Software Integration Guide

SigPlusExtLite Version 3.0

Contents

Introduction

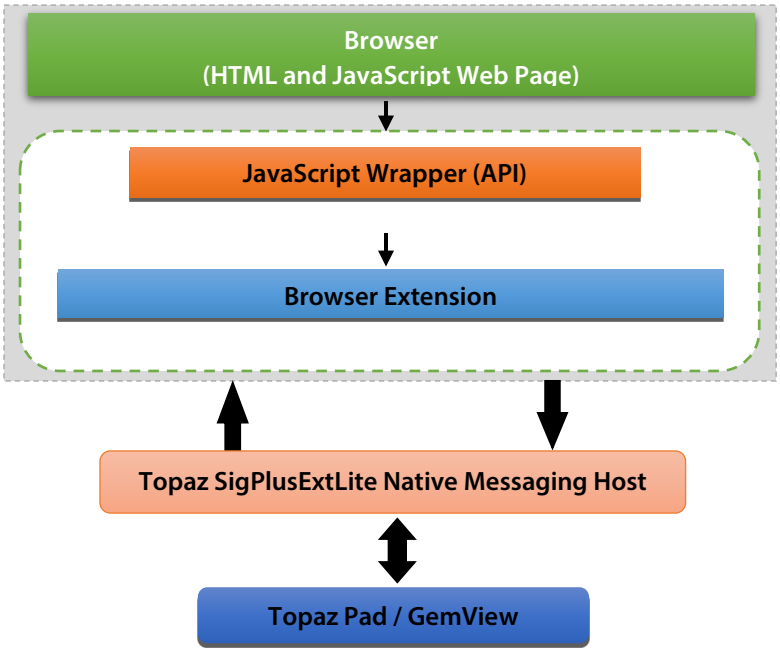
Topaz SigPlusExtLite allows web applications running in the Chrome, Firefox, Opera, and Edge (Chromium) browsers to capture handwritten signatures using Topaz signature pads & GemView tablet displays. Version 3.0 provides a simplified API designed to support more features of the Topaz SigPlus control for web use including the following:

- Simplified API for new designs
- Backward compatibility with SigPlusExtLite v2.0 and v1.0.
- Enable Topaz pads to display and clear signatures, text and pictures on pad LCD.
- Create and query "hot-spots" for onscreen controls on Topaz pads.
- Pen-down and pen-up events for use with Topaz signature pads.
- Get version and status of native message handler, extension and SigPlus control.
- Signing in place on web forms displayed on Topaz GemView Tablet Displays

Note: Legacy browsers EdgeHTML and IE are not supported.

Overview and Architecture

The SigPlusExtLite browser extensions provide web pages with the capability to capture signatures using Topaz signature pads and GemView tablet displays connected to client desktops. The diagram below provides a high-level overview.



Webpage

Web pages are HTML, JavaScript, and CSS based code that will be used to integrate the methods exposed by the Topaz SigPlusExtLite browser extensions. The methods exposed by the extension are be in the form of JavaScript functions (API).

JavaScript Wrapper

SigPlusExtLite v3 introduces a JavaScript wrapper in the extension that encapsulates the request/response mechanism using events and listeners. This wrapper simplifies the calls for the end user where the calls will be in form of a simple function call with parameters and return value. The function call is asynchronous, so the async/await javascript tags are used to await the returned value.

This allows for the new javascript functions to be easily accessed and for updates and bug fixes to be distributed automatically. It can be accessed through the "SigPlusExtLiteWrapperURL" attribute that is injected into the webpage by the extension. To include the SigPlusExtLiteWrapper.js, do the following:

```
/**get the SigPlusExtLiteWrapperURL attribute from the content script to read
the sigplusextlite wrapper javascript path*/
var url = document.documentElement.getAttribute('SigPlusExtLiteWrapperURL');
document.write('<script src='+url+'></script>');
```

The wrapper uses JavaScript Object Literals. Defining Object Literals allows grouping of relevant methods and simplifies the end user's implementation.

Browser Extension

Extensions are HTML, JavaScript, and CSS based code modules that can be downloaded and installed on individual browsers. The various components of extensions are background scripts, content scripts, UI elements and other files containing logic.

Native Messaging Host (NMH) Application

The Native Messaging Host Application is a Windows based application that is installed to the client machine. It acts as a bridge to enable communication between the browser and the Topaz device.

Key Features

The following is a summary of the key features:

- Signature capture using Topaz signature pads and GemView tablet displays
- Get version and status of native message handler and extension
- Generate pen down and pen up events
- Create and query "hot-spots" for creation of onscreen controls on LCD pads
- Download and update images on signature pads
- Enable 2nd screen form presentation and signing on Topaz tablet display products
- Allow signing in place on web forms displayed on tablet displays
- Customize the signature capture window displayed on all Topaz devices
- Export signature data in a Topaz SigString or base 64 format
- Export signature images as base 64 JPG or for further processing by the application
- Provides backward compatibility with SigPlusExtLite v2.0
- Capture live view of GemView screen

Operating System and Browser Support

SigPlusExtLite requires Windows 10 or later, or a Windows Server 2012 r2 or newer platform equipped with .NET Framework 4.7.1 or newer. The following are supported:

Chrome	**Firefox**	**Edge**	**Opera**
Version 77 and later	Version 68 and later	Version 80 and later	Version 68 and later

Signature Capture Devices

The SigPlusExtLite supports capturing signatures using both LCD and Non-LCD electronic signature pads and tablet displays from Topaz Systems.

SigPlusExtLite Methods

SigPlusExtLite methods are in the form of asynchronous JavaScript methods contained in Nested Object Literals that are referenced by a JavaScript wrapper file included within the extension. The following sections describe the various methods that are exposed by SigPlusExtLite through this wrapper.

Topaz Object

The Topaz object literal is the base object literal object that can be used to access and call each of the SigPlusExtLite methods. The following approach can be used to reference the nested objects for accessing SigPlusExtLite methods:

Syntax

```
var topaz = Topaz;
var global = topaz.Global;
```

Global Object

The Global object contains methods for performing generic actions.

Syntax

```
var global = Topaz.Global;
```

int Connect()

Method Description
Establishes a connection to the NMH.
Parameters
None
Return Value
1 = Connected successfully; 0 if unknown exception occurs; -1 = Error occurred while connecting to the NMH.

void Disconnect()

Method Description
Disconnects from the NMH.
Parameters
None
Return Value
None.

string GetSigPlusExtLiteVersion()

Method Description

Gets the version SigPlusExtLite (extension) installed.

Parameters

None

Return Value

String. Version number in the format: x.x.x.x (e.g.) 2.0.15.0.
Null value signifies an error.

string GetSigPlusExtLiteNMHVersion()

Method Description

Gets the version SigPlusExtLite NMH (Native Messaging Host) installed.

Parameters

None

Return Value

String. Version number in the format: x.x.x.x
Null value signifies an error.

string GetSigPlusActiveXVersion()

Method Description

Gets the version SigPlus ActiveX installed.

