

Signature Library - COM 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 developer needs to use the components.

- [Introduction](#)
- [ActiveX Components](#)
 - [FISigCOM](#)
 - [FISigCapt](#)
 - [FIWizCom](#)
 - [FISigCOM: SigObj](#)
 - [Summary](#)
 - [Methods](#)
 - [CheckIntegrity](#)
 - [CheckSignedData](#)
 - [Clear](#)
 - [GetProperty](#)
 - [Picture](#)
 - [ReadEncodedBitmap](#)
 - [RenderBitmap](#)
 - [RenderRect](#)
 - [SetProperty](#)
 - [Properties](#)
 - [Properties supported by GetProperty/SetProperty](#)
 - [FISigCOM: SigCtl](#)
 - [Summary](#)
 - [Methods](#)
 - [AboutBox](#)
 - [Capture](#)
 - [GetProperty](#)
 - [SetProperty](#)
 - [Properties](#)
 - [Properties supported by GetProperty/SetProperty](#)
 - [Events](#)
 - [Click](#)
 - [DblClick](#)
 - [KeyDown](#)
 - [KeyPress](#)
 - [KeyUp](#)
 - [MouseDown](#)
 - [MouseMove](#)
 - [MouseUp](#)
- [FISigCOM: SigCtlXHTML](#)
 - [Summary](#)
 - [Methods](#)
 - [CheckHostDocument](#)
 - [CheckHostDocument2](#)
 - [InsertMarkup](#)
 - [Properties](#)
 - [Properties supported by GetProperty/SetProperty](#)
 - [Events](#)
- [FISigCOM: ImgCtlXHTML](#)
 - [Summary](#)
 - [Methods](#)
 - [Properties](#)
- [FISigCOM: Hash](#)
 - [Summary](#)
 - [Methods](#)
 - [Add](#)
 - [Clear](#)
 - [Properties](#)
- [FISigCOM: Key](#)
 - [Summary](#)
 - [Methods](#)
 - [Set](#)
 - [Properties](#)
- [FISigCOM: Enumeration Types](#)
 - [BorderStyle](#)
 - [CaptData](#)

- CaptureData
- CaptureResult
- DisplayMode
- HashType
- IntegrityStatus
- KeyType
- RBFlags
- ReadEncodedBitmapResult
- ShowText
- SignedData
- TimeZone
- FLSigCapt
 - FLSigCapt: DynamicCapture
 - Summary
 - Methods
 - Capture
 - GetProperty
 - SetProperty
 - Properties
 - Properties supported by GetProperty/SetProperty
 - FLSigCapt: eSeal
 - Summary
 - Methods
 - Capture
 - GetProperty
 - SetProperty
 - Properties
 - Properties supported by GetProperty/SetProperty
 - FLSigCapt: Enumeration Types
 - DynamicCaptureResult
 - eSealCaptureMode
 - eSealCaptureResult
 - eSealHAlign
 - eSealVAlign
- FIWizCom
 - FIWizCom: WizCtl
 - Summary
 - Methods
 - AddObject
 - AddObject(ObjectText)
 - AddObject(ObjectButton)
 - AddObject(ObjectCheckbox)
 - AddObject(ObjectRadioButton)
 - AddObject(ObjectSignature)
 - AddObject(ObjectInput)
 - AddObject(ObjectInputEcho)
 - AddObject(ObjectHash)
 - AddObject(ObjectImage)
 - AddObject(ObjectDisplayAtShutdown)
 - AddObject(ObjectInking)
 - AddPrimitive
 - Display
 - FireClick
 - GetObjectState
 - GetProperty
 - PadConnect
 - PadDisconnect
 - Reset
 - SetEventHandler
 - SetProperty
 - Properties
 - Properties supported by GetProperty/SetProperty
 - FIWizCom: InputObj
 - Summary
 - Methods
 - Clear
 - SetEncryption
 - Properties
 - FIWizCom: ObjectOptions
 - Summary
 - Methods
 - GetProperty
 - SetProperty
 - Properties
 - Properties supported by ObjectButton
 - Properties supported by ObjectInputEcho
 - Properties supported by ObjectRadioButton

- FIWizCom: Enumeration Types
 - BorderStyle
 - ButtonOptions
 - CheckboxOptions
 - EncryptAlg
 - EventType
 - InputEchoOptions
 - MarkupStatus
 - ObjectType
 - PrimitiveOptions
 - PrimitiveType
 - TextOptions
- Configuration (Registry Settings)
 - Signature Capture Dialog Settings
 - SigCom DLL Settings
 - SigCapt DLL Settings
 - WizCOM DLL Settings
- Signature Data Encryption
 - Overview
 - Setting an Encryption Key
 - Key Generation using OpenSSL
 - Encryption Status
- Signature ISO Format
 - Overview
 - Implementation
 - API
 - FISigCOM: : AdditionalImportIsoData
 - FISigCOM: ISO methods
 - ImportIsoData
 - ExportIsoData

ActiveX Components

FISigCOM

Classes
SigObj
SigCtl
SigCtlXHTML
ImgCtlXHTML
Hash
Key
Enumerator Types
BorderStyle
CaptData
CaptureData
CaptureResult
DisplayMode
HashType
IntegrityStatus
KeyType
MarkupStatus
RBFlags

ReadEncodedBitmapResult
ShowText
SignedData
TimeZone

FlSigCapt

Classes
DynamicCapture
eSeal
Enumerator Types
DynamicCaptureResult
eSealCaptureMode
eSealCaptureResult
eSealHAlign
eSealVAlign

FlWizCom

Classes
WizCtl
InputObj
Enumerator Types
BorderStyle
ButtonOptions
CheckboxOptions
EncryptAlg
EventType
InputEchoOptions
ObjectType
PrimitiveOptions
PrimitiveType
TextOptions

FlSigCOM: SigObj

Summary

Method
CheckIntegrity
CheckSignedData
Clear
GetProperty
Picture
ReadEncodedBitmap
Render
RenderBitmap
RenderRect
SetProperty
Property
AdditionalData
CrossedOut
ExtraData
Height
Ink
IsCaptured
SigData
SigText
When
Who
Why
Width

The SigObj object encapsulates a captured handwritten signature. When populated it contains a wealth of data that fully describes both the static and dynamic characteristic of a signature and the context in which the signature was captured. A Signature object can be bound (at the moment of capture) to the host document or other data set to provide a means of determining whether or not any changes have been made subsequently, either maliciously or unintentionally.

Methods

CheckIntegrity

Checks the integrity of the Signature object to detect whether it has been tampered with since signing.

<i>IntegrityStatus CheckIntegrity(Key)</i>	
<i>Parameters:</i>	
Key	Optional Key object. If not supplied the code uses Key type MD5 by default.
<i>Return Value: IntegrityStatus</i>	
IntegrityOK	Data has not changed since signature capture
IntegrityFail	Data has changed since signature capture
IntegrityMissing	Signature integrity value not found

IntegrityWrongType	Signature was captured using a different integrity check version
--------------------	------------------------------------------------------------------

CheckSignedData

Checks for a match between a given hash and that provided when the signature was captured. If the signature is bound to a hash (ie. the What argument of the DynamicCapture or SigCtl Capture method was set), then this function checks the signed data against that from which the supplied hash was derived.

This method is called by SigCtlXHTML.CheckHostDocument() which supplies the necessary Hash.

<i>SignedData CheckSignedData(Hash)</i>	
<i>Parameters:</i>	
Hash	Hash object, to be compared with that provided when the signature was captured.
<i>Return Value: SignedData</i>	
DataGood	The supplied hash matches that provided when the signature was captured, thus both Hash objects have been derived from the same data set.
DataNoHash	No hash was specified when the signature was captured (or the object does not contain a signature).
DataBadType	The supplied hash is of a different type (HashType Enum) to that provided when the signature was captured.
DataBadHash	The supplied hash value does not match that provided when the signature was captured, thus the two Hash objects have been derived from different data sets.

Clear

Clears all signature data and thus reinitializes the Signature object.

<i>Void Clear()</i>	
<i>Parameters: none</i>	
<i>Return Value: none</i>	

GetProperty

Returns the current value of a property.

<i>void GetProperty(Name)</i>	
<i>Parameters:</i>	
Name	The name of the property to retrieve
<i>Return Value: Variant</i>	
Variant	Current value of the named property. Empty if no value has been set.

Picture

Renders an image of the signature, creating a Windows Picture object. Optionally encodes the signature data in the generated image using steganographic techniques.

<i>variant Picture(dimensionX, dimensionY, mimeType, inkWidth, nkColor, backgroundColor, paddingX, paddingY, flags)</i>	
<i>Parameters</i>	
See RenderBitmap	As RenderBitmap hardcoded for RenderOutputPicture
<i>Return Value: variant function dependent</i>	
See RenderBitmap	Picture returned as specified by Flags

ReadEncodedBitmap

Reads the encoded SigObj data from an image file, picture object, base-64 encoded string or binary data which was created using `RenderBitmap()`.

<i>ReadEncodedBitmapResult ReadEncodedBitmap(data)</i>	
<i>Parameters</i>	
Data	<p>encoded signature data in one of the following formats: String pathname of an image file or base-64 encoded string. Strings containing these characters are assumed to be filenames:</p> <p>' : ' , ' \ ' or ' . '</p> <p>Otherwise the string is assumed to be one of:</p> <ul style="list-style-type: none"> • base-64 encoded image data • Picture Windows Picture object • Byte Array binary image data
<i>Return Value: ReadEncodedBitmapResult</i>	
ReadEncodedBitmapOK	The Signature data was decoded successfully
ReadEncodedBitmap-NotImage	The Bitmap is not a supported image type
ReadEncodedBitmap-SigDataNotFound	Encoded signature data was not found in image

RenderBitmap

Renders an image of the signature, creating an image file or returning the binary data. Optionally encodes the SigObj data in the generated image using steganography.

<i>variant RenderBitmap(outputFilename, dimensionX, dimensionY, mimeType, inkWidth, nkColor, backgroundColor, paddingX, paddingY, flags)</i>	
<i>Parameters:</i>	
outputFilename	Pathname of the file in which to save the image. May be null or empty if output type specified by flags is not <code>RenderOutputFilename</code> .
dimensionX dimensionY	Dimensions specified as DPI (dots per inch) or Pixels. <ul style="list-style-type: none"> • Negative value = DPI (the signature is not scaled) • Positive value = Pixels (the signature is scaled to fit the area)
mimeType	Specifies the image format as one of: <ul style="list-style-type: none"> • image/bmp • image/tiff • image/png
inkWidth	Specifies the signature ink width in mm
inkColor inkBackground	Signature ink and background colours in OLE_COLOR format
paddingX paddingY	Padding around the signature image, added to both the left and right for paddingX, and both the top and bottom for paddingY. Dimensions are specified as mm or Pixels. <ul style="list-style-type: none"> • Negative value = mm • Positive value = Pixels

Flags	<p>A combination of RBFlags values. A single value from each mandatory group must be included:</p> <p>Output Group (mandatory):</p> <ul style="list-style-type: none"> • RenderOutputBase64Return base-64 encoded string • RenderOutputBinaryReturn bitmap as binary data • RenderOutputFilenameWrite bitmap to file • RenderOutputPictureReturn bitmap as Picture object <p>Colour selection Group (mandatory):</p> <ul style="list-style-type: none"> • RenderColor1BPPRender 1 bit monochrome • RenderColor24BPPRender 24 bit pixel colour • RenderColor32BPPRender 32 bit pixel colour <p>Image format flags (optional):</p> <ul style="list-style-type: none"> • RenderBackgroundTransparent Make background transparent • RenderColorAntiAliasRender colour image with antialiasing (24 and 32 BPP only) <p>Image extension (optional):</p> <ul style="list-style-type: none"> • RenderEncodeData signature data within image • RenderWatermarkInclude watermark within image to indicate presence of encoded data <p>Other (optional):</p> <ul style="list-style-type: none"> • RenderClippedCrop signature image, omitting any parts which were outside of the original capture window. • RenderRelativeRenders the signature image relative to the origin of the original capture window. (Dimensions must be equal DPI (negative) values; padding values must be 0)
<i>Return Value: variant function dependent</i>	
Filename	No return value
Binary	Byte array containing the image file contents
Base64	String containing base-64 representation of image file data
Picture	Windows picture object (as an IDispatch interface pointer)

RenderRect

Renders an image of the signature within a given rectangle on a specified device context.

<i>void RenderRect(hDCTarg, hDCRef, Left, Top, Right, Bottom, InkWidth, InkColor, Option, Zoom, Rotation)</i>	
<i>Parameters:</i>	
hDCTarg	Handle to output device context (as a LONG).
hDCRef	Handle to reference device context (as a LONG). May be the same as hDCTarg.
Left, Top, Right, Bottom	Integer values defining the bounding rectangle in which the signature is to be rendered.
InkWidth	<p>Float value specifying width, in mm, of pen used to draw signature. Note that, for signatures captured on devices which record pressure, width varies to indicate pressure and also that width scales with Zoom,</p> <p>e.g. if InkWidth=0.7 and Zoom=200, actual width will be 1.4mm. (Optional, default value 0.7mm.)</p>
InkColor	OLE_COLOR value specifying colour of pen used to draw signature. (Optional, default is black.)
Option	<p>DisplayMode value specifying the scaling mode of the rendered signature, with possible values:</p> <ul style="list-style-type: none"> • DspForceFit – scale signature to exactly fit the bounding rectangle • DspUseZoom – scale signature according to the Zoom argument • DspBestFit – reduce size of signature to fit area if it is too big, <p>Otherwise show at true size. (Optional , default is DspForceFit.)</p>

Zoom	Percentage by which the signature image is to be scaled. If 'Option' is DspForceFit then 'Zoom' is ignored. If 'Option' is DspUseZoom and the value of 'Zoom' results in a signature larger than the given rectangle then the signature image is not clipped. (Optional, default is 100%.)
Rotation	Short value specifying the angle, measured anticlockwise in degrees, through which the signature image is to be rotated. (Optional, default is 0.) Note: this parameter is currently ignored.
<i>Return Value: none</i>	

SetProperty

Sets the value of a named additional property, overrides a value set in the registry.

