionality
Full qualification: com.florentis.signature.eSeal

## Summary

| Method |
|---|
| capture |
| close |
| Property |
| url |
| hAlign |
| vAlign |
| hScale |
| vScale |
| transparency |
| cacheImage |
| width |
| height |
| name |
| id |
| licence |

## Methods

### capture()

The method inserts an eSeal and optionally captures a handwritten signature.

| *public native int capture( SigCtl sigCtl, int captureMode, String who, String why, Hash what, Key key )* | |
|---|---|
| *Parameters* | |
| sigCtl | Required SigCtl |
| captureMode | Required int may be one of:<br>• eSeal.RequireSignature - Insert eSeal and capture handwritten signature<br>• eSeal.esNoSignature - Insert eSeal (without signature capture)<br>• eSeal.esSignatureOptional - Insert eSeal and capture signature if tablet is available |
| who | Required signatory name |
| why | Required reason for signing |
| what | Optional Hash object (may be null) |
| key | Optional Key object (may be null) |
| *Return Value: int* esealCaptureReturn | |
| 0 | Signature captured successfully. |
| 1 | Signature not captured because user cancelled |
| 100 | No capture service available |
| 101 | Pad Error |
| 200 | Error - unable to parse document contents |
| 300 | Error - unable to load eSeal image |

### close()

The method "Closes" the object, releasing the underlying COM object (and thus freeing resources).

| *public native void close()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

## Properties

| **Property** | **Type** | **Description** |
|---|---|---|
| url | String | Read/Write String contains the URL of the image in a standard format:<br>JPEG, PNG, TIF, GIF, BMP |
| hAlign | int | Read/Write int defines horizontal alignment of the image as one of:<br>• eSeal.esLeft<br>• eSeal.esCentre (default)<br>• eSeal.esCenter<br>• eSeal.esRight |

| | | |
|---|---|---|
| vAlign | int | Read/Write int defines vertical alignment of the image as one of:<br>• eSeal.esTop<br>• eSeal.esCentre (default)<br>• eSeal.esCenter<br>• eSeal.esBottom |
| hScale | int | Read/Write int defines percentage of X dimension (defaults to 100) |
| vScale | int | Read/Write int defines percentage of Y dimension (defaults to 100) |
| transparency | int | Read/Write int defines percentage of transparency:<br>• 100 = maximum, not visible<br>• 0 = minimum, unchanged (default) |
| cacheImage | boolean | Read/Write Boolean false if URL is to be accessed at time of signing, true if image is saved<br>within the eSeal object. Defaults to false. For internal use only. |
| width | int | Read-only int width of image in HIMETRIC 0.01mm units |
| height | int | Read-only int height of image in HIMETRIC 0.01mm units |
| name | String | Read/Write String name for internal use only |
| id | String | Read-only String for internal use only |
| licence | String | The read/write property contains a licence string |

## Class: Hash

The class is used to calculate a one-way hash, the value of which is a fixed length 'string', from an arbitrary length data set.
Full qualification: com.florentis.signature.Hash

### Summary

| Method |
|---|
| add |
| clear |
| close |
| **Property** |
| type |
| hash |

### Methods

#### add()

The method adds data to the Hash object

| public native void add( data ) | |
|---|---|
| *Parameters* | |
| data | The data item can be one of the types:<br><br>boolean, byte, char, short, int, long, float, double, String, byte[] |

| |
|---|
| *Return Value: none* |

### clear()

The method clears the Hash object.

| |
|---|
| *public native void clear()* |
| *Parameters:none* |
| *Return Value: none* |

### close()

The method "Closes" the object, releasing the underlying COM object (and thus freeing resources).

| |
|---|
| *public native void close()* |
| *Parameters:none* |
| *Return Value: none* |

## Properties

| **Property** | **Type** | **Description** |
|---|---|---|
| type | int | Read/Write value sets the type of hashing algorithm to one of:<br>• Hash.none<br>• Hash.md5<br>• Hash.sha1<br>• Hash.sha224<br>• Hash.sha256<br>• Hash.sha384<br>• Hash.sha512 |
| hash | String | Read-only String value |

# Class: Key

The class is used to protect the integrity of signature data
Full qualification: com.florentis.signature.Key