Parameters

None

Return Value

String. Version number in the format x.x.x.x
Null value signifies an error.

int GetDeviceStatus()

Method Description

Gets the status of the installed device.

Parameters

None

Return Value

0 = no device was detected; 1 = Topaz signature pad was detected; 2 = GemView Tablet Display was detected.
Negative value signifies an error. -1 = Error occurred while detecting the device; -2 = SigPlusExtLite is not installed; -3 = SigPlus drivers are not installed. -4 = an older version of SigPlus is installed.

string GetLastError()

Method Description

Gets the message that describes the last error.

Parameters

None.

Return Value

The message that describes the last error. Should be called immediately if any of the methods returns an error value. Returns an empty string if no error has occurred.

GemView Object

The GemView object contains methods for interacting with the GemView Tablet Display.

Syntax

```
var gemView = Topaz.GemView;
```

void PushCurrentTab()

Method Description

Pushes the tab from its current location on the monitor to the GemView screen.

Parameters

None

Return Value

None.

void RevertCurrentTab(int target)

Method Description

Reverts the tab from GemView back to the main monitor.

Parameters

target - 1 = Revert tab back to the same or original window; 2 = Revert tab to a new window.

Return Value

None.

void ModifyIdleScreenLogo(string logo)

Method Description

Modifies the idle screen logo.

Parameters

logo - Base64 encoded logo image data string. The image format should be at most a 60x60 PNG. The default is a 40x40 Topaz logo.

Return Value

None.

void ResetIdleScreenLogo()

Method Description

Resets the idle screen logo back to default Topaz logo.

Parameters

None.

Return Value

None.

void LoadIdleScreen(string configurationUrl, int duration, bool displayLogo)

Method Description

Loads an idle screen on the GemView device. Should be called when there is no activity on the GemView device.

Parameters

configurationUrl - Should be null if local default image is to be loaded. For custom image(s), a XML based configuration file should be provided as a URL. The XML schema for the configuration file is provided in the next section.

duration - Duration in milliseconds between idle screen images.

displayLogo - Flag indicating if the logo should be displayed. The default is a Topaz logo if ModifyIdleScreenLogo is not used to change the logo.

Return Value

None.

void CloseIdleScreen()

Method Description

Closes the Idle Screen on the GemView device.

Parameters

None

Return Value

None.

XML Schema for GemView

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <!-- the schema version -->
  <schema_version>1.0</schema_version>
  <!-- idle screen (attribute enabled = true/false) -->
  <idle_display enabled="true">
    <images>
      <!-- external links (attribute enabled = true/false) -->
      <external_links enabled="true">
                <!-- 7" GemView -->
                <device type="7">
                        <portrait>
                                <!-- list of external links -->
                                <external_link></external_link>
                                <external_link></external_link>
```

```
                    </portrait>
                    <landscape>
                        <!-- list of external links -->
                        <external_link></external_link>
                        <external_link></external_link>
                    </landscape>
                </device>
                <!-- 10" GemView -->
                <device type="10">
                    <portrait>
                        <!-- list of external links -->
                        <external_link></external_link>
                    </portrait>
                    <landscape>
                        <!-- list of external links -->
                        <external_link></external_link>
                    </landscape>
                </device>
                <!-- 16" GemView -->
                <device type="16">
                    <portrait>
                        <!-- list of external links -->
                        <external_link></external_link>
                    </portrait>
                    <landscape>
                        <!-- list of external links -->
                        <external_link></external_link>
                        <external_link></external_link>
                    </landscape>
                </device>
    </external_links>
  </images>
 </idle_display>
</configuration>
```

XML ELEMENT	ATTRIBUTE/VALUE	DESCRIPTION
schema_version	1.0	The XML schema version
idle_display	enabled = true or false	If false, then the idle screen will be disabled. If true, then the idle screen is displayed.
external_links	**enabled = true or false**	**If false, then GemView uses local images to load the idle screen else, it uses downloaded images to load idle screen.**
device	**type = 7,10,16**	**Images for different GemView devices (7”, 10”, 16”).**
portrait		**Grouping for portrait images**
landscape		**Grouping for landscape images**
external_link		**URL of the image to load idle screen in GemView.**

Table 1 Configuration XML Schema for Idle Screen files

To improve performance and to avoid repeatedly downloading large image files, SigPlusExtLite first checks the hash generated from these files. If the hash does not match with any of the image files already present, it will assume the image to be a new one and download it and save it to a local repository. The hash algorithm used to generate the image hash must be SHA256. It is mandatory to create a SHA256 hash and save it as text file (.txt) at the same location as the image. The file names of the image and the text file should be identical as SigPlusExtLite looks for .txt file with the same name as the image file first.

E.g., if the image file name is logo.png then the text file with SHA256 hash should be logo.txt
If the text file of the hash is not found, the image will not be downloaded or displayed in the idle screen.

void StartCaptureGemViewScreen(function renderCapturedGemViewScreen, int delay, bool enableRequestAnimationFrame)

Method Description
Start capturing a live view of the tab pushed to the GemView screen using the PushCurrentTab method.
Parameters
renderCaptureGemViewScreen – function that continuously receives an image containing the present view of the tab pushed to the GemView and an error message, if any. The captured image can be added to a canvas or other element to present a live view of the tab pushed to the GemView.
delay – control the delay between capturing a new live view image. This value defaults to 250 millseconds. It is recommended not to modify this value.
enableRequestAnimationFrame – if enabled, then triggers requestAnimationFrame when performing the rendering of the live view. This value defaults to true. It is recommended not to modify this value.
Return Value
None.

void StopCaptureGemViewScreen()

Method Description
Stop capturing a live view of the tab pushed to the GemView screen.
Parameters
None.
Return Value
None.

Canvas Object

The Canvas object literal is the base object literal object that can be used to access and call the Sign and LCD objects' methods. It can be used for interacting directly with a Topaz signature pad. The following approach can be used to reference the nested objects for accessing the Sign and LCD objects' methods:

Syntax

```
var canvas = Topaz.Canvas;
var sign = canvas.Sign;
var lcd = canvas.LCD;
```

1 - Sign Object

The Sign object contains methods for interacting with a Topaz signature pad directly to sign in an HTML canvas element.

Syntax