<i>void SetProperty(Name, Value)</i>	
<i>Parameters:</i>	
Name	String. Name of the property
Value	Value to assign to property
<i>Return Value: Boolean</i>	
True	Name exists and value is valid
False	Failed to set property

Properties

Property	Type	SigObj Property Description
AdditionalData (CaptData)	Variant	Returns additional data collected at capture time. CaptData parameter is an Enum which specifies the capture related data to return, see Enumerator Types for possible values. Returns requested data as a string; or an empty variant if the object does not contain signature data, or the signature data does not include the requested type of data. (Read-only)
CrossedOut	Boolean	Returns TRUE if the signature has been invalidated by changes to the document and appears crossed out. (Read-only)
ExtraData (Key)	String	ExtraData is a parameterized property that allows the client to store additional data within the signature object after capture. For example, if a signature is being manually validated the system may find it convenient to store the result in the signature itself, rather than as an independent data item. Each ExtraData item must be given an identifying key name and an associated value. There can be only one value for each key and once written the key cannot be modified or removed. All ExtraData key pairs are protected by the signature object integrity hash. e.g. to set a new data item: sig.ExtraData("Validation") = "Passed"; When the property is read: =sig.ExtraData(""); all values are returned in a single string in the format: "key=value;key=value;"
Height	Int	Returns the height of the bounding rectangle of the signature in 0.01mm units. Returns 0 if the object does not contain signature data. (Read-only)
Ink	String	Signature data in a form compatible with the ActiveX interface of CIC InkTools, thus allowing interoperation between the Signature Component and CIC InkTools. Signature data may be exchanged by copying the string value to and from a CIC Ink control via its Ink property.

IsCaptured	Boolean	Returns True if the object contains signature data. Note that IsCaptured = True does not necessarily imply that the When, Who and Why properties are valid. In some circumstances they may be invalid even though the Signature object contains valid signature data. For example, this will be the case if the signature data has been imported from a CIC Ink control via the Ink property. (Read-only)
SigData	VARIANT (containing Byte())	Binary signature data as an array of bytes. May be used to save the signature data to and restore it from a file or database.
SigText	String	Binary signature data as a string, base-64 encoded or hexadecimal. On Read the format is determined by the get_SigText registry setting
When(TimeZone)	DateTime	Returns the time & date of signature capture. Returns 0 if the object does not contain signature data, or the signature data does not contain the time of capture (e.g. if the object was initialized using the Ink property). TimeZone parameter is an Enum which specifies the time-zone to use (local time or GMT/UTC), see Enumerator Types for possible values. Default = 'TimeLocal'. (Read-only)
Who	String	Returns the name of signatory, as specified at time of signature capture. Returns an empty string if the object does not contain signature data, or the signature data does not contain a signatory name e.g. if the object was initialized using the Ink property). (Read-only)
Why	String	Returns the reason for signing, as specified at time of signature capture. Returns an empty string if the object does not contain signature data, or the signature data does not contain a reason for signing e.g. if the object was initialized using the Ink property. (Read-only)
Width	Int	Returns the width of the bounding rectangle of the signature in 0.01 mm units. Returns 0 if the object does not contain signature data. (Read-only)

Properties supported by SetProperty/SetProperty

Name	Get/Set	Type	Description
Licence	Get/Set	String	String containing the license used for signature capture
Component_FileVersion	Get	String	Component version e.g. "4.5.2.210"
Complexity	Get	Double	Signature complexity between 0..1 e.g. 0.5343597732
ExtraDataIntegrity	Get/Set	KeyType	KeyType used to verify integrity of ExtraData
Integrity_CAPICOM_ICertificates	Get	Certificates	If the signature was captured with a Key of KeyType KeyCAPICOM, returns a collection of (one or more) Certificate objects
Integrity_KeyType	Get	KeyType	KeyType used in signature capture
RenderClipped	Get/Set	Boolean	True to clip the output to what was seen within the capture window
SignedData_HashType	Get	HashType	HashType enumerator used by the hash object
UILanguage	Get/Set	String	Selects the language used in the signature capture dialog from: de, el, en, es, fr, it, ja, ko, nl, pl, pt-BR, ru, zh-CN, zh-TW The codes conform to the ISO language codes expanded here: http://www.lingoes.net/en/translator/langcode.htm The codes match the language folders installed as part of the Signature SDK in: C:\Program Files (x86)\Common Files\WacomGSS To select a language, set the property value to a language code, e.g. obj.setProperty("UILanguage", "it"); The language code is not case-sensitive.

Where	Get	String	Returns the location at which the signature was captured (if set DynamicCapture.SetProperty)
			For the following values see the section Signature Data Encryption
IsEncrypted	Get	Boolean	TRUE if the SigObj contains encrypted signature data which has not (yet) been decrypted.
CanEncrypt	Get	Boolean	TRUE if a password or public key has been set. If the SigObj already contains sig data, or if it is subsequently set (via SigData, SigText, Capture etc) it will be encrypted.
CanDecrypt	Get	Boolean	TRUE if a password or private key has been set. Sig data encrypted (with the appropriate password or public key) can be set and will be decrypted.
EncryptionPassword	Set	String	String containing symmetric encryption key
EncryptionPublicKey	Set	String	String containing asymmetric public key
EncryptionPublicKeyFile	Set	String	String containing asymmetric public key filename
EncryptionPrivateKey	Set	String	String containing asymmetric private key
EncryptionPrivateKeyFile	Set	String	String containing asymmetric private key filename

FSigCOM: SigCtl

Summary

Method
AboutBox
Capture
GetProperty
SetProperty
Property
AppData
BackColor
BackStyle
BorderColor
BorderStyle
BorderVisible
BorderWidth
Caption
CtlPadding
DisplayMode
Enabled
Font
ForeColor
InkColor
InkWidth
InputData
InputSignature

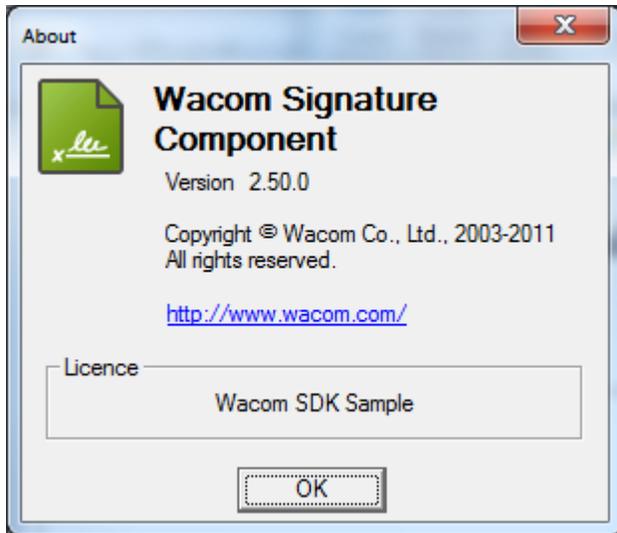
InputWho
InputWhy
Licence
Rotation
ShowWhen
ShowWho
ShowWhy
Signature
TabStop
WhenFormat
Zoom
Event
Click
DbClick
KeyDown
KeyPress
KeyUp
MouseDown
MouseMove
MouseUp
PostCapture
PreCapture
Remove

Note that SigCtlXHTML extends this class and includes common methods and properties

Methods

AboutBox

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



Void AboutBox()

Parameters:none

Return Value: none

Capture

This method is deprecated; the DynamicCapture object should instead be used.

Displays a Signature Capture Window in which a user may sign his or her name using a digitizing tablet. The window contains three buttons by which the user can either clear the currently rendered signature (Clear button) or dismiss the window without capturing a signature (Cancel button) or dismiss the window and capture the currently rendered signature (OK button).

CaptureResult Capture(Who, Why, What, Key)

Parameters:

Who	String specifying the signatory name. Optional if InputWho property is set
Why	String specifying the reason for signing. Optional if InputWhy property is set
What	Hash object Derived from the data set with which the captured signature is to be bound. Typically the data set will be the contents of the document or form signed by the user. SigCtlXHTML: The hash is generated automatically from the contents of the XHTML document and cannot be supplied here. (Optional)
Key	Key object supplied in order to detect malicious or accidental tampering with the signature object or the signature data it contains. SigCtlXHTML: Key type SHA256 is supplied automatically but an alternative can be specified. (Optional)

Return Value: CaptureResult

CaptureOK	Signature captured successfully.
CaptureCancel	Signature not captured because user cancelled Signature Capture Window.
CapturePadError	No capture service available; typically, no functioning digitizing tablet or other device for capturing signatures found.

CaptureNotLicensed	The component has not been licensed to perform capture. This may be because a suitable licence has not been set or because a condition of the licence has not been met. For example, if no suitable hardware was found.
CaptureError	System error or some other unusual error condition.
CaptureAbort	Signature not captured because the handler for the PreCapture event returned False. With SigCtlXHTML, this can indicate a failure to parse document contents

GetProperty

Returns the current value of a property.

<i>void GetProperty(Name)</i>	
<i>Parameters:</i>	
Name	The name of the property to retrieve
<i>Return Value:Variant</i>	
Variant	Current value of the named property. Empty if no value has been set.

SetProperty

Sets the value of a named additional property, overrides a value set in the registry.

<i>void SetProperty(Name, Value)</i>	
<i>Parameters:</i>	
Name	String. Name of the property
Value	Value to assign to property
<i>Return Value:Boolean</i>	
True	Name exists and value is valid
False	Failed to set property

Properties

Property	Type	SigCtl Property Description
AppData(Key)	Variant	Application data identified by Key Parameterised property supporting key/value pairs for the storage of any application specific data. For example, may be used to store a signatory name and reason for signing in the control prior to the signature being captured. The Key may be an integer or string e.g. two separate keys: sig.AppData(0)="a" sig.AppData("0")="b" AppData may be initialised in XHTML using the AppData param name, e.g. <param name="AppData" value="key1=value1;key2=value2;etc=etc"/>
BackColor	OLE_COLOR	Colour of control (Signature Area) background. Default = background colour of container if available, or colour of system window.

BackStyle	Int	Used to determine whether the control is transparent or opaque. 0 = Opaque 1 = Transparent Default: Opaque
BorderColor	OLE_COLOR	Colour of control (Signature Area) border if 'BorderStyle' is 'flat'. Ignored for other values of 'BorderStyle'. Default = colour of system window frame.
BorderStyle	Int	Style of control (Signature Area) border. While any long value can be set, the BorderStyle enumerator includes the most useful values. Default = 0 (BdrFlat)
BorderVisible	Boolean	Shows or hides the control (Signature Area) border: True is visible, False is not visible. Default = True
BorderWidth	Int	Width in pixels of control (Signature Area) border if 'BorderStyle' is 'flat'. Ignored for other values of 'BorderStyle'. Default = 1
Caption	String	Text centrally displayed in the control (Signature Area) when no signature has been captured as an alternative to the default signature guideline. Default = Empty string
CtlPadding	Short	Size, in pixels, of space between the border of the control (Signature Area) and the bounding rectangle of the rendered signature if 'DisplayMode' is 'DspForceFit'. Ignored if 'DisplayMode' is 'DspUseZoom'. Default = 2
DisplayMode	DisplayMode (Enum)	Specifies scaling mode of rendered signature, with possible values: <ul style="list-style-type: none"> • DspForceFit scale signature to exactly fit the control (Signature Area). • DspUseZoom scale signature (relative to its actual size) according to the 'Zoom' property. • DspBestFit reduce signature to fit in box if too big, otherwise show at true size. <p>In each case, the image is centred in the control. If scaling is chosen and the value of the 'Zoom' property results in an image larger than the control area, it will be clipped. Default = 'DspForceFit'.</p>
Font	IFontDisp	Font used to display the Caption property (if set) and the When/Who text (see ShowWhen & ShowWho properties). Default = container font, if available, or 8 point MS Sans Serif.
ForeColor	OLE_COLOR	Colour of the default signature guideline or (if set) the Caption property text, and the colour of any When/Who text displayed in the Signature Area (see ShowWhen & ShowWho properties). Default = foreground colour of container, if available, or system window text colour. Note: the colour of the rendered signature is controlled independently by the InkColor property.
InkColor	OLE_COLOR	Colour of 'ink' used to render signature image. Default = system window text colour
InkWidth	Float	Width, in mm, of 'ink' used to render signature image. Default = 0.7mm Note: <ul style="list-style-type: none"> • min = 0.1mm • max = 2.0 mm • scales with signature • varies with pressure (if pressure data is available)
InputData	String	Allows supplementary data to be stored in the signature. The data is defined in the form of "key=value" pairs separated by ';' e.g. "Account=12345;Branch=AB12" When the signature is captured the InputData values are transferred and become available as the ExtraData in the ISigObjXHTML interface. Key values can be changed by re-setting the parameter, and can be cleared by setting the value to a blank string.
InputSignature	String	Binary Signature SigObj data, base-64 encoded or as a string of hexadecimal digits. On Read the format is determined by the get_SigText registry setting
InputWho	String	Used to pre-set the name of the person who is to sign. This value is then used if the 'Who' parameter in the Capture method is not defined.