## Summary

| |
|---|
| Method |
| set |
| close |
| Property |
| type |

## Methods

### set()

The method sets the type of the Key object

| public native void set( int ) | |
|---|---|
| *Parameters* | |
| type | The key type can be one of the following:<br><br>• Key.none<br>• Key.md5<br>• Key.sha1<br>• Key.sha224<br>• Key.sha256<br>• Key.sha384<br>• Key.sha512 |
| *Return Value: none* | |

### close()

The method "Closes" the object, releasing the underlying COM object (and thus freeing resources).

| *public native void close()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

## Properties

| **Property** | **Type** | **Description** |
|---|---|---|
| type | int | Read-only value returns the type of Key set<br><br>• Key.none<br>• Key.md5<br>• Key.sha1<br>• Key.sha224<br>• Key.sha256<br>• Key.sha384<br>• Key.sha512 |

# Class: SigObj

The class provides properties and methods for the signature
Full qualification: com.florentis.signature.SigObj

## Summary

| Method |
|---|
| clear |

| |
|---|
| checkIntegrity |
| checkSignedData |
| renderBitmap |
| renderRect |
| readEncodedBitmap |
| close |
| Property |
| additionalData |
| crossedOut |
| extraData |
| height |
| ink |
| isCaptured |
| sigData |
| sigText |
| who |
| why |
| when |
| width |

## Methods

### checkIntegrity()

The method checks the integrity of the Signature object to detect whether it has been tampered with since signing

| *public native int checkIntegrity( Key key )* | |
|---|---|
| *Parameters* | |
| key | Optional Key object. If not supplied the code uses Key type MD5 by default. |
| *Return Value: checkIntegrityResult* | |
| 0 | integrityOK Data has not changed since signature capture |
| 1 | integrityFail Data has changed since signature capture |
| 2 | integrityMissing Signature integrity value not found |
| 3 | integrityWrongType Signature was captured using a different integrity check version |

### checkSignedData()

The method checks for a match between a given hash and that provided when the signature was captured.

| *public native int checkSignedData( Hash hash )* |
|---|
| *Parameters* |

| | |
|---|---|
| hash | Required Hash object to be compared with the one provided when the signature was captured |
| *Return Value: checkIntegrityResult* | |
| 0 | Data has not changed since signature capture |
| 1 | No signature captured, or signature was captured without a hash |
| 2 | Signature was captured with a different type of hash |
| 3 | Data has changed since signature capture |
| 4 | Error checking signed data |

## renderBitmap()

The method renders a signature to a file or byte array

| *public native Object renderBitmap( String outputFilename, int dimensionX, int dimensionY, String mimeType, float inkWidth, int inkColor,<br>int backgroundColor, float paddingX, float paddingY, int flags )* | |
|---|---|
| *Parameters* | |
| outputFilename | The pathname of the file to receive the image output. May be null if byte array output is selected by flags. |
| dimensionX<br><br>dimensionY | X/Y dimensions specified as DPI (dots per inch) or Pixels.<br><br>• Negative value = DPI (the signature is not scaled)<br>• Positive value = Pixels (the signature is scaled to the dimensions) |
| mimeType | Specifies the image format as one of:<br><br>• image/bmp<br>• image/jpeg<br>• image/gif<br>• image/tiff<br>• image/png |
| inkWidth | Specifies the signature ink width in mm |
| inkColor<br><br>backgroundColor | Specifies the pen ink and background colours in MS COM COLORREF format (BGR)<br>Examples:<br><br>• cWhite = 0xFFFFFF<br>• cBlack = 0x00<br>• cRed = 0x0000FF |
| paddingX<br><br>paddingY | The specified padding is applied around the signature image, added to both the left and right for paddingX, and both the top and bottom for paddingY.<br><br>X/Y dimensions are specified as mm or Pixels.<br><br>• Negative value = mm (1inch = 25.4mm)<br>• Positive value = Pixels |