```
var sign = Topaz.Canvas.Sign;
```

void SetTabletState(int state)

Method Description
Enables signature capture through the signature pad.
Parameters
state – Setting the tablet state to 1 enables the tablet to capture signatures. Setting the tablet state to 0 disables the tablet to deactivate signature capture.
Return Value
None.

int GetTabletState()

Method Description
Gets the capture state of the tablet.
Parameters
None.
Return Value
Integer. The tablet state. 1 = Tablet open for signature capture; 0 = Off.

void SetTabletCOMPort(int port)

Method Description
Sets the tablet's COM port.
Parameters
port - The COM port into which the signature pad has been plugged. Generally, the value is a number between 1 and 99. For BSB pads, it is usually 9. It can only be set when the SetTabletState is set to 0 (Off).
Return Value
None.

int GetTabletCOMPort()

Method Description
Gets the serial tablet's COM port.
Parameters
None.
Return Value
The COM port number. COM port only affects serial and BSB signature pads. This includes B, BSB, and BBSB models. Typically, BSB signature pads will be set to COM9.

int SetSigStringFormat(int encryptionMode, string encryptionKey, int sigCompressionMode)

Method Description

Sets the format in which the SigString data should be exported.

Parameters

encryptionMode - 0 = Clear text; 1 = DES Encryption; 2 = higher security encryption mode.
encryptionKey - string that contains information/data that will be hashed and used as a key within the encryption method.
sigCompressionMode - 0 = no compression, 1 = lossless compression and 2 = lossy. 2 is not recommended unless storage size is critical. Default value is 1.

Return Value

Integer. 1 = SigString data format was set successfully, negative value signifies an error. -1 = Error occurred while setting the SigString data format.

void SetEncryptionMode(int mode)

Method Description

Sets the Encryption mode. This should be used in conjunction with SetAutoKeyData method.

Parameters

mode - Level of encryption. 0 = Clear text; 1 = DES Encryption; 2 = higher security encryption mode.

Return Value

None.

int GetEncryptionMode()

Method Description

Gets the current level of encryption.

Parameters

None.

Return Value

The current level of encryption.

void SetAutoKeyData(string keyData)

Method Description

Adds data to the AutoKey generation function, distilling it down to a set-length key, which will ultimately encrypt the signature data. Should be used with SetEncryptionMode and SetSigString or GetSigString.

Parameters

keyData - String containing the data that will be used for key generation. This is the same as the encryptionKey parameter used in SetSigStringFormat method.

Return Value

None.

void SetSigCompressionMode(int mode)

Method Description

Sets the compression mode for signatures.

Parameters

0 = no compression, 1 = lossless compression and 2 = lossy. 2 is not recommended unless storage size is critical. *Default value is 1.*

Return Value

None.

int GetSigCompressionMode()

Method Description

Gets the current signature compression mode.

Parameters

None.

Return Value

The current signature compression mode.

void StartSign(Canvas canvas)

Method Description

Starts the process of signing on an HTML canvas

Parameters

canvas - The canvas element on the HTML page. The signature is rendered on the canvas provided.

Return Value

None.

Remarks

The live signature rendering can also be programmatically reconfigured by setting the TabletState function to connect to the signature pads and customizing the callback function attached to the "SignReponse" event. This can be done to trim or rearrange the live signature displayed in the canvas. The following is an example of how this can be achieved.

Connect to the signature pad and set the callback function to receive the live image data.

```
var sign = Topaz.Canvas.Sign;
sign.SetTabletState(1); //Connect to signature pad
top.document.addEventListener("SignResponse", SignResponse, false);
```

Define the callback function that will customize the live rendering of the signature.

```
async function SignResponse() {
  var responseObj = event.target.getAttribute("msg-Attribute");
  var obj = JSON.parse(responseObj);
  var img = new Image();
  var canvasObj = document.getElementById('cnv');
  var ctx = canvasObj.getContext('2d');
  var trimHeight = 100;
  img.onload = function () {
          ctx.drawImage(img, 0, trimHeight, canvasObj.width, canvasObj.height,
      0, 0, canvasObj.width, canvasObj.height);
  }
  img.src = '' + obj.imageData;
}
```

Disconnect from the signature pad and remove the callback function that is actively listening for live signature data.

```
var sign = Topaz.Canvas.Sign;
await sign.SetTabletState(0); //Disconnect from signature pad
top.document.removeEventListener("SignResponse", SignResponse, false);
```

void StopSign()

Method Description

Stops the process of signing.

Parameters

None.

Return Value

None.

void ClearSign()

Method Description

Clears the signature from the canvas and underlying SigPlus control.

Parameters

None.

Return Value

None.

int GetTotalPoints()

Method Description

Gets the total number of points in the current signature. Can be used to detect if a signature is present or not.

Parameters

None.

Return Value

The number of points in the signature.

int GetNumberOfStrokes()

Method Description

Gets the total number of strokes in the current signature.

Parameters

None.

Return Value

The total number of strokes in the signature.

int GetNumberOfPointsInStroke(int strokeNo)

Method Description

Gets the number of points in a stroke in the current signature.

Parameters

strokeNo - The stroke number index in the signature.

Return Value

The total number of points in the specified stroke.

int GetPointXValue(int strokeId, int pointId)

Method Description

Gets the X coordinate of the specified point.

Parameters

strokeId - The index of the stroke for the point.
pointId - The index of the point in the stroke.

Return Value

The X coordinate of the specified point.

int GetPointYValue(int strokeId, int pointId)

Method Description

Gets the Y coordinate of the specified point.

Parameters

strokeId - The index of the stroke for the point.
pointId - The index of the point in the stroke.

Return Value

Integer. The Y coordinate of the specified point.

string GetSignatureImage()

Method Description

Gets the signature image in base64 format.

Parameters

None.

Return Value

Signature image as base 64 string data. The format will be JPG. Returns an empty base 64 string if the signature is not captured.

void SetSigString(string sigString, int encryptionMode, string encryptionKey, int sigCompressionMode)

Method Description

Loads the signature data into the control in ASCII Data compatible format. Data is in the form of an ASCII string. Used to retrieve or place sig data in the control as a property rather than as a method. The data format is Memo Field, ASCII, and Unicode compatible.

Parameters

sigString - String containing the signature data.
encryptionMode - 0 = Clear text; 1 = DES Encryption; 2 = higher security encryption mode.
encryptionKey - string that contains information/data that will be hashed and used as a key within the encryption method.
sigCompressionMode - 0 = no compression, 1 = lossless compression and 2 = lossy. 2 is not recommended unless storage size is critical. Default value is 1.

Return Value

None

string GetSigString()

Method Description

Gets the signature data in Topaz SigString format. The signature data is in ASCII Data compatible format. Data is in the form of an ASCII string.

Parameters

None.

Return Value

Signature data in Topaz SigString format. The data is based on the inputs provided in the SetSigStringFormat method or the combination of SetEncryptionMode, SetAutoKeyData and SetSigCompressionMode. Returns an empty string if the signature is not captured.

void SetImagePenWidth(int width)

Method Description

Sets the pen width to use for the signature in the image. This command does not affect the pen width shown in the control signature window. Variation can appear in perceived pen thickness depending upon the x and y resolution selected for the image.

Parameters

width - The pen width in pixels used for signature in the image. Default is 1.

Return Value

None.

int GetImagePenWidth()

Method Description

Gets the current pen width used for signature in the image.

Parameters

None.

Return Value

The current pen width used for signature in the image.

void SetDisplayPenWidth(int width)

Method Description

Sets the pen width for the displayed signature in the canvas.

Parameters

width - The pen width for the displayed signature in pixels.

Return Value

None.

int GetDisplayPenWidth()

Method Description

Gets the pen width in pixels for the displayed signature in the canvas.

Parameters

None.

Return Value

The pen width for the displayed signature.

void AddPenUpEventListener()

Method Description

Enables the PenUp event through the signature pad.

The PenUp event handling is a three-step process.

1. Call AddPenUpEventListener() in the Topaz.Canvas.Sign Object literal to listen pen up event response.