InputWhy	String	Used to pre-set the reason for signing text. This value is then used if the 'Why' parameter in the Capture method is not defined.
Licence	Variant	Licence object or string
Rotation	Short	Drawing angle. Note: Signature rotation is not currently supported. Setting any value other than 0 will cause an error.
ShowWhen	ShowText (Enum)	Specifies whether and how to display time of signing within the Signature Area. See Enumerator Types for possible values. Default = 'TxtDontShow' Note: the entire area of the control is used to render the signature image, thus it is possible for the When text to be overwritten.
ShowWho	ShowText (Enum)	Specifies whether and how to display signatory name within the Signature Area. See Enumerator Types for possible values. Default = 'TxtDontShow' Note: the entire area of the control is used to render the signature image, thus it is possible for the Who text to be overwritten.
ShowWhy	ShowText (Enum)	Specifies whether and how to display the reason for signing text within the Signature Area. See Enumerator Types for possible values. Default = 'TxtDontShow' Note: the entire area of the control is used to render the signature image, thus it is possible for the Why text to be overwritten.
Signature	SigObj	Signature object created by the Signature control. The SigCtl owns the SigObj – both are generally created and destroyed together. However, the SigCtl may be outlived by the SigObj if the client application retains an outstanding object reference. Property put is by value, rather than by reference, so that an assignment in Visual Basic takes the form: SigCtl.Signature = SigObj rather than: Set SigCtl.Signature = SigObj However, the converse assignment takes the form: Set SigObj = SigCtl.Signature
WhenFormat	String	Format used to display time of signing (a Win32-style date/time format string). Default = HH':'mm dd MMM yyyy
Zoom	Short	Scaling factor (expressed as a percentage of actual size) for rendering the signature image when 'DisplayMode' is 'DspUseZoom'. Ignored if 'DisplayMode' is 'DspForceFit'. Default = 100

Properties supported by SetProperty/SetProperty

Name	Get/Set	Type	Description
Licence	Get/Set	String	String containing the license used for signature capture
Component_FileVersion	Get	String	Component version e.g. "4.5.2.210"
DisablePropertyChangePostCapture	Get/Set	Boolean	Disables changing control properties after a signature has been captured
ShowWhere	Get/Set	ShowText	Specifies whether and how to display the location of signing within the Signature Area. See Enumerator Types for possible values.
UILanguage	Get/Set	String	Selects the language used in the signature capture dialog from: For details (see SigObj properties)
hWnd	Get/Set	Variant	Windows handle used to display the control

Events

Click

Occurs when the user clicks (presses and then releases) a mouse button over a SigCtl

```
Private Sub object_Click()
```

Parameters:none

Return Value: none

DbClick

Occurs when the user double-clicks a mouse button over a SigCtl

Private Sub object_DbClick()

Parameters:none

Return Value: none

KeyDown

Occurs when the user presses a key while a SigCtl has the focus.

Private Sub object_KeyDown(KeyCode, Shift)

Parameters:

KeyCode	Short integer code for the key pressed. KeyCode is passed by reference; changing it sends a different code to the control (however the control does not process key presses so this has no effect).
Shift	Short integer bit field indicating the state of the SHIFT (bit 0), CTRL (bit 1) and ALT (bit 2) keys at the time of the event. These bits correspond to the values 1, 2 and 4 respectively. Some, all or none of the bits can be set, indicating that some, all or none of the keys are down.

Return Value: none

KeyPress

Occurs when the user presses and releases an ANSI key while a SigCtl has the focus.

Private Sub object_KeyPress(KeyAscii)

Parameters:

KeyAscii	A short integer that returns a standard numeric ANSI keycode. KeyAscii is passed by reference; changing it sends a different character to the control (however the control does not process key presses so this has no effect).
----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Return Value: none

KeyUp

Occurs when the user releases a key while a SigCtl has the focus.

Private Sub object_KeyUp(KeyCode, Shift)

Parameters:

KeyCode	Short integer code for the key released. KeyCode is passed by reference; changing it sends a different code to the control, however the control does not process key presses so this has no effect.
Shift	Short integer bit field indicating the state of the SHIFT (bit 0), CTRL (bit 1) and ALT (bit 2) keys at the time of the event. These bits correspond to the values 1, 2 and 4 respectively. Some, all or none of the bits can be set, indicating that some, all or none of the keys are down.

Return Value: none

MouseDown

Occurs when the user presses a mouse button over a SigCtl.

<i>Private Sub object_MouseDown(Button, Shift, X, Y)</i>	
<i>Parameters:</i>	
Button	Short integer that identifies the button that was pressed. The button argument is a bit field with bits corresponding to the left button (bit 0, value = 1), right button (bit 1, value = 2), and middle button (bit 2, value = 4). Only one of the bits is set, indicating the button that caused the event.
Shift	Short integer bit field indicating the state of the SHIFT (bit 0), CTRL (bit 1) and ALT (bit 2) keys at the time of the event. These bits correspond to the values 1, 2 and 4 respectively. Some, all or none of the bits can be set, indicating that some, all or none of the keys are down.
X, Y	Current coordinates of the mouse pointer.
<i>Return Value: none</i>	

MouseMove

Occurs when the user moves the mouse over a SigCtl.

<i>Private Sub object_MouseMove(Button, Shift, X, Y)</i>	
<i>Parameters:</i>	
Button	Short integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). It indicates the complete state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are pressed
Shift	Short integer bit field indicating the state of the SHIFT (bit 0), CTRL (bit 1) and ALT (bit 2) keys at the time of the event. These bits correspond to the values 1, 2 and 4 respectively. Some, all or none of the bits can be set, indicating that some, all or none of the keys are down.
X, Y	Current coordinates of the mouse pointer.
<i>Return Value: none</i>	

MouseUp

Occurs when the user releases a mouse button over a SigCtl.

<i>Private Sub object_MouseUp(Button, Shift, X, Y)</i>	
<i>Parameters:</i>	
Button	Short integer that identifies the button that was released. The button argument is a bit field with bits corresponding to the left button (bit 0, value = 1), right button (bit 1, value = 2), and middle button (bit 2, value = 4). Only one of the bits is set, indicating the button that caused the event.
Shift	Short integer bit field indicating the state of the SHIFT (bit 0), CTRL (bit 1) and ALT (bit 2) keys at the time of the event. These bits correspond to the values 1, 2 and 4 respectively. Some, all or none of the bits can be set, indicating that some, all or none of the keys are down.
X, Y	Current coordinates of the mouse pointer.
<i>Return Value: none</i>	

FlSigCOM: SigCtlXHTML

Summary

Method
(as SigCtl)
CheckHostDocument
CheckHostDocument2
GetProperty
InsertMarkup
SetProperty
Property
(as SigCtl)
Printer
Event
(as SigCtl)

Extends SigCtl, the additions are indicated below.

Methods

CheckHostDocument

Recalculates the document hash to determine whether any changes have been made since signing. The methods call SigObj.CheckSignedData with the necessary Hash data.

<i>SignedData CheckHostDocument()</i>	
<i>Parameters:none</i>	
<i>Return Value:SignedData</i>	
DataGood	The document has not been changed since signing
DataNoHash	Signature has not been captured, or the hash was not calculated
DataBadType	Signature was captured using a different type of hash
DataBadHash	The document hash has changed since signing
DataError	An error occurred while calculating the hash

CheckHostDocument2

Recalculates the document hash and compares it with a value stored within the HTML document itself, eg:

```
<!-- DocHash:081726885B7A342A74 -->
```

The hash is inserted by a Wacom HTML converter component and is used to check that documents are not changed prior to signing.

<i>SignedData CheckHostDocument2()</i>	
<i>Parameters:none</i>	
<i>Return Value:SignedData</i>	
DataGood	The document has not been changed since signing
DataNoHash	Signature has not been captured, or the hash was not calculated
DataBadType	Signature was captured using a different type of hash
DataBadHash	The document hash has changed since signing
DataError	An error occurred while calculating the hash

InsertMarkup

Inserts text into an HTML element. Only plain text will be allowed. It will be assigned to the innerText property of the element. The HTML element must be empty (contain no text or child elements) and the control must not contain a captured signature.

Application Notes:

If a signature with associated markup is re-captured, markup will be removed from the document and lost. In order to re-capture with markup, first clear the signature and re-insert markup.

Attempting to insert markup into an element that does not exist or is not empty, or calling the method on a control that contains a signature will result in an exception being thrown in javascript.

Void InsertMarkup(Id, Text)

Parameters:

Id	String specifying the id of an HTML element in which to insert text.
Text	Text to insert.

Return Value: none

Properties

Property	Type	SigCtlXHTML Property Description
		See SigCtl Properties
Printer	PrintObj	Creates and returns a PrintObj object. Deprecated. (Read-only)

Properties supported by GetProperty/SetProperty

Name	Get/Set	Type	Description
			As SigCtl supported properties with the modification:
Document_HashType	Get	HashType	The HashType used for hash creation prior to signature capture. Only supported in SigCtlXHTML
DisablePropertyChangePostCapture			Not supported in SigCtlXHTML

Events

As SigCtl

FSigCOM: ImgCtlXHTML

Summary

Method
(none)
Property
ImageBase64

ImageHex
Licence
ResizeToImage

The `ImgCtlXHTML` object is used to allow graphics images to be embedded within HTML documents as data instead of being referenced by URLs to external files.

Images may be in a wide range of graphics formats, including PNG, GIF, TIFF, JPEG and BMP.

The image should be encoded using either Hex or Base64 strings. Base64 is preferred because it is more compact.

The image data is embedded in the document by specifying the Class ID of the control and providing the encoded image data, e.g.

```
<object classid="clsid:EFFD1818-3060-49a3-9C22-A06F57BBC167">
  <param name="ImageBase64" value="R01G0...hXBWwICAD"/>
</object>
```

In this example the majority of the image data has replaced with "...." for clarity.

Methods

None

Properties

Property	Type	ImgCtlXHTML Property Description
ImageBase64	String	Sets or gets the image text in Base64 format
ImageHex	String	Sets or gets the image text in hex format
Licence	Variant	Licence object or string
ResizeToImage	Boolean	Write Enable control resize on image change

FlSigCOM: Hash

Summary

Method
Add
Clear
Property
Hash
Type

The Hash object is used to calculate a one-way hash, the value of which is a fixed length 'string', from an arbitrary length data set. This is often referred to as calculating the 'message digest' of a 'message'. The data set will typically be the contents of a document or form, in which case the hash value will be a unique and reliable 'digital fingerprint' of the contents. To be secure, the algorithms used to calculate hash values must be such that it is not possible to forge a data set in order to yield a given hash value. Two of the most widely used and secure hash algorithms are MD5 (which produces a 128-bit hash value) and SHA (which produces a 160-bit hash value). The latter has been adopted as a [Federal Information Processing Standard](#).

Methods

Add

Adds a data item to the data set from which the hash value is calculated. When an item is added the Hash property is automatically updated with the new value of the one-way hash.

<i>void Add(Data)</i>	
<i>Parameters:</i>	
Data	Variant containing the data item to add. Most automation compatible data types may be added to the data set including the following types: boolean, Byte, Byte(), Currency, Date, Double, Integer, Long, Single, String. Attempting to add an unsupported data type results in a run-time error
<i>Return Value:none</i>	

Clear

Clears the current data set from the Hash object, thus resetting the object for the calculation of a new hash value.

<i>void Clear()</i>
<i>Parameters:none</i>
<i>Return Value:none</i>

Properties

Property	Type	Hash Property Description
Hash	String	Returns the current hash value as a "string". Note: this property is used internally by the component and is not intended for use by the client programmer. (Read-only)
Type	HashType (Enum)	Gets and sets type of hash algorithm used. See Enumerator Types for possible values. May be set only if a Hash object contains no data. An attempt to set Type after data has been added will result in an error. Default = 'HashNone', therefore the Type must be reset to a defined algorithm prior to using Hash object.

FlSigCOM: Key

Summary

Method
Set

Property
Type

The Key object is used for protecting the integrity of data ('message authentication'), such as protecting signature data from malicious or accidental tampering. Supported key types include a simple MD5-based key and a more secure Message Authentication Code (MAC) key derived from MD5. The latter requires an application defined key value - a 'password string', which (for security reasons) is best stored in a separate file or input by the user at run-time and not hard-coded in the application itself.

Methods

Set

Initializes the Key by setting its type and, optionally, initialization value.

<i>void Set(Type, Value)</i>	
<i>Parameters:</i>	
Type	KeyType Enum specifying the type of key.
Value	Value to use to initialize key. Must be omitted for key types KeyNone and KeyMD5. For KeyMD5MAC must be either a 16 byte array (BYTE(16) in VB) or a 16 character string. Note that the (Unicode) string is converted to multibyte, so only those characters that convert to a single byte should be used. (Optional)
<i>Return Value:none</i>	

Properties

Property	Type	Key Property Description
Type	KeyType (Enum)	Gets type of key. See Enumerator Types for possible values. Default = 'KeyNone'. (Read-only)

FlSigCOM: Enumeration Types

BorderStyle

Specifies the style of border drawn around a signature control

Name	Value	Description
BdrFlat	0	Solid border
BdrRaised	5	2-pixel wide, 3-d border giving the control the appearance of being raised above its background
BdrEtched	6	2-pixel wide, 3-d border with the appearance of a sunken edge.
BdrBump	9	2-pixel wide, 3-d border with the appearance of a raised edge.
BdrSunken	10	2-pixel wide, 3-d border giving the control the appearance of being sunken into its background

CaptData

Used to select what data is returned by the SigObj.AdditionalData property

Name	Value	Description
------	-------	-------------

CaptDigitizer	26	Identifying information for the digitizer pad
CaptDigitizerDriver	27	Identifying information for the digitizer pad driver software
CaptMachineOS	28	Identifying information for computer operating system
CaptNetworkCard	29	Identifying information for network card

CaptureData

Return values for legacy SigCtlXHTML Capture method. Obsolete

CaptureResult

Returned by the Capture methods of the SigCtrl & SigCtrlXHTML.

