Software Developer Guide

SigPlusNET

Version 2.0.0.42

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Topaz Namespace

SigPlusNet

Provides all the functionality required for customized capture and encryption of electronic handwritten biometric signatures.

SigPlusNET Class

Public class SigPlusNet

Remarks: Provides all the functionality required for customized capture and encryption of electronic handwritten biometric signatures.

Thread Safety: Public static (Shared in Visual Basic) members of this type are safe from multithreaded operations. Instance members are not guaranteed to be thread-safe.

Namespace: Topaz

Assembly: SigPlusNET (in SigPlusNET.dll)

SigPlusNET Members

AccessibilityObject (inherited from Control)

Gets the AccessibleObject assigned to the control.

AccessibleDefaultActionDescription (inherited from Control)

Gets or sets the default action description of the control for use by accessibility client applications.

AccessibleDescription (inherited from Control)

Gets or sets the description of the control used by accessibility client applications.

AccessibleName (inherited from Control)

Gets or sets the name of the control used by accessibility client applications.

AccessibleRole (inherited from Control)

Gets or sets the accessible role of the control.

AccessibilityNotifyClients

Notifies the accessibility client application of the specified AccessibleEvents for the specified child control.

AllowDrop (inherited from Control)

Gets or sets a value indicating whether the control can accept data that the user drags onto it.

Anchor (inherited from Control)

Gets or sets which edges of the control are anchored to the edges of its container.

AutoKeyFinish

public void AutoKeyFinish () ;

Remarks: Completes the auto key generation function.

AutoKeyStart

public void AutoKeyStart () ;

Remarks: Initializes the automatic key generation function which derives a key from the data fed to it via SetAutoKeyData (string), when all data is input then AutoKeyFinish () must be called to complete key generation. If the AutoKeyStart method is not called, then the SetAutoKeyData method is used to pass the

path to the file, which is then used to encrypt the signature. When AutoKeyStart is called then SetAutoKeyData is used to pass in string literals as data.

BackColor (inherited from Control)

Gets or sets the background color for the control. SetUseAmbientColors must be enabled first.

BackColorChanged

Occurs when the value of the BackColor property changes.

BackgroundImage (inherited from Control)

Gets or sets the background image displayed in the control.

BackgroundImageChanged

Occurs when the value of the BackgroundImage property changes.

BeginInvoke

Overloaded. Executes the specified delegate asynchronously with the specified arguments, on the thread that the control's underlying handle was created on.

BindingContext (inherited from Control)

Gets or sets the BindingContext for the control.

BindingContextChanged

Occurs when the value of the BindingContext property changes.

Bottom (inherited from Control)

Gets the distance between the bottom edge of the control and the top edge of its container's client area.

Bounds (inherited from Control)

Gets or sets the size and location of the control including its nonclient elements.

BringToFront

Brings the control to the front of the z-order.

CanFocus (inherited from Control)

Gets a value indicating whether the control can receive focus.

CanSelect (inherited from Control)

Gets a value indicating whether the control can be selected.

Capture (inherited from Control)

Gets or sets a value indicating whether the control has captured the mouse.

CausesValidation (inherited from Control)

Gets or sets a value indicating whether the control causes validation to be performed on any controls that require validation when it receives focus.

CausesValidationChanged

Occurs when the value of the CausesValidation property changes.

ChangeUICues

Occurs when the focus or keyboard user interface (UI) cues change.

ClearSigWindow

```
public void ClearSigWindow(
  short Inside
);
```

Remarks: Erases data either inside or outside of sig window based on value of short inside.

Parameters: inside-if=0 then signature data is erased (in window), if =1 then data outside sig window is erased.

ClearTablet

public void ClearTablet();

Remarks: Clears the signature object of ink.

Click

Occurs when the control is clicked.

ClientRectangle (inherited from Control)

Gets the rectangle that represents the client area of the control.

ClientSize (inherited from Control)

Gets or sets the height and width of the client area of the control.

CompanyName (inherited from Control)

Gets the name of the company or creator of the application containing the control.

Container (inherited from Control)

Gets the Container that contains the Component

Contains

Retrieves a value indicating whether the specified control is a child of the control

ContainsFocus (inherited from Control)

Gets a value indicating whether the control, or one of its child controls, currently has the input focus.

ContextMenu (inherited from Control)

Gets or sets the shortcut menu associated with the control.

ContextMenuChanged

Occurs when the value of the ContextMenu property changes.

ControlAdded

Occurs when a new control is added to the ControlCollection.

ControlRemoved

Occurs when a control is removed from the ControlCollection.

Controls (inherited from Control)

Gets the collection of controls contained within the control.