```
let sign = Topaz.Canvas.Sign
// Add PenUp event listener
await sign.AddPenUpEventListener();
```

2. Add a response function to process the event.

```
OnSignPenUp = function(){
        // Process pen up event
};
```

3. Remove PenUp event listener once the process is completed.

```
await sign.RemovePenUpEventListener();
```

Parameters

None

Return Value

None

void RemovePenUpEventListener()

Method Description

Removes the PenUp event listener.

Parameters

None

Return Value

None

void AddPenDownEventListener()

Method Description

Enables the PenDown event through the signature pad.

The PenDown event handling is a three-step process.

1. Call AddPenDownEventListener() in the Topaz.Canvas.Sign to listen pen down event response.

```
let sign = Topaz.Canvas.Sign;
// Add PenDown event listener
await sign.AddPenDownEventListener();
```

2. Add a response function to process the event.

```
OnSignPenDown = function(){
        // Process pen down event
};
```

3. Remove PenDown event listener once the process is completed.

```
await sign.RemovePenDownEventListener();
```

Parameters

None

Return Value

None

void RemovePenDownEventListener();

Method Description

Removes the PenDown event listener.

Parameters

None

Return Value

None

2 - LCDTablet Object

The LCDTablet object contains methods for interacting with a Topaz signature pad directly to control the LCD screen.

Syntax