| | |
|---|---|
| flags | A bit mask of the following categorised values:<br><br>Output Group: (single value)<br>• SigObj.outputBinary image is returned as a byte array<br>• SigObj.outputBase64 image is returned as a Base 64 encoded string<br>• SigObj.outputFilename outputFilename contains the pathname of the file to be created<br><br>Color selection Group: (single value)<br>• SigObj.color1BPP 1 bit per pixel<br>• SigObj.color24BPP 24 bit per pixel<br>• SigObj.color32BPP 32 bit per pixel<br><br>Optional image format flags:<br>• SigObj.backgroundTransparent transparent background<br>• SigObj.colorAntiAlias option with 24 and 32 BPP<br><br>Optional Image extension:<br>• SigObj.encodeData Encode signature data within image<br>• SigObj.watermark Include watermark within image to indicate presence of encoded data |
| *Return Value: function dependent* | |
| outputFilename | null |
| outputBinary | Byte array containing the image file contents |
| outputBase64 | String containing base-64 representation of image file data |

**Example**

```
try
 {
  com.florentis.signature.SigObj sig = new com.florentis.signature.SigObj();
  sig.sigText( readFileAsString("..\\dataJS1.txt"));
  sig.renderBitmap("..\\temp1.png", -500, -500, "image/png", 1.0f, cRed, cBlue, -1.0f, -1.0f,
      com.florentis.signature.SigObj.outputFilename | com.florentis.signature.SigObj.color32BPP);
 }
catch (Exception e)
 {
  System.out.println("Exception:" + e);
 }
```

## renderRect()

The method renders an image of the signature within a given rectangle on a specified device context.

| *public native void renderRect( long hdcTarg, long hdcRef, int left, int top, int right, int bottom, float inkWidth, int inkColor, int option,<br>short zoom, short rotation )* | |
|---|---|
| *Parameters* | |
| hdcTarg | Required long value specifying handle to output device context. |
| hdcRef | Required long value specifying handle to reference device context.<br>May be the same as hDCTarg |
| left<br>top<br>right<br>bottom | Required int values defining the bounding rectangle in which the signature is to be rendered |
| inkWidth | Optional float value specifying width, in mm, of pen used to draw signature.<br>(Default is 0.7mm.) |
| inkColor | Optionally specifies the pen ink and background colours in MS COM COLORREF format (BGR)<br>(Default is black)<br>Examples:<br>• cWhite = 0xFFFFFF<br>• cBlack = 0x00<br>• cRed = 0x0000FF |
| option | Optional int value specifying the scaling mode of the rendered signature, with possible values:<br>• 0 - dspForceFit scale signature to exactly fit the bounding rectangle (default)<br>• 1 - dspUseZoom scale signature according to the Zoom argument.<br>• 2 - dspBestFit reduce size of signature to fit area if it is too big |
| zoom | Optional short value specifying percentage by which the signature image is to be scaled. (Default is 100%.) |
| rotation | not used |
| *Return Value: none* | |

## readEncodedBitmap()

The method reads the encoded SigObj data from an image file which was created using RenderBitmap()

| *public native int readEncodedBitmap( String filename )* | |
|---|---|
| *Parameters* | |
| filename | Required string contains the pathname of the image file containing the encoded<br>SigObj |
| *Return Value: readEncodedBitmapResult* | |
| 0 | Signature data decoded OK |
| 1 | File not found |
| 2 | Bitmap is not a supported image type |
| 3 | Encoded signature data not found |

## close()

The method "Closes" the object, releasing the underlying COM object (and thus freeing resources).

| *public native void close()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

## Properties

| **Property** | **Type** | **Description** |
|---|---|---|
| additionalData | int | additionalData returns additional capture data eg pad driver version |
| crossedOut | boolean | Read-only Boolean value is true if the signature appears crossed out as invalid |
| extraData | String | Write once, Read string value referenced by key name or "" for all values |
| height | int | Read-only value of the bounding rectangle of the signature in 0.01mm |
| ink | String | Read/Write CIC Ink Tools interface value |
| isCaptured | Boolean | Read-only value indicates if a signature has been captured |
| sigData | byte[] | Read/Write binary SigObj data |
| sigText | String | Read/Write hex string representation of sigData |
| who | String | Read-only string containing signatory name |
| why | String | Read-only string containing reason for signing |
| when | Date | when (int timeZone) returns the time & date of signature capture<br>when(0) returns TIimeLocal<br>when(1) returns TimeGMT |
| width | int | Read-only value of the bounding rectangle of the signature in 0.01mm |