Name	Value	Description
CaptureOK	0	Signature captured successfully.
CaptureCancel	1	Signature not captured because user cancelled Signature Capture Window.
CapturePadError	100	No capture service available; typically, no functioning digitizing tablet or other device for capturing signatures found.
CaptureError	101	System error or some other unusual error condition.
CaptureIntegrityKeyInvalid	102	The integrity key parameter is invalid (obsolete)
CaptureNotLicensed	103	The component has not been licensed to perform capture. This may be because a suitable licence has not been set or because a condition of the licence has not been met, for example required hardware, such as a particular tablet type, was not found.
CaptureAbort	200	Signature not captured because the handler for the PreCapture event returned False. With SigCtlXHTML, this can indicate a failure to parse document contents

DisplayMode

Specifies the method used to draw a signature within a signature control and by the RenderRect method

Name	Value	Description
DspForceFit	0	Scale signature to fit the area
DspUseZoom	1	Use the Zoom property/parameter to scale the signature relative to actual size.
DspBestFit	2	Reduce size of signature to fit display area if it is too big, but otherwise show at true size.

HashType

Specifies the hashing algorithm used by the Hash object.

Name	Value	Description
HashNone	0	No hashing algorithm selected
HashMD5	1	MD5 hashing algorithm
HashSHA1	2	SHA-1 hashing algorithm
HashSHA224	3	SHA-224 hashing algorithm
HashSHA256	4	SHA-256 hashing algorithm
HashSHA384	5	SHA-384 hashing algorithm
HashSHA512	6	SHA-512 hashing algorithm

IntegrityStatus

Returned by the CheckIntegrity method of the SigObj interface.

Name	Value	Description
IntegrityOK	0	Data has not changed since signature capture
IntegrityFail	1	Data has changed since signature capture
IntegrityMissing	2	Signature integrity value not found
IntegrityWrongType	3	Signature was captured using a different integrity check version
IntegrityInsufficientData	4	MD5MAC key value is required in order to check integrity
IntegrityUncertain	5	Unable to check integrity. Obsolete
IntegrityUnsupported	6	Integrity type not supported on the current platform

KeyType

Specifies the type of a Key object

Name	Value	Description
KeyNone	0	No key type set
KeyMD5	1	MD5 key
KeyMD5MAC	2	MD5 MAC key
KeySHA1	3	SHA-1 key
KeySHA224	4	SHA-224 key
KeySHA256	5	SHA-256 key
KeySHA384	6	SHA-384 key
KeySHA512	7	SHA-512 key

RBFlags

Bitmask flags supplied to RenderBitmap to select the required options

Name	Value	Description
RenderOutputBase64	0x002000	Return bitmap as a base-64 encoded string
RenderOutputBinary	0x000800	Return bitmap as binary data
RenderOutputFilename	0x001000	Write bitmap to file
RenderOutputPicture	0x200000	Return bitmap as Picture object
RenderBackground Transparent	0x010000	Render image with a transparent background
RenderColor1BPP	0x020000	Render using 1 bit-per-pixel (monochrome)
RenderColor24BPP	0x040000	Render colour image with 24 bits-per-pixel
RenderColor32BPP	0x080000	Render colour image with 32 bits-per-pixel
RenderColorAntiAlias	0x100000	Render colour image with antialiasing
RenderEncodeData	0x400000	Encode signature data within image
RenderWatermark	0x800000	Encode watermark within image
RenderClipped	0x1000000	Crop signature image, omitting any parts which were outside of the original capture window.
RenderRelative	0x2000000	Renders the signature image relative to the origin of the original capture window

ReadEncodedBitmapResult

Result returned by the ReadEncodedBitmap method

Name	Value	Description
ReadEncodedBitmapOK	0	Signature data decoded OK
ReadEncodedBitmapFileNotFound	1	File not found
ReadEncodedBitmapNotImage	2	Bitmap is not a supported image type
ReadEncodedBitmapSigDataNotFound	3	Encoded signature data not found in image

ShowText

Used to specify if and how "act of signing" information (who signed and/or when) is displayed within the control.

Name	Value	Description
TxtDontShow	0	Don't display the information
TxtShowLeft	1	Display the information left aligned within the control
TxtShowCenter	2	Display the information centred within the control
TxtShowRight	4	Display the information right aligned within the control

SignedData

Returned by the CheckSignedData method of the SigObj interface.

Name	Value	Description
DataGood	0	Data has not changed signature capture
DataNoHash	1	No signature captured, or signature was captured without a hash
DataBadType	2	Signature was captured with a different type of hash
DataBadHash	3	Data has changed since signature capture
DataError	4	Error whilst checking signed data
DataUncertain	5	Unable to check status of data. Obsolete
DataSigMoved	6	The position of the SigCtlXHTML control within the HTML document has been changed

TimeZone

Used to select time zone for time returned by 'When' parameter

Name	Value	Description
TimeLocal	0	Local time when signature was captured
TimeGMT	1	Greenwich Mean Time
TimeUTC	1	Universal Coordinated Time (synonym for TimeGMT)

FlSigCapt

FlSigCapt: DynamicCapture

Summary

Method
Capture
GetProperty
SetProperty
Property
Licence

The DynamicCapture object provides the user interface for the capture of a signature.

Methods

Capture

Displays a Signature Capture Window in which a user may sign his or her name using a suitable digitizing tablet. The window contains three buttons by which the user can either clear the currently rendered signature (Clear button) or dismiss the window without capturing a signature (Cancel button) or dismiss the window and capture the currently rendered signature (OK button).

DynamicCaptureResult <i>Capture(SigCtl, Who, Why, What, Key)</i>	
<i>Parameters:</i>	
SigCtl	SigCtl or SigCtlXHTML object
Who	String specifying the signatory name. Optional if InputWho property of SigCtl is set
Why	String specifying the reason for signing. Optional if InputWhy property is set
What	Hash object Derived from the data set with which the captured signature is to be bound. Typically the data set will be the contents of the document or form signed by the user. SigCtlXHTML: The hash is generated automatically from the contents of the XHTML document and cannot be supplied here. (Optional)
Key	Key object supplied in order to detect malicious or accidental tampering with the signature object or the signature data it contains. SigCtlXHTML: Key type SHA-256 is used automatically but an alternative can be specified. (Optional)
<i>Return Value:DynamicCaptureResult</i>	
DynCaptOK	Signature captured successfully.
DynCaptCancel	Signature not captured because user cancelled Signature Capture Window.
DynCaptPadError	No capture service available; typically, no functioning digitizing tablet or other device for capturing signatures found.
DynCaptError	System error or some other unusual error condition.
DynCaptAbort	Signature not captured because the handler for the PreCapture event returned False. With SigCtlXHTML, this can indicate a failure to parse document contents
DynCaptNotLicensed	The component has not been licensed to perform capture. This may be because a suitable licence has not been set or because a condition of the licence has not been met, for example required hardware, such as a particular tablet type, was not found.

GetProperty

Returns the current value of a property.

```
void GetProperty(Name)
```

<i>Parameters:</i>	
Name	The name of the property to retrieve
<i>Return Value:Variant</i>	
Variant	Current value of the named property. Empty if no value has been set.

SetProperty

Sets the value of a named additional property, overrides a value set in the registry.

<i>void SetProperty(Name, Value)</i>	
<i>Parameters:</i>	
Name	String. Name of the property.
Value	Value to assign to property:
<i>Return Value:Boolean</i>	
True	Name exists and value is valid
False	Failed to set property

Properties

Property	Type	DynamicCapture Property Description
Licence	String	The license object or string

Properties supported by SetProperty/SetProperty

Name	Get/Set	Type	Description
Licence	Get/Set	String	String containing the license used for signature capture
Component_FileVersion	Get	String	Component version e.g. "4.5.2.210"
UILanguage	Get/Set	String	Selects the language used in the signature capture dialog from: For details (see SigObj properties)
Where	Get/Set	String	Specifies the location of signing. The string is displayed in the signature capture dialog and saved in the signature data
CaptureInkColor	Get/Set	String	Color of 'ink' used to render the signature image as a comma separated decimal RGB value. e.g. for Blue: setProperty("CaptureInkColor", "0,0,1") Individual color values are in the decimal range 0.0 to 1.0, factoring the RGB maximum of 255 e.g. 0.0 for 0, 0.5 for 127, 1.0 for 255
CaptureInkWidth	Get/Set	String	Decimal Width, in mm, of 'ink' used to render the signature image. Default = 0.7mm Note: <ul style="list-style-type: none"> • min = 0.1mm • max = 2.0 mm • scales with signature • varies with pressure (if pressure data is available)
CaptureExtentX	Get/Set	DWORD	Sets width of the capture area in pixels. If not set, the size of the capture area is determined by the size of the digitizer. If this value is set, CaptureExtentY must also be set.

CaptureExtentY	Get/Set	DWORD	Sets height of the capture area in pixels. If not set, the size of the capture area is determined by the size of the digitizer. If this value is set, CaptureExtentX must also be set.
ButtonLocation	Get/Set		Location of buttons in the capture window: 0 - Outside the capture area 1 - Inside the capture area 2 - Hidden (not shown)
LicenceString	Get	String	License name, extracted from license string (if set)
stuPort	Get/Set	String	Com port for serial interface e.g. "COM6"
stuBaudRate	Get/Set	DWORD	Serial linespeed e.g. 128000
			The following values are for use with the STU-540 device. For details see: https://developer-docs.wacom.com/display/DevDocs/STU-540++Signature+Screen+Upload
stuSignatureMode	Get/Set	DWORD	Select Signature mode: 0 - use default mode 1 - use signature mode 2 - use non-signature mode
stuSigModeScreenNum	Get/Set	DWORD	Select Signature screen number 1..3
stuSigModeWhy	Get/Set	String	Why string
stuSigModeWho	Get/Set	String	Who String
stuSigModeWhen	Get/Set	String	When String
stuSigModeOK	Get/Set	String	Text for OK button e.g. <code>dc.SetProperty("stuSigModeOK","OK (screen 1)");</code>
stuSigModeClear	Get/Set	String	Text for Clear button
stuSigModeCancel	Get/Set	String	Text for Cancel button
stuSigModeFontName	Get/Set	String	Font name e.g. "Courier New"
stuSigModeFontSize	Get/Set	DWORD	Font size e.g. 10
stuSigModeConfig	Set	String	Signature Mode configuration filename e.g. <code>dc.SetProperty("stuSigModeConfig", "c:\\config\\STU-screens.config,1");</code>

FLSigCapt: eSeal

Summary

Method
Capture
GetProperty
SetProperty
Property
CacheImage
HAlign
Height
HScale
Id
Licence
Name

Transparency
URL
VAlign
VScale
Width

The eSeal object provides the user interface for the insertion of an eSeal image into a signature area with the optional capture of a handwritten signature.

Methods

Capture

Optionally displays a Signature Capture Window (see DynamicCapture) and inserts an eSeal image in the signature area. If the eSeal has not already been cached, the URL is accessed to download the referenced file.

<i>eSealCaptureResult Capture(SigCtl, Mode, Who, Why, What, Key)</i>	
<i>Parameters:</i>	
SigCtl	SigCtl or SigCtlXHTML object
Mode	eSealCaptureMode: <ul style="list-style-type: none"> • esRequireSignature - Insert eSeal and capture handwritten signature • esNoSignature - Insert eSeal (without signature capture) • esSignatureOptional - Insert eSeal and capture signature if tablet is available
Who	String specifying the signatory name. Optional if InputWho property of SigCtl is set
Why	String specifying the signatory name. Optional if InputWho property of SigCtl is set
What	Hash object Derived from the data set with which the captured signature is to be bound. Typically the data set will be the contents of the document or form signed by the user. SigCtlXHTML: The hash is generated automatically from the contents of the XHTML document and cannot be supplied here. (Optional)
Key	Key object supplied in order to detect malicious or accidental tampering with the signature object or the signature data it contains. SigCtlXHTML: Key type SHA-256 is used automatically but an alternative can be specified. (Optional)
<i>Return Value:eSealCaptureResult</i>	
esCaptureOK	Signature captured successfully.
esCaptureCancel	Signature not captured because user cancelled Signature Capture Window.
esCapturePadError	No capture service available; typically, no functioning digitizing tablet or other device for capturing signatures found.
esCaptureError	System error or some other unusual error condition.
esCaptureAbort	Signature not captured because the handler for the PreCapture event returned False. With SigCtlXHTML, this can indicate a failure to parse document contents
esCaptureNoImage	Unable to load eSeal image from URL, or parameter error
esCaptureNotLicensed	The component has not been licensed to perform capture. This may be because a suitable licence has not been set or because a condition of the licence has not been met, for example required hardware, such as a particular tablet type, was not found.

GetProperty

Returns the current value of a property.

<i>void GetProperty(Name)</i>	
<i>Parameters:</i>	
Name	The name of the property to retrieve
<i>Return Value:Variant</i>	
Variant	Current value of the named property. Empty if no value has been set.

SetProperty

Sets the value of a named additional property, overrides a value set in the registry.