```
var lcd = Topaz.Canvas.LCDTablet;
```

int GetTabletModelNumber()

Method Description
Gets the model number of the tablet.
Parameters
None.
Return Value
Integer. The tablet model number. 1 = TL(BK)766, 8 = TL(BK)755 or TL(BK)750, 11 or 12 = TL(BK)462, 15 = TL(BK)460, 43 = TLBK43LC, 57 = TLBK57GC, 58 = All Topaz "SE" signature pad models.
Negative value signifies an error. -1 = Error occurred while retrieving the tablet model number.

int GetTabletSerialNumber()

Method Description
Gets the serial number of the tablet. This can be a valuable tool for determining which "SE" model you have exactly. For e.g. if the model number is 58, you can check for the specific serial number. If you have multiple LCD1x5 tablets (either T-L(BK)462 or T-L(BK)460 tablets), you can also use the tablet serial number to uniquely identify each signature pad.
Parameters
None.
Return Value
Integer. The tablet serial number. 550 = TLBK766SE, 551 = TLBK462SE, 553 or 557 = TLBK755SE or TLBK750SE.
Negative value signifies an error. -1 = Error occurred while retrieving the tablet serial number.

void SetTabletLogicalXSize(int xSize)

Method Description
Sets the range of horizontal values to be used to represent signatures in Logical Tablet Coordinates. This is not related to the displayed image file, or tablet sizes. This is the X-range used for the Topaz vector format, the internally used format. In most applications, this should be set to match the active tablet X resolution.
Parameters
xSize - The tablet logical X size. Default is 2150.
Return Value
None.

int GetTabletLogicalXSize()

Method Description

Get the current logical X size used to represent signatures in Logical Tablet Coordinates. This can be used to set the drawing area in conjunction with logical Y size.

Parameters

None.

Return Value

The Tablet Logical X size.

void SetTabletLogicalYSize(int ySize)

Method Description

Sets the range of vertical values to be used to represent signatures in Logical Tablet Coordinates. This is not related to the displayed image file, or tablet sizes. This is the Y-range used for the Topaz vector format, the internally used format. In most applications, this should be set to match the active tablet Y resolution.

Parameters

ySize - The tablet logical Y size. Default is 1400.

Return Value

None.

int GetTabletLogicalYSize()

Method Description

Get the current logical Y size used to represent signatures in Logical Tablet Coordinates. This can be used to set the drawing area in conjunction with logical X size.

Parameters

None.

Return Value

The tablet logical Y size.

void SetJustifyMode(int justifyMode)

Method Description

Sets the current justification mode i.e. how the signature is sized and positioned in the signature box (does not affect the signature data).

Parameters

justifyMode - The justification mode. 0 = Normal - No justification (default); 1 = Justify and zoom signature (upper left corner); 2 = Justify and zoom signature (upper right corner); 3 = Justify and zoom signature (lower left corner); 4 = Justify and zoom signature (lower right corner); 5 = Justify and zoom signature (center of control).

Return Value

None.

int GetJustifyMode()

Method Description

Gets the current justification mode i.e. how the signature is sized and positioned in the signature box.

Parameters

None.

Return Value

The justification mode. 0 = Normal - No justification (default); 1 = Justify and zoom signature (upper left corner); 2 = Justify and zoom signature (upper right corner); 3 = Justify and zoom signature (lower left corner); 4 = Justify and zoom signature (lower right corner); 5 = Justify and zoom signature (center of control).

void SetJustifyX(int buffer)

Method Description

Sets the buffer size in the logical tablet coordinates as a "signature free zone" of the left and right edge of the SigPlus object. Default is 0.

Parameters

buffer - Justification X buffer size in pixels to be set.

Return Value

None.

int GetJustifyX()

Method Description

Gets the buffer size of the left and right edge (default is 0).

Parameters

None.

Return Value

The buffer size of the left and right edge.

void SetJustifyY(int buffer)

Method Description

Sets the buffer size in the logical tablet coordinates as a "signature free zone" of the top and bottom edge of the SigPlus object. Default is 0.

Parameters

buffer - Justification Y buffer size in pixels to be set.

Return Value

None.

int GetJustifyY()

Method Description

Gets the buffer size of the top and bottom edge (default is 0).

Parameters

None.

Return Value

The buffer size of the top and bottom edge.

int GetLCDSize()

Method Description

Requests the size of the LCD in question. Returned as one value - part of the value is in the low bytes and the other part of the value is in the high bytes.

Parameters

None.

Return Value

Size of the LCD as a single integer value.
Negative value signifies an error. -1 = Error occurred in getting the size of the LCD.

void SetLCDPixelDepth(int pixelDepth)

Method Description

Used to specify color or black and white images when passing an image to paint to the LCD using the LCDWriteImage function. Call this method appropriately prior to painting to the LCD.

Parameters

pixelDepth - The pixel depth. Depth = 0 - Tells SigPlus to expect a black and white image for painting. Depth = 8 - Tells SigPlus to expect a color image for painting.

Return Value

None.

bool SetLCDCompression(int lcdCompMode, int lcdZCompMode, int lcdCompFast, int lcdCompSlow)

Method Description

Used to specify whether to include data compression when writing text/graphics to a "SE" signature pad, the T-LBK57GC pad, or the T-LBK43LC pad. Also, used to specify the level of compression to use (max = 127 for both fast and slow compression).

Parameters

lcdCompMode – 0 = Off, 1 = On.
lcdZCompMode - 0 = Off, 1 = On.
lcdCompFast – level of compression (max = 127).
lcdCompSlow – level of compression (max = 127).

Return Value

None.

int LCDSetWindow(int xPos, int yPos, int width, int height)

Method Description

Sets a signature window that restricts the ink of the SigPlus object to the window on the LCD itself (associated with the SetSigWindow method).

Parameters

xPos – start position of x coordinates.
yPos – start position of y coordinates.
width – width of the image in LCD pixels.
height – height of the image in LCD pixels.

Return Value

1 = Window set to LCD device successfully. Negative value signifies an error. -1 = Error occurred setting the window to the LCD device.

void SetImageWidth(int width)

Method Description

Sets the width of the image in pixels.

Parameters

Width of the image in pixels.

Return Value

None.

Note: Method name changed from SetImageXSize (refer to SigWeb document) as it provides more clarity.

int GetImageWidth()

Method Description

Gets the current width of the image in pixels.

Parameters

None

Return Value

The current image width. If 0, then value is specified by the LogicalXSize.

Note: Method name changed from GetImageXSize (refer to SigWeb document) as it provides more clarity.

void SetImageHeight(int height)

Method Description

Sets the height of the image in pixels.

Parameters

height as integer. Height of the image in pixels.

Return Value

None.

Note: Method name changed from SetImageYSize (refer to SigWeb document) as it provides more clarity.

int GetImageHeight()

Method Description

Gets the current height of the image in pixels.

Parameters

None

Return Value

The current image height. If 0, then value is specified by the LogicalYSize.

Note: Method name changed from GetImageYSize (refer to SigWeb document) as it provides more clarity.

int LCDWriteImage(int destination, int mode, int xPos, int yPos, int width, int height, int format, string url)

Method Description

Writes an image to the LCD using the URL specified.

Parameters
destination - The image destination. 0 = Write image to the LCD directly (the foreground); 1 = Write image to the background memory of the tablet. Use LCDRefresh to bring it to the foreground.
mode - The LCD mode: 0 = Clear - LCD display is cleared at the specified location; 1 = Complement - Invert the pixels at the given location; 2 = Write Opaque - Transfers contents of the background memory to the LCD display, overwriting the content of the LCD display; 3 = WriteTransparent: The contents of the background memory in the tablet are combined with and transferred to the visible LCD memory.