# Class: WizCtl

The class extends java.awt.Canvas to reproduce the LCD display and provides the java interface to the COM control.
Full qualification: com.florentis.signature.WizCtl

## Summary

| Method |
|---|
| padConnect |
| padDisconnect |
| reset |
| addObject |
| addPrimitive |
| getObjectState |
| setFont |
| setEventHandler |
| display |
| fireClick |

| |
|---|
| processEvents |
| endProcessEvents |
| close |
| Property |
| inkingPad |
| enableWizardDisplay |
| padWidth |
| padHeight |
| zoom |
| licence |
| Enumerations |
| ObjectType |
| PrimitiveType |
| AlignmentType |
| CheckboxOptions |
| PrimitiveOptions |
| EventType |
| InputOptions |
| EncryptionAlg |

## Enumeration values

| **ObjectType** |
|---|
| objectText |
| objectButton |
| objectCheckbox |
| objectSignature |
| objectInput |
| objectInputEcho |
| objectRadioButton |

| **PrimitiveType** |
|---|
| primitiveLine |
| primitiveRectangle |
| primitiveEllipse |

| **AlignmentType** |
|---|
| textAlignLeft |

| |
|---|
| textAlignRight |
| textAlignCentre |
| textAlignJustify |

| CheckboxOptions |
|---|
| checkboxUnchecked |
| checkboxChecked |
| checkboxDisplayTick |
| checkboxDisplayCross |

| CheckboxOptions |
|---|
| checkboxUnchecked |
| checkboxChecked |
| checkboxDisplayTick |
| checkboxDisplayCross |

| PrimitiveOptions |
|---|
| primitiveLineSolid |
| primitiveLineDashed |
| primitiveOutline |
| primitiveFill |
| primitiveFillXOR |

| EventType |
|---|
| evTextClicked |
| evButtonClicked |
| evCheckboxChecked |
| evCheckboxUnchecked |
| evInputMinReached |
| evInputMaxReached |
| evInputExceeded |

| InputOptions |
|---|
| echoNoSpacing |
| echoHalfSpacing |
| echoSingleSpacing |

| |
|---|
| echoDoubleSpacing |
| echoUnderline |

| **EncryptionAlg** |
|---|
| encryptNone |
| encryptTripleDES |

## Methods

### padConnect()

Connects to the signature tablet / pad.

| *public native boolean padConnect()* | |
|---|---|
| *Parameters:none* | |
| *Return Value: boolean* | |
| 0 | Success |
| <>0 | Failed to connect |

### padDisconnect()

Disconnects the signature tablet / pad.

| *public native void padDisconnect()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

### reset()

Disables events, removes all internal controls and prepares for setting the display. Does not change the current display

| *public native void reset()* |
|---|
| *Parameters:none* |
| *Return Value: none* |

### addObject()

Adds an item to the pad control list.

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectType enum value |
| id | String, Specifies an identifier for the object |
| x, y | Position of the top left corner of the object on the pad display.<br>Value can either be absolute position in pixels, or one of the strings:<br>• X: "left", "right", "centre"<br>• Y: "top", "middle", "bottom" |

| | |
|---|---|
| data | data dependent on object type. |
| options | value dependent on object type |
| *Return Value:none* | |

### addObject(ObjectText)

Displays a text string on the pad using the current font

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectText |
| id | The following values have special meanings when used with a signature object:<br>"who"Text in Data will also be used as name of signatory.<br>"why"Text in Data will also be used as reason for signing.<br>"when"Reserved for future use<br>(Can be null or an empty string) |
| X, Y | Position of the top left corner of the object on the pad display.<br>Value can either be absolute position in pixels, or one of the strings:<br>X: "left", "right", "centre"<br>Y: "top", "middle", "bottom" |
| data | Text to display. |
| options | A value from the TextOptions Enumerator (Optional) |
| *Return Value:none* | |

### addObject(ObjectButton)

Creates a button – text surrounded by a rectangle which generates a click event when tapped with the pen. Text is displayed in the current font