<i>void SetProperty(Name, Value)</i>	
<i>Parameters:</i>	
Name	String. Name of the property.
Value	Value to assign to property:
<i>Return Value:Boolean</i>	
True	Name exists and value is valid
False	Failed to set property

Properties

Property	Type	eSeal Property Description
CacheImage	boolean	False if URL is to be accessed at time of signing.
HAlign	eSealHAlign (enum)	Horizontal alignment of image within the signature area (defaults to 1=centre)
Height	Int	Height of image in HIMETRIC (0.01mm) units (Read-only)
HScale	Int	percentage scaling of image X dimension (defaults to 100)
id	String	GUID for internal use only (Read-only)
Licence	Variant	Licence object or string
Name	String	Name for internal use only (Read-only)
Transparency	Int	Percentage transparency, 0 = unchanged, 100=not visible (defaults to 0=unchanged)
URL	String	URL of image file. Maximum size is 32K. Use compressed format or reduce bit depth if necessary.
VAlign	eSealVAlign (enum)	Vertical alignment of image within the signature area (defaults to 1=centre)
VScale	Int	Percentage scaling of image Y dimension (defaults to 100)
Width	Int	Width of image in HIMETRIC (0.01mm) units (Read-only)

Properties supported by GetProperty/SetProperty

Name	Get/Set	Type	Description
Licence	Get/Set	String	String containing the license used for signature capture
Component_FileVersion	Get	String	Component version e.g. "4.5.2.210"
Where	Get/Set	String	Specifies the location of signing. The string is displayed in the signature capture dialog and saved in the signature data

FlSigCapt: Enumeration Types

DynamicCaptureResult

Returned by the Capture method of DynamicCapture.

Name	Value	Description
DynCaptOK	0	Signature captured successfully.
DynCaptCancel	1	Signature not captured because user cancelled Signature Capture Window.
DynCaptPadError	100	No capture service available; typically, no functioning digitizing tablet or other device for capturing signatures found.
DynCaptError	101	System error or some other unusual error condition.
DynCaptIntegrityKeyInvalid	102	The integrity key parameter is invalid (obsolete)
DynCaptNotLicensed	103	The component has not been licensed to perform capture. This may be because a suitable licence has not been set or because a condition of the licence has not been met, for example required hardware, such as a particular tablet type, was not found.
DynCaptAbort	200	Signature not captured because the handler for the PreCapture event returned False. With SigCtlXHTML, this can indicate a failure to parse document contents

eSealCaptureMode

Selects signature capture options when inserting an eSeal into a signature area.

Name	Value	Description
esRequireSignature	0	Insert eSeal and capture handwritten signature
esNoSignature	1	Insert eSeal (without signature capture)
esSignatureOptional	2	Insert eSeal and capture signature if tablet is available

eSealCaptureResult

Returned by the Capture method of eSeal.

Name	Value	Description
esCaptureOK	0	Signature captured successfully.
esCaptureCancel	1	Signature not captured because user cancelled Signature Capture Window.
esCapturePadError	100	No capture service available; typically, no functioning digitizing tablet or other device for capturing signatures found.
esCaptureError	101	System error or some other unusual error condition.
esCaptureNotLicensed	103	The component has not been licensed to perform capture. This may be because a suitable licence has not been set or because a condition of the licence has not been met, for example required hardware such as a particular tablet type, was not found.

esCaptureAbort	200	Signature not captured because the handler for the PreCapture event returned False. With SigCtlXHTML, this can indicate a failure to parse document contents
esCaptureNoImage	300	Unable to load eSeal image from URL, or parameter error

eSealHAlign

Used to select the horizontal alignment of an eSeal image in the signature area.

Name	Value	Description
esLeft	0	Left align.
esCentre	1	Centre.
esCenter	1	Centre
esRight	2	Right align

eSealVAlign

Used to select the vertical alignment of an eSeal image in the signature area.

Name	Value	Description
esTop	0	Top align.
esMiddle	1	Centre.
esBottom	2	Bottom

FlWizCom

FlWizCom: WizCtl

Summary

Method
AddObject
AddPrimitive
Display
FireClick
GetObjectState
GetProperty
PadConnect
PadDisconnect
Reset
SetEventHandler
SetProperty
Property
BackColor

BorderColor
BorderStyle
BorderVisible
BorderWidth
EnableWizardDisplay
Font
InkingPad
Licence
PadHeight
PadWidth
Zoom
Event
PadEvent

Methods

AddObject

Adds an item to the pad control list.

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
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. Value can either be absolute position in pixels, or one of the strings: <ul style="list-style-type: none"> X: "left", "right", "centre" Y: "top", "middle", "bottom"
Data	VARIANT. Value dependent on object type.
Options	VARIANT. Value dependent on object type
<i>Return Value:none</i>	

AddObject(ObjectText)

Displays a text string on the pad using the current font

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
Type	ObjectText
Id	The following values have special meanings when used with a signature object: "who"Text in Data will also be used as name of signatory. "why"Text in Data will also be used as reason for signing. "when"Reserved for future use (Can be null or an empty string)

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	Text to display.
Options	A value from the TextOptions Enumerator (Optional)
<i>Return Value:none</i>	

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>									
<i>Parameters:</i>									
Type	ObjectButton								
Id	The following values have special meanings when used with signature or input objects: <table border="1" data-bbox="293 764 1390 1094"> <tr> <td>"OK"</td> <td>Accepts current input. With a signature, stores the captured signature in the signature object and terminates input. Until a signature has been captured, the button is disabled by the Wizard Control. With an Input object, the button is disabled until the required minimum number of characters has been entered.</td> </tr> <tr> <td>"Clear"</td> <td>Clears current input allowing user to start again With a signature, clears any captured 'ink' from the display With an Input object, clears all entered input</td> </tr> <tr> <td>"Cancel"</td> <td>With a signature, clears any captured 'ink' and terminates input</td> </tr> <tr> <td>"Delete"</td> <td>With an input object, deletes the last character</td> </tr> </table>	"OK"	Accepts current input. With a signature, stores the captured signature in the signature object and terminates input. Until a signature has been captured, the button is disabled by the Wizard Control. 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 With a signature, clears any captured 'ink' from the display 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
"OK"	Accepts current input. With a signature, stores the captured signature in the signature object and terminates input. Until a signature has been captured, the button is disabled by the Wizard Control. 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 With a signature, clears any captured 'ink' from the display 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. Value can either be absolute position in pixels, or one of the strings: X: "left", "right", "centre" Y: "top", "middle", "bottom"								
Data	Text to display.								
Options	Either, an integer specifying button width in pixels or an ObjectOptions object. If the given width is less than the width of the text (in the current font), it is ignored. (Optional)								
<i>Return Value:none</i>									

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
Type	ObjectCheckbox
Id	The following values have special meanings when used with a signature object. 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. Value can either be absolute position in pixels, or one of the strings: X: "left", "right", "centre" Y: "top", "middle", "bottom"

Data	Text to display.
Options	A combination of values from the CheckboxOptions enum (Optional)
<i>Return Value:none</i>	

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
Type	ObjectCheckbox
Id	The following values have special meanings when used with a signature object. 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. Value can either be absolute position in pixels, or one of the strings: X: "left", "right", "centre" 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.
<i>Return Value:none</i>	

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
Type	ObjectSignature
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	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)
<i>Return Value:none</i>	

AddObject(ObjectInput)

Provides an input mechanism for PIN code entry.

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
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
<i>Return Value:none</i>	

AddObject(ObjectInputEcho)

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
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. (Optional)
<i>Return Value:none</i>	

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.

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
Type	ObjectHash
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	Hash object representing data to be bound to a captured signature.
Options	Not used, should be omitted
<i>Return Value:none</i>	

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.

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
Type	ObjectImage
Id	Supports the same reserved Ids as button objects. See AddObject(ObjectButton) above. To make the image clickable supply a named Id e.g. "myimage" To prevent the image being clickable supply a blank Id: ""
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: String: containing "://" is assumed to be the URL of an image file containing a character: '.' or '\' is assumed to be the name of an image file on the local file system otherwise assumed to be base64-encoded image data Picture: OLE picture object (IPicture or IPictDisp interface) Array of bytes: Binary image data as an array of bytes
Options	Not used, should be omitted
<i>Return Value:none</i>	

AddObject(ObjectDisplayAtShutdown)

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
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
<i>Return Value:none</i>	

AddObject(ObjectInking)

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

<i>void AddObject(Type, Id, X, Y, Data, Options)</i>	
<i>Parameters:</i>	
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
<i>Return Value:none</i>	