xPos – start positions of the x coordinates.
yPos – start positions of the y coordinates.
width – width of the image in LCD pixels.
height – height of the image in LCD pixels.
format - The image format. 0 = Compressed BMP (default) must have .bmp extension; 1 = Uncompressed BMP must have .bmp extension; 2 = Mono. BMP must have .bmp extension; 3 = JPG Q=20 must have .jpg extension; 4 = JPG Q=100 must have .jpg extension; 5 = Uncompressed TIF must have .tif extension; 6 = Compressed TIF must have .tif extension; 7 = WMF (windows metafile) must have .wmf extension; 8 = EMF (enhanced metafile) must have .emf extension; 9 = TIF (1-bit) must have .tif extension; 10 = TIF (1-bit inverted) must have .tif extension; 11 = TIF (Group 4) must have .tif extension.
url – url path to the image to be displayed.
Return Value
1 = Image written to LCD device successfully; negative value signifies an error. -1 = Error occurred while writing the image to the LCD device.

int LCDWriteString(int destination, int mode, int xPos, int yPos, string text)
Method Description
Writes a string to the LCD using the text specified.
Parameters
destination - 0 = Write string to the LCD directly (the foreground); 1 = Write string to the background memory of the tablet.
mode - The LCD mode. 0 = Clear - LCD display is cleared at the specified location; 1 = Complement - Invert the pixels at the given location; 2 = Write Opaque - Transfers contents of the background memory to the LCD display, overwriting the content of the LCD display; 3 = WriteTransparent: The contents of the background memory in the tablet are combined with and transferred to the visible LCD memory.
xPos – start position x coordinates.
yPos – start position y coordinates.
text - The text to be displayed on the LCD screen.
Return Value
1 = String written to LCD device successfully; negative value signifies an error. -1 = Error occurred while writing the string to the LCD device.

void LCDSetFont(int height, int weight, int italic, int underline, int pitchAndFamily, string faceName)

Method Description

Sets the size, type, and properties of font used when calling the LCDWriteString method. The arguments are all defined in the LOGFONT data structure (see CreateFont function of Windows API) in Windows for logical fonts.

Parameters

height - Height of font in pixels
weight - Font weight as a number between 0 and 900. 0=default, 400=normal, and 700=bold.
italic - If this value is non-zero, the text is italicized.
underline - If this value is non-zero, the text is underlined.
pitchAndFamily - Specifies the pitch (fixed or variable width) and font family used if the font you request is unavailable. If you specify a font that's likely to be, then this argument can be left as 0.
faceName - Font's name—for example, "Times New Roman", "Courier New", "Arial"

Return Value

None.

void SetLCDCaptureMode(int mode)

Method Description

Sets the current LCD Capture Mode.

Parameters

mode - The LCD capture mode to use. 1 = The tablet is set for 'auto erase' mode (clears LCD in around 4 seconds); 2 = The tablet is set to persistent ink capture (required for using the LCD to display text/graphics). This keeps the data on the tablet until LCDRefresh is called.

Return Value

None.

int GetLCDCaptureMode()

Method Description

Gets the current LCD capture mode.

Parameters

None.

Return Value

The mode LCD is set to capture signatures in. 1 = The tablet is set for 'auto erase' mode (clears LCD in basically 4 seconds); 2 = The tablet is set to persistent ink capture (required for using the LCD to display text/graphics). This keeps the data on the tablet until LCDRefresh is called.
Negative value signifies an error. -1 = Error occurred while retrieving the LCD capture mode.

void LCDRefresh(int mode, int xPos, int yPos, int width, int height)

Method Description

Sends a refresh command to the device.

Parameters

mode - 0 = Clear - LCD display is cleared at the specified location; 1 = Complement inverts the pixels at the given location; 2 = Write Opaque transfers contents of the background memory to the LCD display, overwriting the content of the LCD display.
xPos – start position of the X coordinate.
yPos - start position of the Y coordinate.
width – width of area in LCD pixels.
height - height of area in LCD pixels.

Return Value

None.

int SetSigWindow(int coord, int xPos, int yPos, int width, int height)

Method Description

Sets a window in the control and for ink to render inside. This is used to separate signature area from hot spots (see KeyPadAddHotSpot for more details).

Parameters

coord - Coordinate system used for this window (inking area). 0 = Logical tablet coordinates; 1 = LCD coordinates (generally LCD coordinates are used).
xPos - starting X positions for the SigWindow.
yPos - starting Y positions for the SigWindow.
width – width of the window in specified coordinates system.
height - height of the window in specified coordinates system.

Return Value

1 = SigWindow was set successfully; negative value signifies an error. -1 = Error occurred while setting the SigWindow.

Remarks

This function sets a window in the logical tablet space that restricts the operation of some functions to the specified window.
The functions behave as follows:

JustifyMode will only operate on points inside of this window.
ExportSigFile and WriteImageFile will only operate on points inside the window.
SigString only operates on points inside of the window.
ClearTablet will only clear in the window.

This behavior is enabled by setting the start and stop values to non-zero. The window defaults to (0,0,0,0). The window can be enabled at one spot, re-enabled at another, etc., without disabling in between, and then disabled when the various parts of the tablet data have been separated and stored. To determine the logical values in the control for the installed tablet, see the TabletLogicalXSize and TabletLogicalYSize properties.

int KeyPadAddHotSpot(int keyCode, int coord, int xPos, int yPos, int width, int height)

Method Description

Defines the location of a tablet hot spot, which is used by the developer to detect pen taps. Should be used with pen events (PenUp and/or PenDown). Never place hot spots inside of a SigWindow. See SetSigWindow for more details.

Parameters

keyCode - Value defining the hot spot index (should be unique).
coord - Coordinate system used for this 'hot spot' (generally this is 1 for LCD coordinate points).
xPos – X position to start.
yPos - Y positions to start.
Width – width in LCD coordinates.
Height - height in LCD coordinates.

Return Value

1 = Hotspot was added successfully; negative value signifies an error. -1 = Error occurred while adding the hotspot.

int KeyPadQueryHotSpot(int keyCode)

Method Description

Queries whether the specified hot spot has been tapped by the user. Returns a value if the control contains data that is within the hot spot (definition of the KeyCode) on the tablet.

Parameters

keyCode - Hot spot (KeyCode definition).

Return Value

The number of points within the hot spot KeyCode definition.

void KeyPadClearHotSpotList()

Method Description

Clears the control's internal list of hot spots created. This includes all hot spots created at all indices.

Parameters

None.

Return Value

None.

void ClearSigWindow(int inside)

Method Description

Erases the data either inside or outside the SigWindow.

Parameters

Inside - Value: 0 = Data is erased from the SigWindow. 1 = Data is removed outside of the SigWindow (for clearing the hot spot buffer).

Return Value

None.

void ClearTablet()

Method Description

Clears the signature object of any ink in the control. To clear ink from the LCD, refer to LCDRefresh method.

Parameters

None.

Return Value

None.

Signature Capture Window

The SignatureCaptureWindow object literal is the base object literal object that can be used to access and call the Sign and CustomWindow object's methods. It can be used for opening a signature capture window to collect the signature for a Topaz signature pad or GemView Tablet Display. The following approach can be used to reference the nested objects for accessing the Sign and CustomWindow object's methods:

Syntax