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | | |
|---|---|---|
| *Parameters:* | | |
| type | ObjectButton | |
| id | The following values have special meanings when used with signature or input objects: | |
| | "OK" | Accepts current input.<br>With a signature, stores the captured signature in the signature object and terminates input.<br>Until a signature has been captured, the button is disabled by the Wizard Control.<br>With an Input object, the button is disabled until the required minimum number of characters has been entered. |
| | "Clear" | Clears current input allowing user to start again<br>With a signature, clears any captured 'ink' from the display<br>With an Input object, clears all entered input |
| | "Cancel" | With a signature, clears any captured 'ink' and terminates input |
| | "Delete" | With an input object, deletes the last character |
| X, Y | Position of the top left corner of the object on the pad display.<br>Value can either be absolute position in pixels, or one of the strings:<br>X: "left", "right", "centre"<br>Y: "top", "middle", "bottom" | |
| data | Text to display. | |
| options | Either, an integer specifying button width in pixels or an ObjectOptions object.<br>If the given width is less than the width of the text (in the current font), it is ignored.<br>(Optional) | |
| *Return Value:none* | | |

### addObject(ObjectCheckbox)

Creates a checkbox – a small rectangle followed by text which toggles its state and generates an event when tapped with the pen. Text is displayed in the current font

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectCheckbox |
| id | The following values have special meanings when used with a signature object.<br>Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel. |
| X, Y | Position of the top left corner of the object on the pad display.<br>Value can either be absolute position in pixels, or one of the strings:<br>X: "left", "right", "centre"<br>Y: "top", "middle", "bottom" |
| data | Text to display. |
| options | A combination of values from the CheckboxOptions enum (Optional) |
| *Return Value:none* | |

### addObject(ObjectRadioButton)

Creates a radio button – a small circle followed by text. Radio buttons are used in groups where tapping on one with the pen selects it and deselects the currently selected button in the group. Tapping with the pen also generates an event. Text is displayed in the current font

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectCheckbox |
| id | The following values have special meanings when used with a signature object.<br>Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel. |
| X, Y | Position of the top left corner of the object on the pad display.<br>Value can either be absolute position in pixels, or one of the strings:<br>X: "left", "right", "centre"<br>Y: "top", "middle", "bottom" |
| data | Text to display. |
| options | ObjectOptions object specifying the name of the group to which this radio button belongs and, optionally, whether this option is initially selected. |
| *Return Value:none* | |

### addObject(ObjectSignature)

Puts the pad into signature capture mode and specifies a signature object or control in which a captured signature is saved. It is an error to add more than one ObjectSignature to the current control list

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectSignature |
| id | Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel;<br>can be null or an empty string |
| X, Y | Values ignored |
| data | A signature object or control. (Note: cannot be a SigCtlXHTML if an ObjectHash has been added) |
| options | A Key object to use for setting integrity of captured signature. (Optional) |
| *Return Value:none* | |

## addObject(ObjectInput)

Provides an input mechanism for PIN code entry.

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectInput |
| id | Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel; can be null or an empty string |
| X, Y | Values ignored |
| data | InputObj to be used for handling pin pad input |
| options | Not used, should be omitted |
| *Return Value:none* | |

## addObject(ObjectInputEcho)

Specifies location of and character to use for ObjectInput 'echo'.

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectInputEcho |
| id | Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel; can be null or an empty string |
| X, Y | Values ignored |
| data | Character to be used for each button press |
| options | Either, a combination of values from the InputEchoOptions enum or an ObjectOptions object.<br>(Optional) |
| *Return Value:none* | |

## addObject(ObjectHash)

Supplies a Hash object representing data to be bound to a captured signature. It is an error to add more than one ObjectHash to the current control list Cannot be used in conjunction with a SigCtlXHTML control (ie in a web page) as the latter automatically binds to the host document.

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectHash |
| id | Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel;<br>can be null or an empty string |
| X, Y | Values ignored |
| data | Hash object representing data to be bound to a captured signature. |
| options | Not used, should be omitted |
| *Return Value:none* | |

## addObject(ObjectImage)

Displays an image on the pad. The image can optionally be made clickable in which case click events are generated when the image is tapped with the pen.

