Signature Components Java API

Global Signature Development Team

June 2014

Contents

Doc version: 25.06.2014 11:37

1. Introduction

This document should be read with reference to Signature-Components-API for a detailed description of the classes implemented in Java.

2. 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`

Methods:

aboutBox()

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

Prototype:

```
public static native void aboutBox()
```

close()

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

Prototype:

```
public native void    close();
```

Properties:

signature

The read/write property contains a SigObj class

licence

The read/write property contains a licence string

3. Class: DynamicCapture

The class provides the signature capture functionality

Full qualification: `com.florentis.signature.DynamicCapture`

Methods:

capture()

The method calls signature capture

Prototype:

```
public static native int capture(
  SigCtl sigCtl,
  String who,
  String why,
  Hash   what,
  Key    key
);
public static native int capture(
  SigCtl sigCtl,
  String who,
  String why,
  Hash   what,
  Key    key,
  Object jframe
);
```

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)

jframe
Frame object above which signature capture window should be displayed

Return value:

0	- dynCaptOK	Signature captured successfully.
1	- dynCaptCancel	Signature not captured because user cancelled
100	- dynCaptPadError	No capture service available
101	- dynCaptError	Error - Pad Error

200 - dynCaptAbort Error - unable to parse document contents

close()

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

Prototype:

```
public native void close();
```

Properties:

licence

The read/write property contains a licence string

4. Class: eSeal

The class provides the eSeal capture functionality

Full qualification: `com.florentis.signature.eSeal`

Methods:

capture()

The method inserts an eSeal and optionally captures a handwritten signature

Prototype:

```
public static 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: the following:

eSeal.RequireSignature	- Insert eSeal and capture handwritten signature
eSeal.esNoSignature	- Insert eSeal (without signature capture)
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:

0	- esCaptOK	Signature captured successfully.
1	- esCaptCancel	Signature not captured because user cancelled
100	- esCaptPadError	No capture service available
101	- esCaptError	Error - Pad Error
200	- esCaptAbort	Error - unable to parse document contents
300	- esCaptNoImage	Error - unable to load eSeal image

close()

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

Prototype:

```
public native void     close();
```

Properties:

url

Read/Write String contains the URL of the image in a standard format:
JPEG, PNG, TIF, GIF, BMP

hAlign

Read/Write int defines horizontal alignment of the image as one of:

- eSeal.esLeft
- eSeal.esCentre (default)
- eSeal.esCenter
- eSeal.esRight

vAlign

Read/Write int defines vertical alignment of the image as one of:

- eSeal.esTop
- eSeal.esCentre (default)
- eSeal.esCenter
- eSeal.esBottom

hScale

Read/Write int defines percentage of X dimension (defaults to 100)

vScale

Read/Write int defines percentage of Y dimension (defaults to 100)

transparency

Read/Write int defines percentage of transparency:

- 100 = maximum, not visible
- 0 = minimum, unchanged (default)

cacheImage

Read/Write Boolean false if URL is to be accessed at time of signing, true if image is saved within the eSeal object. Defaults to false. For internal use only.

width

Read-only int width of image in HIMETRIC 0.01mm units

height

Read-only int height of image in HIMETRIC 0.01mm units

name

Read/Write String name for internal use only

id

Read-only String for internal use only

licence

The read/write property contains a licence string

5. 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`

Methods:

add()

The method adds data to the Hash object

Prototype:

```
public static native void add( data );
```

Parameters:

data

The data item can be one of the types:

boolean, byte, char, short, int, long, float, double, String, byte[]

clear()

The method clears the Hash object

Prototype:

```
public static native void clear()
```

close()

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

Prototype:

```
public native void close();
```

Properties:

type

Read/Write value sets the type of hashing algorithm to one of:

- Hash.none
- Hash.md5
- Hash.sha1
- Hash.sha224
- Hash.sha256
- Hash.sha384
- Hash.sha512

hash

Read-only String value

6. Class: Key

The class is used for protecting the integrity of data

Full qualification: `com.florentis.signature.Key`

Methods:

set()

The method sets the type of the Key object

Prototype:

```
public static native void set( int type );
```

Parameters:

type

The key type can be one of the following:

- Key.none
- Key.md5
- Key.sha1
- Key.sha224
- Key.sha256
- Key.sha384
- Key.sha512

close()

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

Prototype:

```
public native void close();
```

Properties:

type

Read-only returns the type of Key set

7. Class: SigObj

The class provides properties and methods for the signature

Full qualification: `com.florentis.signature.SigObj`

Methods:

clear()

The method clears the object

Prototype:
```
public static native void clear();
```

checkIntegrity(Key key);

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

Prototype:
```
public static native int checkIntegrity( Key key );
```

Parameters:

key
Optional Key object. If not supplied the code uses Key type MD5 by default.

Return value:

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(Hash hash);

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

Prototype:
```
public static native int  checkSignedData( Hash hash ) ;
```

Parameters:

hash
Required Hash object to be compared with the one provided when the signature was captured

Return value:

0 - DataGood	Data has not changed since signature capture
1 - DataNoHash	No signature captured, or signature was captured without a hash
2 - DataBadType	Signature was captured with a different type of hash
3 - DataBadHash	Data has changed since signature capture
4 - DataError	Error whilst checking signed data

renderBitmap()

The method renders a signature to a file or byte array

Prototype:

```
public static native byte[] renderBitmap(
  String outputFilename,
  int    dimensionX,
  int    dimensionY,
  String mimeType,
  float  inkWidth,
  int    inkColor,
  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
dimensionY
X/Y dimensions specified as DPI (dots per inch) or Pixels.
Negative value = DPI (the signature is not scaled)
Positive value = Pixels (the signature is scaled to the dimensions)

mimeType
Specifies the image format as one of:
image/bmp
image/jpeg
image/gif
image/tiff
image/png

inkWidth
Specifies the signature ink width in mm

inkColor
inkBackground

Specifies the pen ink and background colours in MS COM COLORREF format (BGR)
Examples:

cWhite = 0xFFFFFF
cBlack = 0x00
cRed = 0x0000FF

paddingX
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.

X/Y dimensions are specified as mm or Pixels.

Negative value = mm (1inch = 25.4mm)
Positive value = Pixels

flags
A bit mask of the following categorised values:

Output Group: (single value)

SigObj.outputBinary	image is returned as a byte array
SigObj.outputBase64	image is returned as a Base 64 encoded string
SigObj.outputFilename	outputFilename contains the pathname of the file to be created

Color selection Group: (single value)

SigObj.color1BPP	1 bit per pixel
SigObj.color24BPP	24 bit per pixel
SigObj.color32BPP	32 bit per pixel

Optional image format flags:

SigObj.backgroundTransparent	transparent background
SigObj.colorAntiAlias	option with 24 and 32 BPP

Optional Image extension:

SigObj.encodeData	Encode signature data within image
SigObj.watermark	Include watermark within image to indicate presence of encoded data

Return Value:
If flags include `outputBinary or outputBase64` then the generated image data is returned. Otherwise null is returned.

Example

```
try
 {
  com.florentis.signature.SigObj sig =
              new com.florentis.signature.SigObj();

  sig.sigText( readFileAsString("..\\data\\JS1.txt"));

  sig.renderBitmap("..\\temp\\1.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

Prototype:

```
public static native void renderRect(
      long hdcTarg,
      long hdcRef,
      int  left,
      int  top,
      int  right,
      int  bottom,
      float inkWidth,
      int inkColor,
      int option,
      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. May be the same as hDCTarg.

left, top, right, 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. (Default is 0.7mm.)

InkColor
Optional int OLE_COLOR value specifying colour of pen used to draw signature. (Default is black.)

Option
Optional int value specifying the scaling mode of the rendered signature, with possible values:

0 - dspForceFit scale signature to exactly fit the bounding rectangle (default)
1 - dspUseZoom scale signature according to the Zoom argument.
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()

Prototype:

```
public static native int readEncodedBitmap( String filename );
```

Parameters:
filename
Required string *contains the* pathname of the image file containing the encoded SigObj

Return Value:

0 - readEncodedBitmapOK	Signature data decoded OK
1 - readEncodedBitmapFileNotFound	File not found
2 - readEncodedBitmapNotImage	Bitmap is not a supported image type
3 - readEncodedBitmapSigDataNotFound	Encoded signature data not found

close()

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

Prototype:
public native void close();