```
var signatureCaptureWindow = Topaz.SignatureCaptureWindow;
var sign = signatureCaptureWindow.Sign;
var customWindow = signatureCaptureWindow.CustomWindow;
```

1 – Sign Object

The Sign object contains methods for opening a signature capture window to collect the signature for a Topaz signature pad or GemView Tablet Display.

Syntax

```
var sign = Topaz.SignatureCaptureWindow.Sign;
```

int SetImageDetails(int format, int width, int height, bool transparency, bool scaling, int maxUpScalePercent)

Method Description

Sets the image details for the signature.

Parameters

format - Format of the signature image to be exported after signature capture. 1 = JPG; 2 = PNG. The default format is JPG if not set. Width and height in pixels of signature image to be exported. Default to 300, 150 if not set.
width – width of the signature image.
height – height of the signature image.
transparency - Transparency flag indicating if the signature image background should be transparent. Default transparency is set to false.
scaling - Flag indicating if the signature image should be scaled.
maxUpScalePercent - maximum allowed upscale percentage (0 to 100). Defaults to 25 if not set.

Return Value

1 = image details were set successfully, negative value signifies an error. -1 = Error occurred while setting the image details.

int SetPenDetails(string colorcode, int thickness)

Method Description

Sets the pen details for the signature.

Parameters

colorcode - HTML color code as hexadecimal string e.g. #5882FA". Defaults to "#000000" if not set.
Thickness - Thickness of the pen stroke in pixels (1 to 5). Defaults to 2 if not set.

Return Value

1 = pen details were set successfully, negative value signifies an error. -1 = Error occurred while setting the pen details.

int SetMinSigPoints(int points)

Method Description

Sets the minimum number of points collected required to qualify the signer marks as a valid signature.

Parameters

points - The minimum number of points required to qualify the signer marks as a valid signature. The default is 25.

Return Value

Integer. 1 = Minimum signature points were set successfully, negative value signifies an error. -1 = Error occurred while setting the minimum signature points.

void StartSign(bool showCustomWindow, int sigCompressionMode, int encryptionMode, string encryptionKey)

Method Description

Starts the process of signing on a signature window. (Parameters are optional)

Parameters

showCustomWindow – True if the custom window is to be displayed. False if the default window is to be displayed. If not set, defaults to false.

sigCompressionMode – 0 = no compression, 1 = lossless compression and 2 = lossy. 2 is not recommended unless storage size is critical. If not set, defaults to 1.

encryptionMode – 0 = Clear text; 1 = DES Encryption; 2 = higher security encryption mode. If not set, defaults to 0.

encryptionKey – string that contains information/data that will be hashed and used as a key within the encryption method. If not set, defaults to "".

Return Value

None.

bool IsSigned()

Method Description

Gets a value indicating if the signature was successful. Should be checked after calling the StartSign method.

Parameters

None

Return Value

True if the signature was successful. False in case of failure.

void SignComplete()

Method Description

Completes the process of signing. Clears any temporary signature data stored and resets the process for the next signature. Should be called after getting the signature related information such as signature image, signature data, pad information etc. If the LoadSignatureCaptureWindow was called, then the signature capture window will be unloaded.

Parameters

None.

Return Value

None.

void LoadSignatureCaptureWindow(bool showCustomWindow)

Method Description

Loads the signature capture window to decrease any delays that may occur when starting the signature capture window when StartSign is called. SignComplete can be called to unload the signature capture window.

Parameters

showCustomWindow – True if the custom window is to be displayed. False if the default window is to be displayed. If not set, defaults to false.

Return Value

None.

string GetSignatureImage()

Method Description

Gets the signature image in base64 format.

Parameters

None.

Return Value

Signature image as base 64 string data. The format will be JPG. Returns an empty string if the signature is not captured.

string GetSigString()

Method Description

Gets the signature data in Topaz SigString format. The signature data is in ASCII Data (VB script) compatible format. Data is in the form of an ASCII string.

Parameters

None.

Return Value

Signature data in Topaz SigString format.

2 – CustomWindow Object

The CustomWindow object contains methods for customizing the signature capture window.

Syntax
The setters and getters for different attributes or properties are defined as methods below. The setter statements are only applied once the save method is called.