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectImage |
| id | Supports the same reserved Ids as button objects. See AddObject(ObjectButton) above |
| X, Y | Position of the top left corner of the object on the pad display. Value can either be absolute position in pixels, or one of the strings: X: "left", "right", "centre" Y: "top", "middle", "bottom" |
| data | Image to display. Can be any one of the following: Filename: Name, including path, of image file URL: URL of an image file A string value is assumed to be a URL if it contains "://", otherwise it is assumed to be a filename Picture: OLE picture object (IPicture or IPictDisp interface) Image data: Binary image data as an array of bytes |
| options | Not used, should be omitted |
| *Return Value:none* | |

## addObject(ObjectDisplayAtShutdown)

Causes the current control set to remain displayed on the pad following disconnection.

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectDisplayAtShutdown |
| id | Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel; can be null or an empty string |
| X, Y | Values ignored |
| data | Not used, should be omitted |
| options | Not used, should be omitted |
| *Return Value:none* | |

## addObject(ObjectInking)

Provides a mechanism for capturing pad 'ink' as an image

| *public native void addObject( int type, String id, Object x, Object y, Object data, Object options )* | |
|---|---|
| *Parameters:* | |
| type | ObjectInking |
| id | Cannot be any of the values reserved for text or button objects: who, why, Ok, Clear, Cancel; can be null or an empty string |
| X, Y | Values ignored |
| data | Not used, should be omitted |
| options | Not used, should be omitted |
| *Return Value:none* | |

A snapshot of current ink is retrieved, in PNG format as a base-64 encoded string, using the WizCtl GetProperty method as follows:

```
  pngAsText = wizctl.getProperty("ObjectInking_Bitmap");
```

Alternatively, the image can be written to a file using SetProperty. The format of the image is determined by the file name extension (bmp, jpg, png or tif):

```
  wizctl.setProperty("ObjectInking_Bitmap", "c:\\file.png");
```

In both cases, the size of the image is the size of the LCD screen

## addPrimitive()

Adds a graphics primitive item to the internal list.

| *public native void* a*ddPrimitive( int type, Object x1, Object y2,  Object x2, Object y2, Object primdata, Object options )* | |
|---|---|
| *Parameters:* | |
| type | PrimitiveType enum value. |
| x1, y1 | If Type is PrimitiveLine, start position of line, otherwise position of top-left corner of bounding rectangle of item.<br>Value can be the absolute position in pixels or one of the strings:<br>"left", "right", "centre" (for X1) or "top", "middle", "bottom" (for Y1). |
| x2, y2 | If Type is PrimitiveLine, end position of line, otherwise position of bottom-right corner of bounding rectangle of item.<br>Value can be the absolute position in pixels, one of the strings:<br>"left", "right", "centre" (for X2) or "top", "middle", "bottom" (for Y2)<br>or a string in the format "+V" or " V" (where V is an integer) for a value relative to X1 or Y1. |
| primdata | Line width in pixels (Optional, default value 1) |
| options | Combination of PrimitiveOptions values (Optional, default value PrimitiveLineSolid + PrimitiveOutline) |
| *Return Value:none* | |

## getObjectState()

Returns state information of a given object or an empty Variant if specified object does not exist.

| public native Object getObjectState( String id ) | |
|---|---|
| *Parameters:* | |
| id | Identifier of object. |
| *Return Value: Object* | |
| Object | ObjectCheckbox:     1 = Checked; 0 = Unchecked<br>ObjectInput:             number of characters currently in input buffer<br>Other Object types:  Integer, value undefined<br>Id not recognised:     Empty object |

## setFont()

Sets the current font for new wizard objects.

| public native void setFont( Font font ) | |
|---|---|
| *Parameters:* | |
| font | Font selection |
| *Return Value:none* | |

## setEventHandler()

Sets the function to be called to handle tablet control events.

| *public void setEventHandler(WizCtlEvents handler)* | |
|---|---|
| *Parameters:* | |
| handler | WizCtlEvents  event handler |
| *Return Value:none* | |

## display()

Clears current display contents, turns on backlight (if not already on), updates display with all buffered objects and primitives and enables event handling.

| *public native void display()* |
|---|
| *Parameters:none* |
| *Return Value:none* |

## fireClick()

Simulates 'click' on an object (button, checkbox, image etc).
Allows, for example, a signature to be accepted by clicking a button on the PC screen rather than taping the OK button on the pad.