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

```
var 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.

<i>void AddPrimitive(Type, X1, Y2, X2, Y2, Data, Options)</i>	
<i>Parameters:</i>	
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. Value can be the absolute position in pixels or one of the strings: "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. Value can be the absolute position in pixels, one of the strings: "left", "right", "centre" (for X2) or "top", "middle", "bottom" (for Y2) or a string in the format "+V" or " V" (where V is an integer) for a value relative to X1 or Y1.
Data	Line width in pixels (Optional, default value 1)
Options	VARIANT. Combination of PrimitiveOptions values (Optional, default value PrimitiveLineSolid + PrimitiveOutline)
<i>Return Value:none</i>	

Display

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

<i>void Display()</i>
<i>Parameters:none</i>
<i>Return Value:none</i>

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.

<i>void FireClick(Id)</i>	
<i>Parameters:</i>	
Id	Id of pad control for which to simulate click
<i>Return Value:none</i>	

GetObjectState

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

<i>Variant GetObjectState(Id)</i>	
<i>Parameters:</i>	
Id	Identifier of object.
<i>Return Value: Variant</i>	

Variant	ObjectCheckbox: 1 = Checked; 0 = Unchecked ObjectInput: number of characters currently in input buffer Other Object types: Integer, value undefined Id not recognised: Empty variant
---------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

GetProperty

Returns the current value of a property.

<i>void GetProperty(Name)</i>	
<i>Parameters:</i>	
Name	The name of the property to retrieve
<i>Return Value:Variant</i>	
Variant	Current value of the named property. Empty if no value has been set.

PadConnect

Connects to the signature tablet / pad.

<i>Boolean PadConnect(Id)</i>	
<i>Parameters:none</i>	
<i>Return Value: Boolean</i>	
True	Success
False	Failed to connect

PadDisconnect

Disconnects the signature tablet / pad.

<i>void PadDisconnect()</i>	
<i>Parameters:none</i>	
<i>Return Value:none</i>	

Reset

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

<i>void Reset()</i>
<i>Parameters:none</i>
<i>Return Value:none</i>

SetEventHandler

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

<i>void SetEventHandler(EventHandler)</i>
<i>Parameters:</i>

EventHandler	Dispatch interface with a default method. See Notes below for method parameters
<i>Return Value:none</i>	

The EventHandler function should have the prototype:

<i>void EventHandler(IDispatch * WizardControl, VARIANT Id, VARIANT Type)</i>	
<i>Parameters:</i>	
WizardControl	WizardControl object generating event
Id	Id of the WizardControl object which is the source of the event
Type	Type of event generated
<i>Return Value:none</i>	

SetProperty

Sets the value of a named additional property, overrides a value set in the registry.

<i>void SetProperty(Name, Value)</i>	
<i>Parameters:</i>	
Name	String. Name of the property
Value	Value to assign to property
<i>Return Value:Boolean</i>	
True	Name exists and value is valid
False	Failed to set property

Properties

Property	Type	WizCtl Property Description
BackColor	OLE_COLOR	Background colour
BorderColor	OLE_COLOR	Colour of 'Flat' border
BorderStyle	long	Type of border displayed if BorderVisible is True. See BorderStyle enumeration type.
BorderVisible	Boolean	True to draw border round wizard display; False to draw without a border
BorderWidth	long	Width in pixels of 'Flat' border
EnableWizardDisplay	Boolean	Enables or disables mouse interaction with objects (buttons etc) on the wizard display Note: Mouse interaction is not currently implemented
Font	IFontDisp	Font used to display text of all subsequently added objects
InkingPad	Boolean	True if the pad has a display(Read-only)
Licence	Variant	Licence object or string
PadHeight	short	Height of the pad display in pixels. Read-only if inking pad
PadWidth	short	Width of the pad display in pixels. Read-only if inking pad
Zoom	float	Sets the scaling of the wizard display relative to the physical pad display as a percentage. (Default: 100)

Properties supported by SetProperty/SetProperty

Name	Get/Set	Type	Description
Licence	Get/Set	String	String containing the license used for signature capture
Component_FileVersion	Get	String	Component version e.g. "4.5.2.210"
DynamicSignatureCapture_RetryCount	Get/Set	String	The number of times the Clear button can be used during signature capture
ForceMonochrome	Get/Set	Boolean	When true the Wizard sends text displays as monochrome images. This reduces the time to update the display on colour STU tablets.
ObjectInking_Bitmap	Get	String	Snapshot of captured 'ink', in PNG image format, as a base-64 encoded string. (See AddObject(ObjectInking))
InputValue	Set		Sets the value for an ObjectInput
ObjectOptions.*	Get	Boolean	True if specified option is supported. Possible value are: ObjectOptions.Button.Width ObjectOptions.Button.Height ObjectOptions.Button.Invert ObjectOptions.Echo.Char ObjectOptions.Echo.Spacing ObjectOptions.Echo.Underline ObjectOptions.Echo.Length
CaptureInkWidth	Get/Set	String	Decimal Width, in mm, of 'ink' used to render the signature image. Default = 0.7mm Note: <ul style="list-style-type: none"> • min = 0.1mm • max = 2.0 mm • scales with signature • varies with pressure (if pressure data is available)
CaptureInkColor	Get/Set	String	Color of 'ink' used to render the signature image as a comma separated decimal RGB value. e.g. for Blue: setProperty("CaptureInkColor", "0,0,1") Individual color values are in the decimal range 0.0 to 1.0, factoring the RGB maximum of 255 e.g. 0.0 for 0, 0.5 for 127, 1.0 for 255
ObjectForegroundColor	Get/Set	String	Foreground color to use for objects added subsequently as a comma separated decimal RGB value.
ObjectBackgroundColor	Get/Set	String	Background color to use for objects added subsequently as a comma separated decimal RGB value.

FlWizCom: InputObj

Summary

Method
Clear
SetEncryption
Property
Data
EncryptionType
MaxLength
MinLength
Text

The InputObj object is used to handle the input data when the wizard control is configured for PIN code entry. The InputObj handles all the interaction between the user and the pad, greatly simplifying the coding effort needed. It is used to set the PIN configuration, including its minimum and maximum lengths and the type of encryption required.

Methods

Clear

Resets the input object ready to restart PIN capture.

<i>void Clear()</i>
<i>Parameters:none</i>
<i>Return Value:none</i>

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

<i>Void SetEncryption(Type, Key)</i>	
<i>Parameters:</i>	
Type	EncryptAlg enum value specifying type of encryption to be used.
Key	VARIANT. The encryption key to be used
<i>Return Value:none</i>	

Properties

Property	Type	InputObj Property Description
Data	VARIANT	Gets the data entered by the user in the form of a byte array.
EncryptionType	LONG	Returns the type of encryption being used. Note: The encryption type is set using the SetEncryption method.
MaxLength	short	The maximum number of digits in the PIN
MinLength	short	The minimum number of digits in the PIN
Text	BSTR	Gets the data entered by the user. If no encryption is used the string will contain the digits as entered, otherwise the result will be a Base64 encoded encrypted string.

FlWizCom: ObjectOptions

Summary

Method

GetProperty
SetProperty
Property
none

The ObjectOptions object is used to pass multiple options to the WizCtl.AddObject method.

Methods

GetProperty

Returns the current value of a property.

<i>void GetProperty(Name)</i>	
<i>Parameters:</i>	
Name	The name of the property to retrieve
<i>Return Value:Variant</i>	
Variant	Current value of the named property. Empty if no value has been set.

SetProperty

Sets the value of a property.

<i>void SetProperty(Name, Value)</i>	
<i>Parameters:</i>	
Name	The name of the property to set
Value	Value to assign to property.
<i>Return Value:Boolean</i>	
True	

Properties

Values of any type may be assigned to any named property, however only certain properties are recognised for each object type. Recognised properties are expected to have values of a particular type (or of a type that can be converted to the expected type).

Properties supported by ObjectButton

Property	Expected Type	Property Description
Align	Integer	A combination of values from the ButtonOptions enumeration specifying the alignment of text within the button rectangle
Greyed	Boolean	True to create a disabled button (displayed greyed out)
Height	Integer	Height to make button in pixels
Invert	Boolean	True to display the button inverted (white text in a black rectangle)
Width	Integer	Width to make button in pixels

Properties supported by ObjectInputEcho

Property	Expected Type	Property Description
CharSet	String	Single-character string specifying character to use for echoing input e.g. "*" if input is for a password or PIN
Spacing	Integer	One of the spacing values from the InputEchoOptions enumeration.
Underline	Boolean	False to turn off display of underlines to indicate input echo character positions.

Properties supported by ObjectRadioButton

Property	Expected Type	Property Description
Group	String	Name of group to which a radio button belongs. Required
Checked	Boolean	True if the radio button is initially selected. If multiple radio buttons in a group are created with Checked=true, the last one added will be the one selected.

FIWizCom: Enumeration Types

BorderStyle

Used to set or get the style of border drawn around the control

Name	Value	Description
BdrFlat	0	Solid border
BdrRaised	5	2-pixel wide, 3-d border giving the control the appearance of being raised above its background
BdrEtched	6	2-pixel wide, 3-d border with the appearance of a sunken edge.
BdrBump	9	2-pixel wide, 3-d border with the appearance of a raised edge.
BdrSunken	10	2-pixel wide, 3-d border giving the control the appearance of being sunken into its background

ButtonOptions

WizCtl: used to specify alignment of text without button rectangle (via ObjectOptions object)

Name	Value	Description
BtnAlignCentre	0x00	Centre horizontally
BtnAlignMiddle	0x00	Centre vertically
BtnAlignLeft	0x01	Left align
BtnAlignRight	0x02	Right align
BtnAlignTop	0x04	Top align
BtnAlignBottom	0x08	Bottom align

CheckboxOptions

WizCtl: used to specify initial checkbox state when adding a checkbox object and to select the display type

Name	Value	Description
CheckboxUnchecked	0x00	Initial state unchecked

CheckboxChecked	0x01	Initial state checked
CheckboxDisplayTick	0x02	Indicate checked with a tick symbol
CheckboxDisplayCross	0x04	Indicate checked with a cross

EncryptAlg

Specifies the type of encryption to be used

Name	Value	Description
EncryptNone	0	None
EncryptTripleDES	1	Triple DES

EventType

Type of Event for callback function

Name	Value	Description
EvTextClicked	0	Text Clicked
EvButtonClicked	1	Button Clicked
EvCheckboxChecked	2	Checkbox Checked
EvCheckboxUnchecked	3	Checkbox Unchecked
EvInputMinReached	4	Input Min Reached
EvInputMaxReached	5	Input Max Reached
EvInputExceeded	6	Input Exceeded

InputEchoOptions

Used to specify the way an InputEcho field echoes user input

Name	Value	Description
EchoNoSpacing	0x00	No space between characters
EchoHalfSpacing	0x01	Half space
EchoSingleSpacing	0x02	Single space
EchoDoubleSpacing	0x04	Double space
EchoUnderline	0x08	Underline echoed characters

MarkupStatus

HTML document Markup status

Name	Value	Description
mkupSuccess	0	Markup added successfully
mkupDocError	1	Unable to access HTML document
mkupEltNotFound	2	Failed to find named element in HTML document

mkupEltNotEmpty	3	Named element in HTML document is not empty
mkupEltEmpty	4	Named element in HTML document is empty
mkupError	5	Error inserting markup into HTML document

ObjectType

Used to specify object type required in calls to AddObject method

Name	Value	Description
ObjectText	0	Display text
ObjectButton	1	Button
ObjectCheckbox	2	Checkbox
ObjectSignature	3	Signature
ObjectInput	4	Input
ObjectInputEcho	5	Input Echo
ObjectHash	6	Hash
ObjectImage	7	Display image
ObjectDisplayAtShutdown	8	Do not clear pad display on disconnect
ObjectInking	9	Enable ink capture as image
ObjectRadioButton	10	Radio Button

PrimitiveOptions

Used to specify options for graphics primitives in calls to AddPrimitive method

Name	Value	Description
PrimitiveLineSolid	0x01	Solid
PrimitiveLineDashed	0x02	Dashed
PrimitiveOutline	0x04	Outline
PrimitiveFill	0x08	Fill shape
PrimitiveFillXOR	0x10	Invert area of shape

PrimitiveType

Used to specify graphics primitive type required in calls to AddPrimitive method

Name	Value	Description
PrimitiveLine	0	Line
PrimitiveRectangle	1	Rectangle
PrimitiveEllipse	2	Ellipse

TextOptions

Used to specify text alignment

Name	Value	Description
TextAlignLeft	0	Align left
TextAlignRight	1	Align right
TextAlignCentre	2	Centre the text
TextAlignJustify	3	Insert spaces to align left and right

Configuration (Registry Settings)

Signature Capture Dialog Settings

Various aspects of signature capture, including the physical layout of the capture window (see figure below), can be configured using registry settings under the key:

```
HKEY_LOCAL_MACHINE\Software\Florentis\sd
```

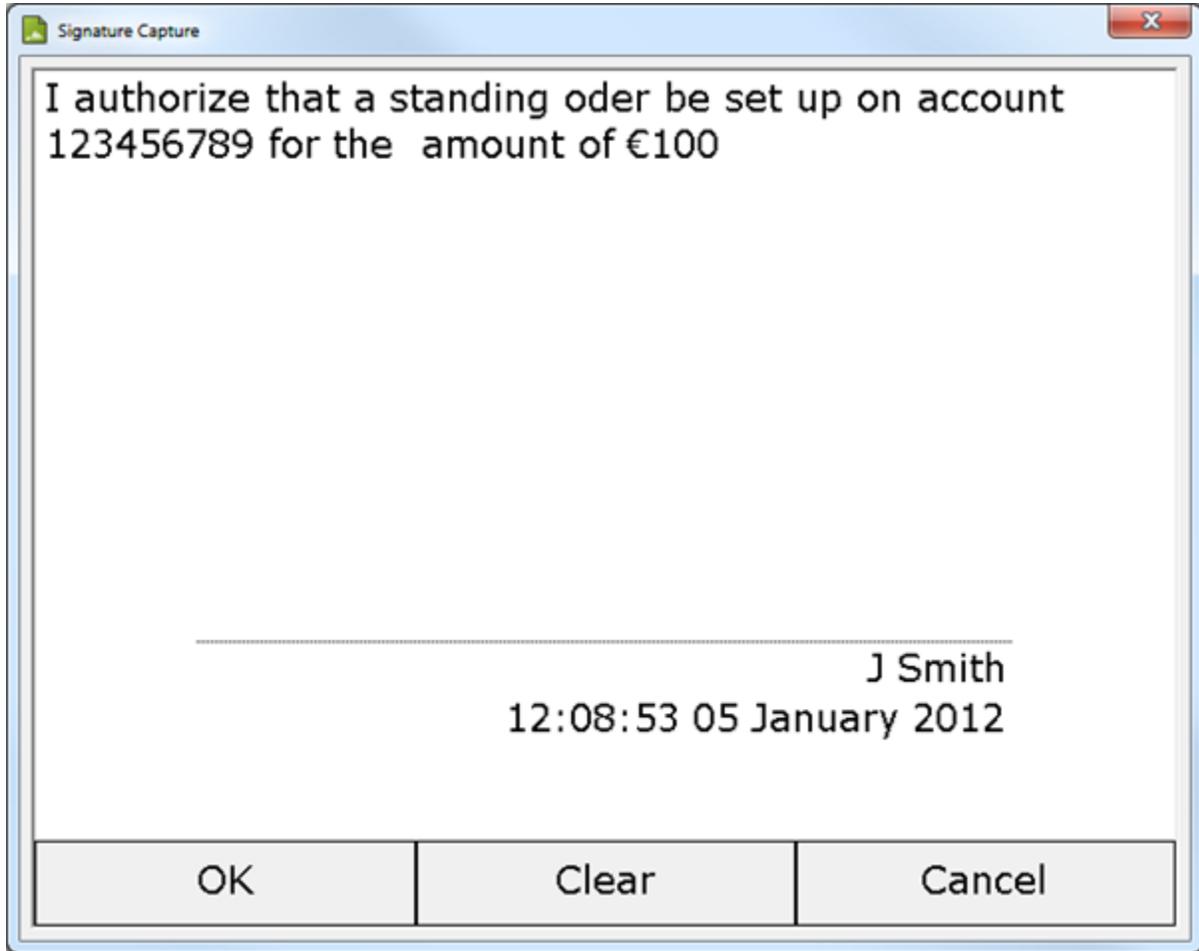
On 64-bit Windows with a 32-bit Signature Library installation:

```
HKEY_LOCAL_MACHINE\Software\WOW6432Node\Florentis\sd
```

```
HKEY_LOCAL_MACHINE\Software\Florentis\sd  
On 64-bit Windows with a 32-bit signature library installation:  
HKEY_LOCAL_MACHINE\Software\WOW6432Node\Florentis\sd
```

The signature capture window contains the elements:

- Why - I authorize...
- Who - J Smith
- When - 12:08:53 ...
- Buttons - OK/Clear/Cancel
- Capture area - pen area restrained to the dialog

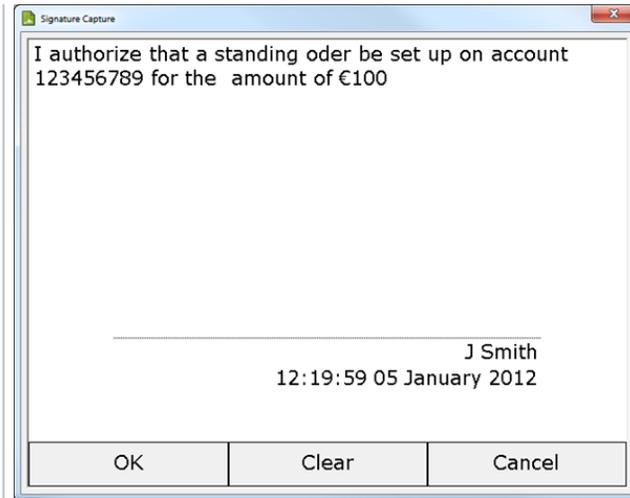


Capture Window element placement

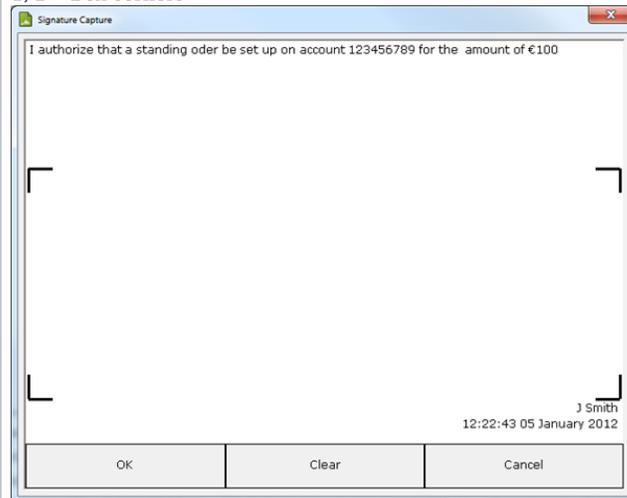
The following table lists the values which can be used together with their type and effect(s).

Signature Capture Style		
Value	Type	Description
BtnsInside	DWORD	Non-zero (default) displays and responds to the OK/Clear/Cancel buttons on the pad display. A zero value displays the buttons outside the Windows capture dialog, and removes them from the pad display.
CaptureBackground	DWORD	A value to control the background texture of the capture window and for colour tablets. The default value is 0. 0 - disable background. 1 - use paper white texture. (all other values reserved for future use)
CaptureExtentX	DWORD	Sets width of the capture area in pixels. If not set, the size of the capture area is determined by the size of the digitizer. If this value is set, CaptureExtentY must also be set.
CaptureExtentY	DWORD	Sets height of the capture area in pixels. If not set, the size of the capture area is determined by the size of the digitizer. If this value is set, CaptureExtentX must also be set.
CaptureInkColor	String	A string representation of three comma separated floating numbers between 0.0 and 1.0 inclusive for the RGB values of the ink on the screen and tablet (where supported). The default value is "0, 0, 0.545098".

CaptureInkWidth	String	A string representation of a floating point number, stating the width of the ink on screen, in mm. The default value is "0.8".
CaptureService	String	Limits the capture service(s) used: <ul style="list-style-type: none"> • hid • signpad • tabletpc • wintab One or more values can be defined, e.g. CaptureService=signpad;wintab. For tablet PCs this should be set to "tabletpc;wintab".
EnableSimulation	DWORD	Enable/disable simulation of signature capture when a digitizer is not installed. For example, to run a signature capture demonstration with no digitizer installed, simply ensure this key has been defined as non zero in the registry: sd: EnableSimulation=1 0 (or not present) = disable non-zero = enable
epInkWidth	DWORD	An enumeration value directly sent to the ePad-Ink to control ink width. The default value is 1. The following registry keys have been removed: <ul style="list-style-type: none"> • WhoInside • WhyInside • WhenInside • WhereInside • InkingDigitizer
InkingDigitizer	DWORD	(v3.x no longer supported) Value: 0 - Assume non-inking characteristics. Value: 1 - Assume inking characteristics. Explicitly tunes the capture service for an inking digitizer. Dependent upon the underlying software drivers, this typically results in changes to the user interface, mouse handling and cursor display.
InsideFontFaceName	String	Name of font to use for text within the capture area.
InsideFontCharSet	DWORD	Character set to use for text within the capture area. (Appropriate values may be found in the Win32 API documentation.)
InsideFontHeight	DWORD	Point size of font to use for text within the capture area.
OutsideFontFaceName	String	Name of font to use for text outside of the capture area.
OutsideFontCharSet	DWORD	Character set to use for text outside of the capture area.
OutsideFontHeight	DWORD	Point size of font to use for text outside of the capture area.
ShowWaitDlg	DWORD	Optionally display 'initialising signature capture' to start signature capture: Value: 0 - do not display Value: 1 - display when starting signature capture (default) The option is more noticeable with a low speed devices such as a serial STU
SignatureGuideType	DWORD	0 = horizontal line:

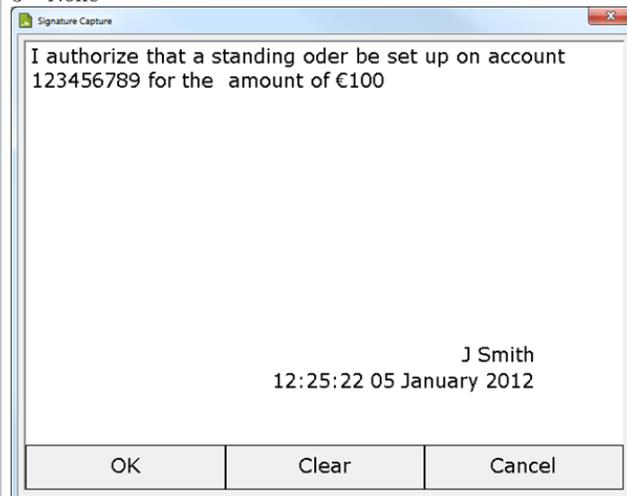


1, 2 = Box corners

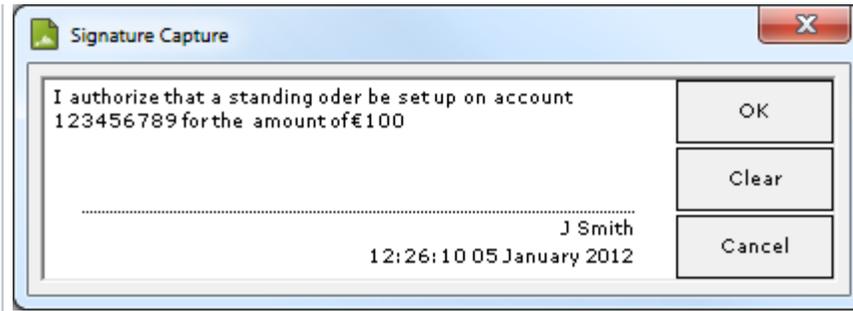


For these options, either BtnsInside should be 0, or InsideFontHeight should be reduced, otherwise buttons may overlap name and time; reason for signing should be limited to two lines otherwise top of box may be overwritten. Type 1 is a fixed size box; type 2 makes a (limited) adjustment to the top of the box for longer reason for signing strings.

3 = None



4 = Buttons on right



stuInkWidth	DWORD	An enumeration value directly sent the STU tablet to control ink width. Valid values are typically 0, 1, or 2. See the STU documentation for full details. The default value as per the tablet firmware.
SyncCursor	DWORD	Non-zero to force the system cursor to mirror pen movements on screen during signature capture.
WhenInside	DWORD	(v3.x no longer supported) Non-zero to force date and time to appear within the capture area; zero to force it to appear outside (below) the capture area.
WhereInside	DWORD	(v3.x no longer supported) Non-zero to force location to appear within the capture area; zero to force it to appear outside (below) the capture area.
WhoInside	DWORD	(v3.x no longer supported) Non-zero to force signatory name to appear within the capture area; zero to force it to appear outside (below) the capture area.
WhyInside	DWORD	(v3.x no longer supported) Non-zero to force reason for signing to appear within the capture area; zero to force it to appear outside (above) the capture area.
WizardShowWait	DWORD	Applies to STU-430 and STU-530 only. If the setting is missing or non-zero, an hourglass icon will be displayed when the Wizard control updates the STU display
wtLibrary	String	Full path and name of wintab dll to be used.
wtButtonMask	DWORD	Default: 0xffffffff. Value: bit-mask - Wintab Specification 1.1, section 7.4.1, pkButtons (absolute mode) Wintab driver specific. A digitizer stylus may implement multiple buttons, of which the 'tip' is one. In general it is not possible to determine which button is the tip, therefore the capture service assumes that all button activity is tip-related. However, it has been found that some buttons mounted on the side of the stylus can be pressed in error whilst signing, resulting in the recording of erroneous data when a pen in close proximity is mistakenly considered to be down. Setting the button mask to the explicit tip button bit resolves this problem.
wtMapDigitizer	DWORD	Default: 0 Value: 0 - do not map digitizer Value: 1 - map digitizer Wintab driver specific. The capture window does not change the tablet mapping as a default. This means that on a Bamboo/Intuos tablet the signature capture window uses a partial area of the tablet surface. The behaviour can be changed to use the whole tablet surface for signature capture by selecting the wtMapDisgitizer option: If the capture window is moved once a signature has been collected, no further ink is permitted. However you are free to press any of the buttons; pressing the "Clear" button will allow ink to be collected again.
wtPKTDATA	DWORD	Value: bit-mask - Wintab Specification 1.1, section 7.1, WTPKT Wintab driver specific. Some Wintab drivers do not correctly identify the data that they collect. This bit mask adds (binary OR) to the data reported by the digitizer. Note: altering this value may affect the forensic data that is stored.

Settings relating to the SigCom DLL can be configured using registry settings under the key:

HKEY_LOCAL_MACHINE\Software\Florentis\SigCom

On 64-bit Windows with a 32-bit Signature Library installation:

HKEY_LOCAL_MACHINE\Software\WOW6432Node\Florentis\SigCom

The following table lists the values which can be used together with their type and effect(s).

SigCOM Settings		
Value	Type	Description
get_SigText	DWORD	Selects the format of the data returned when reading SigObj.SigText 16 = base 16 (hexadecimal) 64 = base 64 (default) any other value will generate an error when the property is read. The setting is not used when <i>writing</i> the signature data - the format is detected automatically
language	String	User-interface language preference(s) as comma or semi-colon separated list of language IDs. e.g.: [HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Florentis\SigCapt] "language" = "FR; " If multiple IDs are defined the first language found will be used, For example for Belgian French or fallback to standard French: "language" = "FR-BE;FR"
ResourceDLL	String	Full pathname to a language resource DLL Obsolete/Deprecated

SigCapt DLL Settings

Settings relating to the SigCapt DLL can be configured using registry settings under the key:

HKEY_LOCAL_MACHINE\Software\Florentis\SigCapt

On 64-bit Windows with a 32-bit Signature Library installation:

HKEY_LOCAL_MACHINE\Software\WOW6432Node\Florentis\SigCapt

The following table lists the values which can be used together with their type and effect(s).

SigCapt Settings		
Value	Type	Description
language	String	User-interface language preference(s) as comma or semi-colon separated list of language ids.
ResourceDLL	String	Full pathname to the language resource DLL Obsolete/Deprecated

WizCOM DLL Settings

Settings relating to the SigCapt DLL can be configured using registry settings under the key:

HKEY_LOCAL_MACHINE\Software\Florentis\WizCOM

On 64-bit Windows with a 32-bit Signature Library installation:

HKEY_LOCAL_MACHINE\Software\WOW6432Node\Florentis\WizCOM

The following table lists the values which can be used together with their type and effect(s).

WizCOM Settings		
Value	Type	Description
language	String	User-interface language preference(s) as comma or semi-colon separated list of language ids.
ResourceDLL	String	Full pathname to the language resource DLL Obsolete/Deprecated

Signature Data Encryption

Overview

From version 4.x industry standard encryption can be applied to signature data while maintaining existing SigObj processing.

Signature data can be obtained from, or stored in a SigObj by:

- Reading SigText property
- Reading SigData property
- Steganographic encoding in signature image (RenderBitmap)
- COM object persistence (ie writing to an IStream)

Correspondingly, a SigObj can be initialized with signature data by:

- Assignment to SigText property
- Assignment to SigData property
- Reading from encoded bitmap (ReadEncodedBitmap)
- COM object persistence (ie read from an IStream)
- Signature capture

The added layer of encryption prevents unauthorized use while allowing authorized access to the full range of operations of a signature object (SigObj).

Both symmetric and asymmetric encryption methods are supported:

- Symmetric encryption uses the AES algorithm with a 256-bit key in CBC mode.
- Asymmetric encryption uses OpenSSL "envelope" functions. Signature data is encrypted with a symmetric "session key" (AES, 256-bit, CBC mode as above).
The session key is itself is then encrypted using the public key.

Unencrypted (binary) FSS data begins with the 2 bytes 'F' & 'S'. Encrypted data begins with an 8-byte header - "wgssAES_" for AES (symmetric) encryption; "wgssRSA_" for RSA (asymmetric) encryption.

When a SigObj is initialized with encrypted signature data, until the data is decrypted:

- SigCtl will display a 'locked' icon to indicate the signature is encrypted
- SigText & SigData will return the encrypted value (and continue to do so after decryption)
- Clear() method deletes the encrypted data, returning the object to its uninitialized state
- All other properties & methods fail with E_ACCESSDENIED

When the decryption key is set, data will be decrypted and SigObj will operate as normal (except for SigText/SigData returning encrypted values as noted above).

Setting an Encryption Key

SetProperty is used to enable encryption and decryption by setting symmetric or asymmetric key(s). It can be "turned off" by calling SetProperty with a null/empty key.

For symmetric encryption:

```
SetProperty("EncryptionPassword", password);
```

For asymmetric encryption:

```
SetProperty("EncryptionPublicKey", key);  
SetProperty("EncryptionPublicKeyFile", filename);  
SetProperty("EncryptionPrivateKey", key);  
SetProperty("EncryptionPrivateKeyFile", filename);
```

"key" (string) or file contents must be in PEM format.

If the key is password-protected, the password can be passed to SetProperty by appending it to the key or filename, separated by a comma. For example:

```
SetProperty("EncryptionPrivateKeyFile", "private_key.pem,mypassword")
```

In practice the password will most likely be retrieved by the application at runtime rather than being hardcoded and visible.

Key Generation using OpenSSL

Private key:

```
openssl.exe genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
```

To make the key password protected, add: `-aes128 -pass pass:password`

(optionally, use `-aes256` instead of `-aes128`)

Public key:

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

Encryption Status

The following values can be retrieved using `GetProperty`:

<code>GetProperty("IsEncrypted")</code>	Returns TRUE if the SigObj contains encrypted signature data which has not (yet) been decrypted.
<code>GetProperty("CanEncrypt")</code>	Returns TRUE if a password or public key has been set. If the SigObj already contains sig data, or if it is subsequently set (via SigData, SigText, Capture etc), it will be encrypted.
<code>GetProperty("CanDecrypt")</code>	Returns TRUE if a password or private key has been set. Sig data encrypted (with the appropriate password or public key) can be set and will be decrypted.

Signature ISO Format

Overview

Signatures captured using the Signature Library are saved in the proprietary Wacom Forensic Signature Store (FSS) format. The format is used consistently to provide compatibility between different platforms and applications, for example mobile signature capture with Windows SignatureScope analysis.

To provide interoperability with third-party software an ISO format signature can be used. A number of ISO standards are defined but the Signature Library implements the following:

- ISO 19785 XML format
- binary data format specified in ISO19794-7

An example of an ISO format signature is as follows where the signature pen data is contained within the `<BDB>` tags:

Sample signature in ISO format