```
var customWindow = Topaz.SignatureCaptureWindow.CustomWindow;
```

void SetSigningWindowTitle(string title)
Method Description
Sets the title of the signing window.
Parameters
title - Title text. Defaults to "Topaz SigPlusExtLite" if not set.
Return Value
None.

string GetSigningWindowTitle()
Method Description
Gets the title of the signing window.
Parameters
None.
Return Value
The signing window title.

void SetSigningWindowBackColor(string colorCode)
Method Description
Sets the background color of the signing window.
Parameters
colorCode - The colorCode should be a HTML color code as hexadecimal string e.g. #5882FA". Defaults to "#F0F0F0" if not set.
Return Value
None.

string GetSigningWindowBackColor()
Method Description
Gets the background color of the signing window.
Parameters
None.
Return Value
The HTML color code as hexadecimal string. E.g. "#F0F0F0".

void SetSigningWindowSize(int width, int height)

Method Description

Sets the size of the signing window.

Parameters

width - The width of the signing window in pixels. Defaults to 785 pixels if not set.
height - The height of the signing window in pixels. Defaults to 340 pixels if not set.

Return Value

None.

Remarks

The signing window will have restrictions on minimum and maximum values for width and height. For Topaz signature pads the minimum width and height is 785 and 340 pixels respectively, while the maximum width and height of GemView depends on the display size. The table below shows the minimum and maximum values (width and height) for different GemView devices.

GEMVIEW TYPE	MINIMUM SIZE		MAXIMUM SIZE		GEMVIEW MODE
	WIDTH	HEIGHT	WIDTH	HEIGHT	
7 inch	785	340	1024	560	Landscape
	600	340	600	984	Portrait
10 inch	785	340	1280	760	Landscape
	785	340	800	1240	Portrait
16 inch	785	340	1366	728	Landscape
	768	340	768	1326	Portrait

Table 29 GemView signing windows size restrictions

Size can only be defined when the window state is set to "Normal". For more information, see The SetSigningWindowState method.

int GetSigningWindowWidth()

Method Description

Gets the width of the signing window in pixels.

Parameters

None.

Return Value

The width of the signing window in pixels.

int GetSigningWindowHeight()

Method Description

Gets the height of the signing window in pixels.

Parameters

None.

Return Value

The height of the signing window in pixels.

void SetSigningWindowLocation(int x, int y)

Method Description

Sets the location of the signing window.

Parameters

x - The x coordinate of the signing window location. Defaults to 0 if not set.
y - The y coordinate of the signing window location. Defaults to 0 if not set.

Return Value

None.

Remarks

Location can only be defined when the window state is set to "Normal". For more information, see SetSigningWindowState method.

int GetSigningWindowLocationX()

Method Description

Gets the x coordinate of the signing window location.

Parameters

None.

Return Value

The x coordinate of the signing window location.

int GetSigningWindowLocationY()

Method Description

Gets the y coordinate of the signing window location.

Parameters

None.

Return Value

The y coordinate of the signing window location.

void SetSigningWindowState(int windowState)

Method Description

Sets the window state of the signing window.

Parameters

windowState - 0 = Normal State; 2 = Maximized State. Defaults to 0 (Normal) if not set.

Return Value

None.

int GetSigningWindowState()

Method Description

Gets the window state of the signing window.

Parameters

None.

Return Value

Integer. The window state of the signing window; 0 = Normal State; 2 = Maximized State.

void SetSigningWindowBorderStyle(int borderStyle)

Method Description

Sets the border style of the signing window.

Parameters

borderStyle - 0 = None; 1 = Fixed Single; 2 = Fixed 3D; 3 = Fixed Dialog; 4 = Sizable. Defaults to 3 (Fixed Dialog) if not set.

Return Value

None.

int GetSigningWindowBorderStyle()

Method Description

Gets the border style of the signing window.

Parameters

None.

Return Value

The border style of the signing window; 0 = None; 1 = Fixed Single; 2 = Fixed 3D; 3 = Fixed Dialog; 4 = Sizable.

void SetSigningWindowToolbarBackColor(string colorCode)

Method Description

Sets the background color of the signing window toolbar.

Parameters

colorCode - The HTML color code as hexadecimal string e.g. #5882FA". Defaults to "#FED7B0" if not set.

Return Value

None.

string GetSigningWindowToolbarBackColor()

Method Description

Gets the background color of the signing window toolbar.

Parameters

None.

Return Value

The HTML color code as hexadecimal string. E.g. "#F0F0F0";

void SetSigningWindowToolbarSize(int size)

Method Description

Sets the size of the signing window toolbar.

Parameters

size - Size is calculated in pixels. Width is applied for left and right docked toolbars, height is applied for top, and bottom docked toolbars. Defaults to 30 pixels if not set.

Return Value

None**.**

Remarks

The signing toolbar will have restrictions on minimum and maximum values for size. The size attribute is dependent on the dock attribute. For top and bottom docked toolbars size relates to height and width will be fixed to width of the signing window. For left and right docked toolbars, the size relates to width and the height of the toolbar will be fixed to the signing window height. The default dock of the toolbar is top.

For desktops with Topaz signature pads connected, minimum size is 30 while the maximum size depends on the monitor size. The table below shows the minimum and maximum sizes for different GemView devices.

GEMVIEW TYPE	MINIMUM SIZE	MAXIMUM SIZE	GEMVIEW MODE
7 inch	30	112	Landscape
	30	196	Portrait
10 inch	30	152	Landscape
	30	248	Portrait
16 inch	30	145	Landscape
	30	265	Portrait

Table 30 GemView signing toolbar size restrictions

int GetSigningWindowToolbarSize()

Method Description

Gets the size of the signing window toolbar.

Parameters

None.

Return Value

The size of the signing window toolbar in pixels.

Remarks

Use it in conjunction with the GetSigningWindowToolbarDock method. If the docking location is 1 (Top) or 2 (Bottom), the size signifies height and if the docking location is 3 (Left) or 4 (Right), the size signifies width.

void SetSigningWindowToolbarPadding(int left, int top, int right, int bottom)

Method Description

Sets the padding of the signing window toolbar.

Parameters

left – Left padding in pixels. Defaults to 0 pixels if not set.
top – Top padding in pixels. Defaults to 0 pixels if not set.
right – Right padding in pixels. Defaults to 0 pixels if not set.
bottom – Bottom padding in pixels. Defaults to 0 pixels if not set.

Return Value

None.

int GetSigningWindowToolbarLeftPadding()

Method Description

Gets the left padding of the signing window toolbar.

Parameters

None.

Return Value

The left padding of the signing window toolbar.

int GetSigningWindowToolbarTopPadding()

Method Description

Gets the top padding of the signing window toolbar.

Parameters

None.

Return Value

The top padding of the signing window toolbar.

int GetSigningWindowToolbarRightPadding()

Method Description

Gets the right padding of the signing window toolbar.

Parameters

None.

Return Value

The right padding of the signing window toolbar.

int GetSigningWindowToolbarBottomPadding()

Method Description

Gets the bottom padding of the signing window toolbar.

Parameters

None.

Return Value

The bottom padding of the signing window toolbar.

void SetSigningWindowToolbarDock(int dockLocation)

Method Description

Sets the docking location of the signing window toolbar.

Parameters

dockLocation - 1 = Top; 2 = Bottom; 3 = Left; 4 = Right. Defaults to 1 (Top) if not set.

Return Value

None.

int GetSigningWindowToolbarDock()

Method Description

Gets the docking location of the signing window toolbar.

Parameters

None.

Return Value

The docking location of the signing window toolbar. 1 = Top; 2 = Bottom; 3 = Left; 4 = Right.

void SetSigningWindowToolbarIconBackColor(string colorCode, int iconType)

Method Description

Sets the background color of the signing window toolbar icon.

Parameters

colorCode - The HTML color code as hexadecimal string e.g. #5882FA". Defaults to "#4B5320" if not set.
iconType - 0 = All; 1 = OK; 2 = Cancel; 3 = Clear.

Return Value

None.

string GetSigningWindowToolbarIconBackColor(int iconType)

Method Description

Gets the background color of the signing window toolbar icon.

Parameters

iconType - 1 = OK; 2 = Cancel; 3 = Clear.

Return Value

The HTML color code as hexadecimal string. E.g. "#F0F0F0";

void SetSigningWindowToolbarIconImage(string imageData, int iconType)

Method Description

Sets the image of the signing window toolbar icon.

Parameters

imageData - imageData as Base64 encoded icon image data. The image should be in SVG format.
iconType – 1 = OK; 2 = Cancel; 3 = Clear.

Return Value

None.

String GetSigningWindowToolbarIconImage(int iconType)

Method Description

Gets the custom image of the signing window toolbar icon.

Parameters

iconType - 1 = OK; 2 = Cancel; 3 = Clear.

Return Value

Base64 encoded icon image data.

void SetSigningWindowToolbarIconSize(int size)

Method Description

size - Sets the size of the signing window toolbar icon.

Parameters

Size is calculated in pixels. The value is applied to both height and width to maintain the aspect ratio. Defaults to 24 pixels if not set.

Return Value

None**.**

Remarks

The signing toolbar icons will have restrictions on minimum and maximum values for size. For desktops with Topaz signature pads connected, minimum size is 24 while the maximum size depends on the monitor size.

The table below shows the minimum and maximum sizes for different GemView devices.

GEMVIEW TYPE	MINIMUM SIZE	MAXIMUM SIZE	GEMVIEW MODE
7 inch	24	112	Landscape
	24	196	Portrait
10 inch	24	152	Landscape
	24	248	Portrait
16 inch	24	145	Landscape
	24	265	Portrait

Table 31 GemView toolbar icon size restrictions

int GetSigningWindowToolbarIconSize()

Method Description

Gets the size of the signing window toolbar icon.

Parameters

None.

Return Value

The size of the signing window toolbar icon in pixels.

void SetSigningWindowToolbarIconMargin(int left, int top, int right, int bottom)

Method Description

Sets the margins of signing window toolbar icons.

Parameters

left – Left padding in pixels. Defaults to 0 pixels if not set.
top – Top padding in pixels. Defaults to 1 pixels if not set.
right – Right padding in pixels. Defaults to 5 pixels if not set.
bottom – Bottom padding in pixels. Defaults to 2 pixels if not set.

Return Value

None.

int GetSigningWindowToolbarIconLeftMargin()

Method Description

Gets the left margin of the signing window toolbar icons.

Parameters

None.

Return Value

The left margin of the signing window toolbar icons.

int GetSigningWindowToolbarIconTopMargin()

Method Description

Gets the top margin of the signing window toolbar icons.

Parameters

None.

Return Value

The top margin of the signing window toolbar icons.

int GetSigningWindowToolbarIconRightMargin()

Method Description

Gets the right margin of the signing window toolbar icons.

Parameters

None.

Return Value

The right margin of the signing window toolbar icons.

int GetSigningWindowToolbarIconBottomMargin()

Method Description

Gets the bottom margin of the signing window toolbar icons.

Parameters

None.

Return Value

The bottom margin of the signing window toolbar icons.

void SetSigningAreaBackColor(string colorCode)

Method Description

Sets the background color of the signing area.

Parameters

colorCode - HTML color code as hexadecimal string e.g. #5882FA". Defaults to "#FFFFFF" if not set.

Return Value

None.

string GetSigningAreaBackColor()

Method Description

Gets the background color of the signing area.

Parameters

None.

Return Value

The HTML color code as hexadecimal string. E.g. "#FFFFFF ".

void SetSigningAreaSize(int width, int height)

Method Description

Sets the size of the signing area.

Parameters

width - Width in pixels. Defaults to 785.

height – height in pixels. Defaults to 240.

Note: Size is defined by width and height attributes. When the "dock" attribute is set to "None", both width and height are applied. For left and right docked signing area, only width is applied and for top and bottom docked signing area, only height is applied. For more information, see The SetSigningAreaDock method.

Return Value

None.

Remarks

The signing area has restrictions on minimum and maximum values for width and height. For Topaz signature pads, minimum width and height is 785 and 240 respectively while for GemView maximum width and height depends on the GemView size. The table below shows the minimum and maximum values for different GemView devices.

GEMVIEW TYPE	MINIMUM SIZE		MAXIMUM SIZE		GEMVIEW MODE
	WIDTH	HEIGHT	WIDTH	HEIGHT	
7 inch	785	240	1024	448	Landscape
	600	240	600	787	Portrait
10 inch	785	240	1280	608	Landscape
	785	240	800	992	Portrait
16 inch	785	340	1366	582	Landscape
	768	340	768	1060	Portrait

Table 32 GemView signing area size restrictions.

int GetSigningAreaWidth()

Method Description

Gets the width of the signing area in pixels.

Parameters

None.

Return Value

The width of the signing area.

int GetSigningAreaHeight()

Method Description

Gets the height of the signing area in pixels.

Parameters

None.

Return Value

The height of the signing area.

void SetSigningAreaLocation(int x, int y)

Method Description

Sets the location of the signing area.

Parameters

x - The x coordinate of the signing area location. Defaults to 0 if not set.
y - The y coordinate of the signing area location. Defaults to 0 if not set.

Return Value

None.

Remarks

Location can only be defined when the "dock" is "None". For more information, see The SetSigningAreaDock method.

int GetSigningAreaLocationX()

Method Description

Gets the x coordinate of the signing area location.

Parameters

None.

Return Value

The x coordinate of the signing area location.

int GetSigningAreaLocationY()

Method Description

Gets the y coordinate of the signing area location.

Parameters

None.

Return Value

The y coordinate of the signing area location.

void SetSigningAreaDock(int dockLocation)

Method Description

Sets the docking location of the signing window toolbar.

Parameters

dockLocation - 0 = None; 1 = Top; 2 is Bottom; 3 = Left; 4 = Right; 5 = Fill. Defaults to 0 (None) with X and Y attributes as 0 if not set.

Return Value

None.

int GetSigningAreaDock()

Method Description

Gets the docking location of the signing area.

Parameters

None.

Return Value

The docking location of the signing area. 0 = None; 1 = Top; 2 = Bottom; 3 = Left; 4 = Right; 5 = Fill.

int Save()

Method Description

Saves and persists the custom window attributes.

Parameters

None.

Return Value

1 = Custom window attributes were saved successfully. Negative value signifies an error. -1 = Error occurred while saving the custom window properties.

Remarks

Once all the custom window properties have been set using the methods defined, the Save() method should be called to persist these properties.

void Reset()

Method Description

Resets the custom window back to its default attributes.

Parameters

None.

Return Value

None.