| *public native void fireClick( String id )* | |
|---|---|
| *Parameters:* | |
| Id | Id of pad control for which to simulate click |
| *Return Value:none* | |

## processEvents()

NOTE: This has no equivalent (nor is needed) in the Microsoft COM interface.
Once display() has been called, call this method to wait for input from the pad, which is delivered through implementing onPadEvent().
This call does not return until onPadEvent() returns false, the thread is interrupted or endProcessEvents() is called.

| *public native void processEvents() throws InterruptedException* |
|---|
| *Parameters:* none |
| *Return Value:none* |

## endProcessEvents()

Terminates endProcessEvents()

NOTE: This has no equivalent (nor is needed) in the Microsoft COM interface.
Call this method to signal the processEvents method to terminate and return.

| *public native void endProcessEvents()* |
|---|
| *Parameters:* none |
| *Return Value:none* |

## close()

"Closes" the object, releasing the underlying COM object (and thus freeing resources).

Note that, while the COM object is also released by finalize() during garbage collection, experience has shown that creating numerous WizCtl objects, for example as part of a frequently repeated process, can lead to problems if close() calls are not used.

| |
|---|
| public native void close() |
| *Parameters:* none |
| *Return Value:none* |

## Properties

| **Property** | **Type** | **Description** |
|---|---|---|
| inkingPad | boolean | Read-only boolean, True if the pad has a supported LCD display |
| enableWizardDisplay | boolean | Read-Write boolean enables/disables wizard controls |
| padWidth | int | Read-only int width of pad display in pixels |
| padHeight | int | Read-only int height of pad display in pixels |
| zoom | float | Read-Write float scaling of pad, 100 = 100% |
| licence | String | The read/write property contains a licence string |

# Interface: WizCtl.WizCtlEvents

This interface when implemented provides feedback events when an action is taken on the pad.
NOTE: This has slightly different behaviour than the COM version.
Full qualification: com.florentis.signature.WizCtl.WizCtlEvents

## Methods

### onPadEvent()

This is only called from within WizCtl.processEvents(). Return true to continue processing events or return false for processEvents() to return.

| | |
|---|---|
| boolean onPadEvent( WizCtl wizCtl, String id, Object eventType ) | |
| *Parameters:* | |
| wizCtl | The wizard control. |
| eventType | wizard event type |
| *Return Value: boolean* | |
| continue | Return true to continue processing events or return false for processEvents() to return. |

# Class: InputObj

The class provides the Input control for PIN code input
Full qualification: com.florentis.signature.WizCtl.InputObj

## Summary

| |
|---|
| Method |

| |
|---|
| clear |
| setEncryption |
| close |
| Property |
| minLength |
| maxLength |
| text |
| data |
| encryptionType |

## Methods

### clear()

Resets the input object ready to restart PIN capture.

| |
|---|
| *public native void clear()* |
| *Parameters:none* |
| *Return Value:none* |

### setEncryption()

Sets the encryption of the InputObject data.
Once set, encryption cannot be changed except by first calling the Clear method.
Currently, the only encryption algorithm supported is TripleDES. For this algorithm, "Key" must be a 24-byte (192-bit) binary value either in a byte array (a SafeArray of type VT_UI1) or a base64 encoded string. In addition, the following information will be required for decryption:
Cipher mode:CBC (cipher block chaining)
Initialisation Vector:8 bytes, all zero
Padding mode:PKCS 5

| | |
|---|---|
| *public native void setEncryption( int type, byte[] key )* | |
| *Parameters:* | |
| type | EncryptAlg enum value specifying type of encryption to be used. |
| key | byte array: the encryption key to be used |
| *Return Value:none* | |

### close()

"Closes" the object, releasing the underlying COM object (and thus freeing resources).

| |
|---|
| *public native void close()* |
| *Parameters:none* |
| *Return Value:none* |

## Properties:

| Property | Type | Description |
|---|---|---|
| minLength | int | Read-Write int minimum number of digits in the PIN |
| maxLength | int | Read-Write int maximum number of digits in the PIN |
| text | String | Read-only String containing the input data (optionally encrypted) |
| data | byte[] | Read-only byte[] containing the input data |
| encryptionType | int | Read-only int returns the type set by setEncryption |

