

# Device ActiveX Interface Document

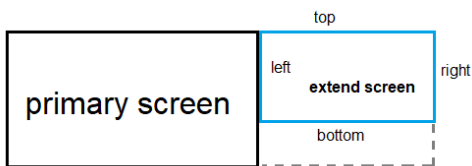
Version 2.x

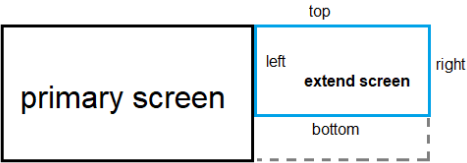
SignViewer ActiveX	
Methods	Description
<b>SetPenSizeRange</b>	<p><b>[Syntax]</b> void SetPenSizeRange(LONG nMin, LONG nMax)</p> <p><b>[Remark]</b> Set range of the pen tip width in pixel. If min=max, tip width is fixed. Default value: nMin=1, nMax=4.</p> <p><b>[Parameter]</b> Min: minimum tip width Max: maximum tip width</p>
<b>GetPenSizeRange</b>	<p><b>[Syntax]</b> void GetPenSizeRange(LONG* lpMin, LONG* lpMax)</p> <p><b>[Remark]</b> Get range of the pen tip width.</p> <p><b>[Parameter]</b> min: returned minimum tip width max: returned maximum tip width</p>
<b>SetPenColor</b>	<p><b>[Syntax]</b> void SetPenColor(BYTE nRed, BYTE nGreen, BYTE nBlue)</p> <p><b>[Remark]</b> Set pen color. nRed, nGreen, and nBlue correspond to the R, G, and B values of the specified RGB. Range of value: 0-255.</p> <p><b>[Parameter]</b> nRed: R value of the specified RGB nGreen: G value of the specified RGB nBlue: B value of the specified RGB</p>
<b>GetPenColor</b>	<p><b>[Syntax]</b> void GetPenColor(BYTE* lpRed, BYTE* lpGreen, BYTE* lpBlue)</p> <p><b>[Remark]</b> Get color of the pen currently in use.</p> <p><b>[Parameter]</b> nRed: R value of the current RGB nGreen: G value of the current RGB</p>

	nBlue: B value of the current RGB
<b>SetBKColor</b>	<p><b>[Syntax]</b> void SetBKColor(BYTE nRed, BYTE nGreen, BYTE nBlue)</p> <p><b>[Remark]</b> Set ActiviteX background color. nRed, nGreen, and nBlue correspond to the R, G, and B values of the specified RGB. Range of value: 0-255.</p> <p><b>[Parameter]</b> nRed: R value of the specified RGB nGreen: G value of the specified RGB nBlue: B value of the specified RGB</p>
<b>GetBKColor</b>	<p><b>[Syntax]</b> void GetBKColor(BYTE* lpRed, BYTE* lpGreen, BYTE* lpBlue)</p> <p><b>[Remark]</b> Get ActiviteX background color.</p> <p><b>[Parameter]</b> nRed: R value of the current RGB nGreen: G value of the current RGB nBlue: B value of the current RGB</p>
<b>SetBorderColor</b>	<p><b>[Syntax]</b> void SetBorderColor(BYTE nRed, BYTE nGreen, BYTE nBlue)</p> <p><b>[Remark]</b> Set ActiviteX border color. nRed, nGreen, and nBlue correspond to the R, G, and B values of the specified RGB. Range of value: 0-255.</p> <p><b>[Parameter]</b> nRed: R value of the specified RGB nGreen: G value of the specified RGB nBlue: B value of the specified RGB</p>
<b>GetBorderColor</b>	<p><b>[Syntax]</b> void GetBorderColor(BYTE* lpRed, BYTE* lpGreen, BYTE* lpBlue)</p> <p><b>[Remark]</b> Get ActiviteX border color.</p> <p><b>[Parameter]</b> nRed: R value of the current RGB nGreen: G value of the current RGB nBlue: B value of the current RGB</p>