```
?xml
version="1.0" encoding="UTF-8"?>
<BIR xmlns="http://standards.iso.org/iso-iec/19785/-3/ed-2/">
<!--
{iso registration-authority cbeff(19785) biometric-organization(0)
jtc1-sc37(257) patron-format(1) xml-full(7)} -->
<!--
<OBJECT_IDENTIFIER>1.1.19785.0.257.1.7</OBJECT_IDENTIFIER> -->
<Version><Major>2</Major><Minor>0</Minor></Version>
<CBEFFVersion><Major>2</Major><Minor>0</Minor></CBEFFVersion>
<BIRInfo>
<Creator>WacomGSS Signature SDK - 67</Creator>
<Integrity>>false</Integrity>
<CreationDate>2018-06-19T14:15:17Z</CreationDate>
</BIRInfo>
<BDBInfo>
<CreationDate>2018-06-19T14:15:09Z</CreationDate>
<Type>SignatureSign</Type>
</BDBInfo>
<BDB>U0RJACaxMADBYOC0gIAAp7
/gtICAAJ3PgOnEYAAAAf8AAAAA0qJ5A6AAAAgGjQ5A6ADIApWgJQ5A6AGQA5gGjQ5A6AJYBDQgJQ5A6AMgBJwGjQ5A6APoBPQgJQ5A6ASwBUQ
GjN5A5AV4BZQgJN5A5AZABdgGjN5A5AcIBgwgJN5A5AfQBjwGjN5A5AiYBmQgJN5A5AlgBogGjN5A5AooBqgGjN5A5ArwBsgGjN5A5Au4BuQGjN5
A5AyABwAGjN5A5A1IBxgGjN5A5A4QBzAGjN5A5A7YB0gGjN5A5A+gB1wGjN5A5BB0B3gGjN5A5BEwB5AGjN5A5BH4B6QGjLZBCBLAB7gGjLZBCBO
IB8gGjIJBEBRQB9gGjFpBGBUYB+AGjCZBHBXgB+WGi+JBIBaoB/QGj5JBIBdwB/gGjypBJBg4B/gGjirZBJbKAB/WGi5BjBnIB/WGiZpBJbqQB
/WGiOJBIBtYB/WGiBZBHBwGj/WGhzJBHBzOB/WGhjpbHB2wB/WGhTzBHB54B/WGhB5BHB9AB/WGgvZBHCAIB/WGgcJBICDQB/WGgJJBjCGYB
/WGf2JBjCjGjB/WGfjJBjCMoB/WGfQjBjJCPwB/WGe9ZBKCS4B/WGeqZBLCWAB/WGeXpBMCZIB/WGeFZBMCCQB/WGdypBNCFYB/WGdfpBOCigB
/WGdMZBOClob/WGc45BOCOWB/WGclJB0Cr4B/WGcQ5BNCvAB/WGb8JBNCyEB/WGbmpBMC1QB/WGbrJBLC4YB/WGa7pBKC7gB/WGanZBKC+oB
/WGaW5BLDBwB/WGaIZBMDE4B/WGZ85BMDIAB/WGZ55BNDLIB/WGZuJBPDQOB/WGZqpBQDRYB/WGZqpBQDUgB/WGZqpBQDXkB/WGZqpBQDawB
/WGZqpBQDd4B/WGZqpBQDhAB/WGZqpBQDkIB/WGZqpBQDnQB/WGZqpBQDqYB/WGZqpBQDtGB/WGZqpBQDwoB/WGZqpBQDzWB/WGZqpBQD24B
/WGZqpBQD6AB/WGZopBGD9IB/WGZopBGEAQB/WGZopBGEDYB/WGZpJA0EGgB/WGZpZaIEJoB/WGZppAHEMwB/WGZp4/1EP4B/WGZp4+5ETAB
/WGZqI+FEWIB/WGZqY9IEZQB/WGZqY8DECYB/WGZq466EfgB/WGZq45vEioB/WGZqo4jElwB/WGZq43VEo4B/WGZq42GESAB/WGZq402EvIB
/WGZq4znEYQB/WGZq4yYELYB/WGZqoxKE4gB/WGZqov6E7oB/WGZqIuvE+wB/WGZo4tvFB4B/WGZn4s4FFAB/WGZnosLFIIB/WGZn4roFLQB
/WGZn4rPFOYB/WGZoIq/FRgB/WGZoIq/FUoB/WGZoIq/FXwB/WGZoIq/Fa4B/WGZroq3FeAB/WGZroq3FhEB/WGZroq3FkMB/WGZroq3FnYB
/WGZroq3FqgB/WGZtYqtFtoB/WGZtYqtFwwB/WGZtYqtFz4B/WGZv4qpf3AB/WGZv4qpf6IB/WGZzoqoF9QB/WGZ2IqpGAYB/WGZ44qpGDgB
/WGZ8YqpGGoB/WGaAIqrGjWb/WGaEoqrGM4B/WGaJoqsGQAB/WGaPYqsGTIB/WGaWIqtGWQB/WGaeoqtGZYB/WGaoYqtGcgB/WGa0IqtGfoB
/WGbBoqtGiWb/WGbRYquG14B/WGbi4quGpAB/WGb3IquGsIB/WGcNIquGvMB/WGclYquGyUB/WGdAIquG1gB/WGddIqvG4oB/WGd8IqvG7wB
/WGedIqvG+4B/WGe/ogvHCAB/WGfjYqwhfIB/WGgH4qyHIQB/WGgs4q2HLYB/WGhQIq6HOGb/WGhu4q8HROB/WGiJIq+HUwB/WGieorAHX4B
/WGivorCHbAB/WGi7orEHeIB/WGjDIrEHhQB/WGjF4rDHkYB/WGjF4rDHngB/WGjF4rDHqoB/WGjF4rDHTwB/WGjF4rDHw4B/WGjF4rDH0AB
/WGjF4rDH3IB/WGjF4rDH6QB/WGjF4rDH9UB/WGjF4rDIAgB/WGjGorOIDoB/WGjGorOIGwB/WGjGorOIJ4B/WGjGorOINAB/WGjGorOIQIB
/WGjGorOITQB/WGjGorOIWYB/WGjGorOIZgB/WGjGorOICoB/WGjGorOIfwB/WGjGorOIi4B/WGjGorOImAB/WGjEoraIpIB/WGjEoraIsQB
/WGjEIrWivYB/WGjD4sDIyGj/WGjD4sfiIloB/WGjDotCI4wB/WGjDotvI74B/WGjD4uji/AB/WGjEIVkJCIB/WGjEYwtJFQB/WGjE4yAJIYB
/WGjFYzeJLgB/WGjF41HJOoB/WGjGo26JRwB/WGjHI43JU4B/WGjHY60JYAB/WGjHo8gJbIB/WGjIIL+BjeB/WGjI4/TJhYB/WGjJZAUJkGjB
/WGjJpBDJnoB/WGjJpBhJqWb/WGjJ5BsJt4B/WGjJ5BsJxAB/WGjJ5BsJ0EB/WGjKpB5J3QB/WGjKpB5J6UB/WGjKpB5J9gB/WGjKpB5KAoB
/WGjKZCDKdWB/WGjKZCDK4B/WGjKZCDKAB/WGjKZCDKNIB/WGjKZCDKQB/WGjKZCDKTYB/WGjKZCDKWgB/WGjMJB5KZOB/WGjMJB5KcWB
/WGjMJB5Kf4B/WGjMJB5KjAB/WGjMpBvKmIB/WGjMpBvKpQB/WGjMpBvKsYB/WGjMpBvKvGjB/WGjMpBvKyOB/WGjMZBkK1WB/WGjMZBkK44B
/WGjMZBkK8AB/WGjMZBkK/IB/WGjMZBkLcMB
/WGjMZBkLFYB+WGjJpB3LcBzwGjJZChLloBjAGjHZDcLowBQQGjDJEkLR4A8QGjDJEkLVAAAACjDJEkLYIAAAA=</BDB>
</BIR>
```

Implementation

The SigObj object has been extended to provide conversions between FSS and ISO formats. As a security measure a signature collected by Wacom Signature Capture can only be saved as either FSS or ISO, not both. However after importing an ISO signature it can be saved in the FSS format. The option to encrypt the ISO signature data is included. The following table indicates the import/export options available:

Originating Source	Can export as ISO	Can save as FSS
Wacom Signature Capture	one or the other, but not both	
FSS (originating from Wacom Signature Capture)	-	✓
FSS (saved from ISO)	✓	✓

ISO	✓	✓
-----	---	---

Note that calling RenderBitmap with the option RenderEncodeData will effectively export FSS to the bitmap. To avoid attempting the output of both ISO and FSS formats the RenderEncodeData option will need to be avoided when exporting the ISO format.

API

Additions to fISigCOM for ISO support:

Classes
AdditionalImportIsoData
Enumerator Types
ImportIsoFlags
ExportIsoFlags

fISigCOM: SigObj

Method
ImportIso
ExportIso

Enumeration Types:

ImportIsoFlags

Specifies the ISO format expected for import.

Name	Value	Description
ImportIsoFlag_iso19784_7_binary	1	Binary format ISO
ImportIsoFlag_iso19785_3_xml	2	XML format ISO
ImportIsoFlag_iso19784_7_encrypted_binary	3	Binary encrypted ISO
ImportIsoFlag_iso19784_7_encrypted_text	4	Base64 encrypted ISO

ExportIsoFlags

Specifies the exported ISO format.

Name	Value	Description
ExportIsoFlag_iso19784_7_binary	1	Binary format ISO
ExportIsoFlag_iso19785_3_xml	2	XML format ISO
ExportIsoFlag_iso19784_7_encrypted_binary	3	Binary encrypted ISO
ExportIsoFlag_iso19784_7_encrypted_text	4	Base64 encrypted ISO

Class:

fISigCOM: : AdditionalImportIsoData

Summary

Property
Who
Why
When
TimeZoneOffset
ExtraData

Property	Type	SigObj Property Description
Who	String	Specifies the name of signatory of the imported signature.
Why	String	Specifies the reason for signing of the imported signature.
When	DATE	Specifies the local time and date when the signature was captured. An OLE Automation Date data type is expected. Note that to supply a date in Javascript the type must be explicitly converted, for example: <code>additionalData.When = (new Date(2018,6-1,27, 11,0,0)).getVarDate();</code> <code>// 27-06-2018 11:00:00</code>
TimeZoneOffset	Long	Time zone offset in MINUTES to UTC. For example -60 for UK BST.
ExtraData	String	ExtraData is a parameterized property that allows the client to store additional data within the signature object after capture. (See SigObj property ExtraData) For example: <code>additionalData.ExtraData("Key1") = "value for Key1";</code> <code>additionalData.ExtraData("Key2") = "value for Key2";</code>

FlSigCOM: ISO methods

ImportIsoData

Reads an ISO format signature into the signature object.

<i>ImportIso(iso, importIsoFlags, additionalIsoData)</i>	
Parameters	
iso	<p>Variant ISO data: String or Byte Array</p> <p>When the ImportIsoFlag_iso19785_3_xml flag is used, the variant iso is expected to be a string. While the whole document is scanned for compliance, only the root BIR is accessed and it is assumed the BDB block is not encrypted.</p> <p>If the ImportIsoFlag_iso19784_7_binary is used, the variant iso is expected to be a BYTE []. Additional data within the BDB block is ignored.</p>
importIsoFlags	<p>enum type ImportIsoFlags, specifies the ISO format expected for import:</p> <ul style="list-style-type: none"> • ImportIsoFlag_iso19784_7_binary • ImportIsoFlag_iso19785_3_xml • ImportIsoFlag_iso19784_7_encrypted_binary • ImportIsoFlag_iso19784_7_encrypted_text

additionalIsoData	object AdditionalImportData
<i>Return Value: ImportIsoRetVal</i>	

When the ImportIsoFlag_iso19784_7_encrypted_binary flag is used, the variant iso is expected to be a BYTE[] containing encrypted iso19784_7 binary data. The appropriate decryption key must be set, via SetProperty, before calling ImportISO

When the ImportIsoFlag_iso19784_7_encrypted_text flag is used, the variant iso is expected to be a string containing base64-encoded encrypted iso19784_7 binary data. As for ImportIsoFlag_iso19784_7_encrypted_binary, the decryption key must be set before calling ImportISO.

ImportIsoRetVal	Type	SigObj Property Description
Succeeded	Bool	True if call was successful, otherwise False
DiagnosticMessage	String	Plain text error message
DiagnosticCode	String	Plain text error code..

The diagnostic messages are not localised and are intended for use by Wacom developers in the case of support being needed.

ExportIsoData

Creates an ISO format output of the current signature. Export is only allowed if the SigObj has not been loaded from an FSS, or been saved as FSS (by accessing SigData or SigText). Either the BDB binary or a minimal XML wrapper can be retrieved. Attempting a disallowed export will result in an exception being thrown.

<i>ExportIso(exportIsoFlags)</i>	
<i>Parameters</i>	
exportIsoFlags	enum type exportIsoFlags, specifies the ISO format expected for import: <ul style="list-style-type: none"> ExportIsoFlag_iso19784_7_binary ExportIsoFlag_iso19785_3_xml ExportIsoFlag_iso19784_7_encrypted_binary ExportIsoFlag_iso19784_7_encrypted_text

For ExportIsoFlag_iso19784_7_encrypted_binary and ExportIsoFlag_iso19784_7_encrypted_text, An encryption key must be set via SetProperty prior to calling ExportIso

<i>Return Value: iso (or exception)</i>	
iso binary	BDB binary format ISO
iso xml	BDB xml format ISO
iso encrypted binary	BDB binary encrypted format ISO
iso Base64 encrypted	BDB Base64 encrypted format ISO