## Class: Font

The class is used when setting the display font
Full qualification: com.florentis.signature.WizCtl.Font

### Summary

| Method |
|---|
| close |
| Property |
| name |
| size |
| bold |
| underline |
| strikethrough |
| weight |

### Methods

#### close()

"Closes" the object, releasing the underlying COM object (and thus freeing resources).

| public native void close() |
|---|
| *Parameters:none* |
| *Return Value:none* |

### Properties

| Property | Type | Description |
|---|---|---|
| name | String | Read-Write String value |
| size | double | Read-Write double value |
| bold | boolean | Read-Write boolean value |

|  |  |  |
|---|---|---|
| underline | boolean | Read-Write boolean value |
| strikethrough | boolean | Read-Write boolean value |
| weight | short | Read-Write short value |

## Class: ObjectOptions

The class is used to set options with the addObject method
Full qualification: com.florentis.signature.WizCtl.ObjectOptions

### Summary

|  |
|---|
| Method |
| setProperty |
| close |
| Property |
| none |

### Methods

#### setProperty()

|  |  |
|---|---|
| *public native void setProperty( String key, String value )*<br>*public native void setProperty( String key, int value )*<br>*public native void setProperty( String key, boolean value )* | |
| *Parameters:* | |
| key | The name of the property to set |
| value | The value to assign to the named property |
| *Return Value:none* | |

**Remarks:**
The supported properties depend on the type of wizard object being added:

**WizCtl.objectButton**

1 - Left2 - Right4 - Top8 - Bottom

| **Property** | **Value** |
|---|---|
| Width | Button width in pixels |
| Height | Button height in pixels |
| Align | Alignment of text within button. A combination of the values:<br><br>0 - Vertically / horizontally centred<br>1 - Left<br>2 - Right<br>4 - Top<br>8 - Bottom |
| Invert | true to display button as white text on a black rectangle |

| | |
|---|---|
| Greyed | true to display button greyed-out and disabled |

**WizCtl.objectInputEcho**

| Property | Value |
|---|---|
| CharSet | Single character string specifying character to display as echo (eg, "*" for password entry) |
| Spacing | SpacingSpecifies spacing between echoed characters as follows:<br>0 - No spacing<br>1 - Half character width spacing<br>2 - Single character width spacing<br>4 - Double character width spacing |
| Underline | true to display lines under the position of each character |

**WizCtl.objectRadioButton**

| Property | Value |
|---|---|
| Group | String: Name of group to which a radio button belongs. Required |
| Checked | Boolean: True if the radio button is initially selected.<br>If multiple radio buttons in a group are created with Checked=true, the last one added will be the one selected. |

### close()

"Closes" the object, releasing the underlying COM object (and thus freeing resources).

| |
|---|
| *public native void close()* |
| *Parameters:none* |
| *Return Value:none* |

# class FLSX

This class allows an application to provide a custom loader for the native library.
Full qualification: com.florentis.signature.FLSX

## Summary

| Method |
|---|
| setLoader |

## Methods

### setLoader(ILoader)

| | |
|---|---|
| public static void setLoader(ILoader *customLoader*) | |
| *Parameters:* | |
| *customLoader* | The new loader to use. |
| Return Value: *none* | |

You MUST set your own implementation before creating any Signature objects otherwise it will be too late. Note that System.load can only be called once per libname – it is not possible to call this multiple times with different paths. The code is only called when you first create a Signature object. Possible reasons for using this are for using an alternative to system.library.path to locate the DLL, or to even rename it so that you can have both 32-bit and 64-bit DLLs in the same folder.

For example, the following code replicates the default loader from previous and current versions:

```
com.florentis.signature.FLSX.setLoader(new com.florentis.signature.FLSX.ILoader() { @Override public void
loadLibrary() {
  System.loadLibrary("flsx");
}});
```

## Interface: FLSX.ILoader

Provides the mechanism to replace the standard native library loader. See FLSX.setLoader() for more details.
Full qualification: com.florentis.signature.FLSX.ILoader

### Methods

#### loadLibrary()

This will only be called once. See FLSX.setLoader().

| public void loadLibrary() | |
|---|---|
| *Parameters:* | |
| none | |
| *Return Value: none* | |
| | |