<b>IsConnected</b>	<p><b>[Syntax]</b>  VARIANT_BOOL IsConnected(void)</p> <p><b>[Remark]</b>  Device connection status, is the device connected?</p> <p><b>[Parameter]</b>  None</p>
<b>ClearSign</b>	<p><b>[Syntax]</b>  void ClearSign(void)</p> <p><b>[Remark]</b>  Clear handwriting or signature and update background color.</p> <p><b>[Parameter]</b>  None</p>
<b>SaveImageToFile</b>	<p><b>[Syntax]</b>  VARIANT_BOOL SaveImageToFile(BSTR lpstrFileName, SHORT nFormat)</p> <p><b>[Remark]</b>  Save signature as image to the specified location.</p> <p><b>[Parameter]</b>  lpstrFileName: fully qualified file name where the signature will be saved to  nFormat: specifies the image format (0-jpg, 1-png, 2-gif, 3-bmp). Default is 0</p>
<b>SaveImageToFileEx</b>	<p><b>[Syntax]</b>  VARIANT_BOOL SaveImageToFileEx(BSTR lpstrFileName, SHORT nFormat, USHORT width, USHORT height, USHORT quality)</p> <p><b>[Remark]</b>  Save signature as image to the specified location, with image width/height/quality info.</p> <p><b>[Parameter]</b>  lpstrFileName: fully qualified file name where the signature will be saved to  nFormat: specifies the image format (0-jpg, 1-png, 2-gif, 3-bmp). Default is 0  width: image width  height: image height  quality: image coding/packing quality, applicable when format is 0(jpg). Value range: 0-100, default is 100.</p>

<b>SaveSignToFile</b>	<p><b>[Syntax]</b>  VARIANT_BOOL SaveSignToFile(BSTR lpstrFileName, USHORT width, USHORT height)</p> <p><b>[Remark]</b>  Save handwriting as png format image to local file, and the background is transparent (clear). If the specified width or height does not equal to that of the original image, the original image will be resized according to the specified width/height.</p> <p><b>[Parameter]</b>  lpstrFileName: fully qualified file name where the handwriting will be saved to  width: image width  height: image height</p>
<b>SavelImageToBase64</b>	<p><b>[Syntax]</b>  BSTR SavelImageToBase64(SHORT nFormat, USHORT width, USHORT height, USHORT quality)</p> <p><b>[Remark]</b>  Save image as Base64 string. Typically used in Web application for URL data submission and storing.</p> <p><b>[Parameter]</b>  nFormat: specifies the image format (0-jpg, 1-png, 2-gif, 3-bmp). Default is 0  width: image width  height: image height  quality: image coding/packing quality, applicable when format is 0(jpg). Value range: 0-100, default is 100.</p>
<b>SaveSignToBase64</b>	<p><b>[Syntax]</b>  BSTR SaveSignToBase64(USHORT width, USHORT height)</p> <p><b>[Remark]</b>  Save signature/handwriting image as Base64 string. Typically used in Web application for URL data submission and storing. Image format is png, and the background is transparent (clear).</p> <p><b>[Parameter]</b>  width: image width  height: image height</p>
<b>SetPenMode</b>	<p><b>[Syntax]</b>  void SetPenMode(LONG nMode)</p>

	<p><b>[Remark]</b> Set pen working mode.</p> <p><b>[Parameter]</b> nMode: 0 for writing mode, 1 for eraser mode</p>
<b>RegBtnClickEvent</b>	<p><b>[Syntax]</b> LONG RegBtnClickEvent(ULONG nBtnIndex, LONG x, LONG y, LONG w, LONG h)</p> <p><b>[Remark]</b> Register a pen tapping event over a specified area (excluding the ActiveX area). When the pen is tapped within this specified area, the registered event will trigger. If device has screen, x/y coordinate is relative to the upper left corner of the extended screen. See picture below.</p>  <p><b>[Parameter]</b> nBtnIndex: id of the specified area/event x: x coordinate of the registered area y: y coordinate of the registered area w: width of the registered area h: height of the registered area</p>
<b>UnregBtnClickEvent</b>	<p><b>[Syntax]</b> void UnregBtnClickEvent(LONG nBtnIndex)</p> <p><b>[Remark]</b> Unregister pen tapping event</p> <p><b>[Parameter]</b> nBtnIndex: previously registered event id</p>
<b>GetScreenRect</b>	<p><b>[Syntax]</b> VARIANT_BOOL GetScreenRect(LONG* left, LONG* top, LONG* right, LONG* bottom)</p> <p><b>[Remark]</b> Get display area size. Typically the display is configured as "Extended Screen", as shown in the picture below. This function is mainly used to calculate the popup window position on the extended screen.</p>