Properties:

additionalData

additionalData (int captData) returns additional capture data eg pad driver version

crossedOut

Read-only Boolean value is true if the signature appears crossed out as invalid

extraData

extraData(String key) Write once, Read string value referenced by key name or "" for all values

height

Read-only int value of the bounding rectangle of the signature in 0.01mm

ink

Read/Write CIC Ink Tools interface value

isCaptured

Read-only boolean value indicates if a signature has been captured

sigData

Read/Write binary SigObj data

sigText

Read/Write hex string representation of sigData

who

Read-only string containing signatory name

why

Read-only string containing reason for signing

when

when(int timeZone) returns the time & date of signature capture
when(0) returns TlimeLocal, when(1) returns TimeGMT

width

Read-only int value of the bounding rectangle of the signature in 0.01mm

8. 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`

Enumeration values

ObjectType:

objectText
objectButton
objectCheckbox
objectSignature
objectInput
objectInputEcho

PrimitiveType:

primitiveLine
primitiveRectangle
primitiveEllipse

AlignmentType:

textAlignLeft
textAlignRight
textAlignCentre
textAlignJustify

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()

Prototype:

```
public native boolean  padConnect();
```

padDisconnect()

Prototype:

```
public native void     padDisconnect();
```

reset()

Prototype:

```
public native void     reset();
```

addObject()

Prototype:

```
public native void     addObject(int type, String id,
                                 Object x, Object y,
                                 Object data, Object options);
```

addPrimitive()

Prototype:

```
public native void     addPrimitive(int type,
                                    Object x1, Object y1,
                                    Object x2, Object y2,
                                    Object primdata, Object options);
```

getObjectState()

Prototype:

```
public native Object   getObjectState(String id);
```

setFont()

Prototype:
public native void setFont(Font font);

setEventHandler()

Prototype:
public void setEventHandler(WizCtlEvents handler)

display()

Prototype:
public native void display();

fireClick()

Prototype:
public native void fireClick(String id);

processEvents()

Prototype:
public native void processEvents() throws InterruptedException;

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

endProcessEvents()

Terminates endProcessEvents()

Prototype:
public native void 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.

close()

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

Prototype:
public native void close();

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.

Properties:

inkingPad

Read-only boolean, True if the pad has a supported LCD display

enableWizardDisplay

Read-Write boolean enables/disables wizard controls

padWidth

Read-only int width of pad display in pixels

padHeight

Read-only int height of pad display in pixels

zoom

Read-Write float scaling of pad, 100 = 100%

licence

The read/write property contains a licence string

9. 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()

Prototype:

```
boolean onPadEvent(WizCtl wizCtl, String id, Object eventType);
```

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

10. Class: InputObj

The class provides the Input control for PIN code input

Full qualification: `com.florentis.signature.WizCtl.InputObj`

Methods:

clear()

Prototype:

```
public native void     clear();
```

setEncrytion()

Prototype:

```
public native void     setEncryption(int type, byte[] key);
```

close()

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

Prototype:

```
public native void     close();
```

Properties:

minLength
Read-Write int minimum number of digits in the PIN

maxLength
Read-Write int maximum number of digits in the PIN

text
Read-only String containing the input data (optionally encrypted)

data
Read-only byte[] containing the input data

encryptionType
Read-only int returns the type set by setEncryption

11. Class: Font

The class is used when setting the display font

Full qualification: `com.florentis.signature.WizCtl.Font`

Methods:

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

Prototype:
```
public native void close();
```

Properties:

name
Read-Write String value

size
Read-Write double value

bold
Read-Write boolean value

underline
Read-Write boolean

strikethrough
Read-Write boolean value

weight
Read-Write short value

12. Class: ObjectOptions

The class is used to set options with the addObject method

Full qualification: `com.florentis.signature.WizCtl.ObjectOptions`

Methods:

setProperty()

Prototype:
public native void setProperty(String key, String value);
public native void setProperty(String key, int value);
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

Remarks:
Supported properties depend on the type of wizard object being added

WizCtl.objectButton

Width	Button width in pixels
Height	Button height in pixels
Align	Alignment of text within button. A combination of the values 0 - Vertically / horizontally centred 1 - Left 2 - Right 4 - Top 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

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

close()

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

Prototype:

```
public native void     close();
```