	 <p><b>[Parameter]</b></p> <p>left: absolute coordinate of the left side of the extended screen relative to the primary screen</p> <p>top: absolute coordinate of the top side of the extended screen relative to the primary screen</p> <p>right: absolute coordinate of the right side of the extended screen relative to the primary screen</p> <p>bottom: absolute coordinate of the bottom side of the extended screen relative to the primary screen</p>
<b>GetScreenX</b>	<p><b>[Syntax]</b></p> <p>LONG GetScreenX()</p> <p><b>[Remark]</b></p> <p>Get the absolute coordinate (in pixel) of the left side of the extended screen relative to the primary screen. This function is mainly called by JavaScript in Web application.</p> <p><b>[Parameter]</b></p> <p>None</p>
<b>GetScreenY</b>	<p><b>[Syntax]</b></p> <p>LONG GetScreenY()</p> <p><b>[Remark]</b></p> <p>Get the absolute coordinate (in pixel) of the top side of the extended screen relative to the primary screen. This function is mainly called by JavaScript in Web application.</p> <p><b>[Parameter]</b></p> <p>None</p>
<b>GetScreenWidth</b>	<p><b>[Syntax]</b></p> <p>LONG GetScreenWidth()</p> <p><b>[Remark]</b></p> <p>Get the screen width in pixel. This function is mainly called by JavaScript in Web application.</p> <p><b>[Parameter]</b></p> <p>None</p>
<b>GetScreenHeight</b>	<p><b>[Syntax]</b></p> <p>LONG GetScreenHeight()</p> <p><b>[Remark]</b></p> <p>Get the screen height in pixel. This function is mainly</p>

	<p>called by JavaScript in Web application.</p> <p><b>[Parameter]</b></p> <p>None</p>
<b>SetScreenRotation</b>	<p><b>[Syntax]</b></p> <p>void SetScreenRotation(LONG angle)</p> <p><b>[Remark]</b></p> <p>Rotate the screen clock-wise.</p> <p><b>[Parameter]</b></p> <p>angle: degree of angle to rotate. Supported values are: 0, 90, 180, and 270. 0 is the default. If the specified angle is not one of the above values, it will be rounded to the closest one.</p>
<b>MouseControl</b>	<p><b>[Syntax]</b></p> <p>void MouseControl(VARIANT_BOOL bControlled)</p> <p><b>[Remark]</b></p> <p>Whether to control system mouse. If the device has screen and is configured to be "Extended Screen", use "ExtendDisplay" attribute to control the mouse in Extended mode.</p> <p><b>[Parameter]</b></p> <p>bControlled: whether to control system mouse</p>
<b>EnableSign</b>	<p><b>[Syntax]</b></p> <p>void EnableSign(VARIANT_BOOL bEnable)</p> <p><b>[Remark]</b></p> <p>Whether to allow signing in ActiveX.</p> <p><b>[Parameter]</b></p> <p>bEnable: "true" to allow signing, "false" to disable</p>
<b>IsOperated</b>	<p><b>[Syntax]</b></p> <p>VARIANT_BOOL IsOperated()</p> <p><b>[Remark]</b></p> <p>Check whether there is signing operation. Return "true" if there is any writing action detected, "false" otherwise. Function also return "false" after ClearSign is just called.</p> <p><b>[Parameter]</b></p> <p>None</p>
<b>Attributes</b>	<b>Description</b>
<b>BorderVisible</b>	<p><b>[Syntax]</b></p> <p>VARIANT_BOOL BorderVisible</p> <p><b>[Remark]</b></p>

	<p>Set whether ActiveX border is visible.</p> <p><b>[Value]</b> true/false</p> <p><b>[Privilege]</b> [read, write]</p>
<b>ExtendDisplay</b>	<p><b>[Syntax]</b> VARIANT_BOOL ExtendDisplay</p> <p><b>[Remark]</b> If the device has screen and is calling "MouseControl" function, this attribute controls whether the mouse is working in Extended mode.</p> <p><b>[Value]</b> true/false</p> <p><b>[Privilege]</b> [read, write]</p>
<b>DisplayMapMode</b>	<p><b>[Syntax]</b> LONG DisplayMapMode</p> <p><b>[Remark]</b> Set mapping method between the device screen and the primary screen.</p> <p><b>[Value]</b> 0-map to primary screen, 1-map to ActiveX</p> <p><b>[Privilege]</b> [read, write]</p>
<b>SignMode</b>	<p><b>[Syntax]</b> LONG SignMode</p> <p><b>[Remark]</b> Set signing method. In mixed mode, the pen has priority over touch.</p> <p><b>[Value]</b> 0-Use pen    1-Finger touch    2-Mixed</p> <p><b>[Privilege]</b> [read, write]</p>
<b>CursorVisible</b>	<p><b>[Syntax]</b> VARIANT_BOOL CursorVisible</p> <p><b>[Remark]</b> In the mode where writing can be controlled by mouse, this attribute controls whether the mouse cursor is visible or not. Default is "visible".</p> <p><b>[Value]</b></p>



	<p>true/false</p> <p><b>[Privilege]</b></p> <p>[read, write]</p>
<b>SignDrawingQuality</b>	<p><b>[Syntax]</b></p> <p>LONG SignDrawingQuality</p> <p><b>[Remark]</b></p> <p>Set signature writing quality. The higher quality, the more resource it will take and the smoother the handwriting appears. Use this attribute wisely according to hardware spec. Default value is 1 (normality quality).</p> <p><b>[Value]</b></p> <p>0- Do not use anti-aliasing or smoother feature 1- Normal quality 2- Better quality</p> <p><b>[Privilege]</b></p> <p>[read, write]</p>
Events	Description
<b>DevNotifyEvent</b>	<p><b>[Syntax]</b></p> <p>void DevNotifyEvent(LONG status)</p> <p><b>[Remark]</b></p> <p>Event notification when device connects or disconnects.</p> <p><b>[Parameter]</b></p> <p>status=0 for device just disconnected, status=1 for device just connected</p>
<b>BtnClickEvent</b>	<p><b>[Syntax]</b></p> <p>void BtnClickEvent(LONG nBtnIndex)</p> <p><b>[Remark]</b></p> <p>Event notification when pen taps within the registered non-ActiveX area. Related methods: RegBtnClickEvent, nregBtnClickEvent.</p> <p><b>[Parameter]</b></p> <p>nBtnIndex: registered area/event id</p>
<b>TouchEvent</b>	<p><b>[Syntax]</b></p> <p>void TouchEvent(LONG type, LONG data)</p> <p><b>[Remark]</b></p> <p>If device has touch feature, event will be triggered if there is a touch activity.</p> <p><b>[Parameter]</b></p> <p>type: finger motion status</p>

	<ul style="list-style-type: none"> <li>0- Finger lift</li> <li>1- Finger pressed</li> <li>2- Finger pressed and moved</li> </ul> <p>data: coordinate of the touch point. The lower 16 bits is screen x coordinate and the higher 16 bits is screen y coordinate. To get the actual x/y coordinate, do: <math>x = \text{data} \&amp; 0xFFFF</math>, <math>y = \text{data} \gg 16</math>.</p>
<b>KeyEvent</b>	<p><b>[Syntax]</b> void KeyEvent(ULONG PKeyCode, ULONG VKeyCode)</p> <p><b>[Remark]</b> Event will be triggered if a key is pressed.</p> <p><b>[Parameter]</b> PkeyCode: The 0-31 bit mask indicates whether the key is pressed or not. E.g.: when key "1" is pressed, PkeyCode=0x01; when key "2" is pressed, PkeyCode=0x02; when both key "1" and "2" are pressed, PkeyCode=0x03. VkeyCode: The virtual key number on the device, starting from 1, represents the status of only one key.</p>
<b>PenEvent</b>	<p><b>[Syntax]</b> void PenEvent(LONG type, LONG data)</p> <p><b>[Remark]</b> If device has pen feature, event will be triggered if there is a pen activity.</p> <p><b>[Parameter]</b> type: pen motion status</p> <ul style="list-style-type: none"> <li>0- pen hover,</li> <li>1- pen down,</li> <li>2- pen move,</li> <li>3- pen up,</li> <li>4- pen leave</li> </ul> <p>data: coordinate of the pen point. The lower 16 bits is screen x coordinate and the higher 16 bits is screen y coordinate. To get the actual x/y coordinate, do: <math>x = \text{data} \&amp; 0xFFFF</math>, <math>y = \text{data} \gg 16</math>.</p>