CreateAccessibilityInstance

Creates a new accessibility object for the control.

CreateControl

Forces the creation of the control, including the creation of the handle and any child controls.

CreateControlsInstance

Creates a new instance of the control collection for the control.

Created (inherited from Control)

Gets a value indicating whether the control has been created.

CreateGraphics

Creates the Graphics object for the control.

CreateHandle

Creates a handle for the control.

CreateObjRef

Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.

CreateParams
Gets the required creation parameters when the control handle is created.

Cursor (inherited from Control)
Gets or sets the cursor that is displayed when the mouse pointer is over the control.

CursorChanged
Occurs when the value of the Cursor property changes.

DataBindings (inherited from Control)
Gets the data bindings for the control

DefaultImeMode
Gets the default Input Method Editor (IME) mode supported by the control.

DefaultSize
Gets the default size of the control.

DefWndProc
Sends the specified message to the default window procedure.

DesignMode
Gets a value that indicates whether the Component is currently in design mode.

DestroyHandle
Destroys the handle associated with the control.

DisplayRectangle (inherited from Control)
Gets the rectangle that represents the display area of the control.

Dispose
Overloaded. Releases the unmanaged resources used by the Control and optionally releases the managed resources.

Disposed
Adds an event handler to listen to the Disposed event on the component.

Disposing (inherited from Control)
Gets a value indicating whether the control is in the process of being disposed of.

Dock (inherited from Control)
Gets or sets which edge of the parent container a control is docked to.

DockChanged
Occurs when the value of the Dock property changes.

DoDragDrop
Begins a drag-and-drop operation.

DoubleClick
Occurs when the control is double-clicked.

DragDrop
Occurs when a drag-and-drop operation is completed.

DragEnter
Occurs when an object is dragged into the control's bounds.

DragLeave
Occurs when an object is dragged out of the control's bounds.

DragOver

Occurs when an object is dragged over the control's bounds.

Enabled (inherited from Control)

Gets or sets a value indicating whether the control can respond to user interaction.

EnabledChanged

Occurs when the Enabled property values has changed.

EndInvoke

Retrieves the return value of the asynchronous operation represented by the IAsyncResult object passed.

Enter

Occurs when the control is entered.

Equals

Determines whether the specified Object is equal to the current Object.

Events

Gets the list of event handlers that are attached to this Component.

ExportSigFile

```
public bool ExportSigFile(
  string FileName
);
```

Remarks: Writes out a signature file in the Topaz image-free raw tablet data vector file format (.sig extension).

Parameters: FileName – Name of file

Return Value: True if successful, false if not successful

Finalize

```
protected override void Finalize();
```

FindForm

Retrieves the form that the control is on.

Focus

Sets input focus to the control.

Focused (inherited from Control)

Gets a value indicating whether the control has input focus.

Font (inherited from Control)

Gets or sets the font of the text displayed by the control.

FontChanged

Occurs when the Font property value changes.

FontHeight

Gets or sets the height of the font of the control

ForeColor (inherited from Control)

Gets or sets the foreground color of the control. SetUseAmbientColors must be enabled first.

ForeColorChanged

Occurs when the ForeColor property value changes.

GetAnnotate

public string GetAnnotate();

Return Value: Returns current ASCII Annotation string

GetChildAtPoint

Retrieves the childcontrol that is located at the specified coordinates.

GetContainerControl

Returns the next ContainerControl up the control's chain of parent controls.

GetDisplayAnnotate

public bool GetDisplayAnnotate();

Remarks: Gets the current setting to display Annotation

Return Value: True if Annotation is displayed, false if not displayed

GetDisplayAnnotatePosX

public int GetDisplayAnnotatePosX();

Remarks: Gets the current X position for the start of the Annotation String in the signature box.

Return Value: X position for start of the Annotation String, if 0 then text is positioned 5% in from right side of signature box.

GetDisplayAnnotatePosY

public int GetDisplayAnnotatePosY();

Remarks: Gets the current Y position for the start of the Annotation String in the signature box.

Return Value: Y position for start of the Annotation String, if 0 then text is positioned 5% in from bottom edge of signature box.

GetDisplayAnnotateSize

public int GetDisplayAnnotateSize();

Remarks: Gets current Y size in pixels of the Annotation start in the signature box.

Return Value: Text size of the Annotation in pixels, if 0 then the text size is 7.5% of the Y size of the signature box.

GetDisplayMode

public int GetDisplayMode();

Remark: *NOT CURRENTLY IMPLEMENTED*

GetDisplayPenWidth

public int GetDisplayPenWidth();

Return Value: Current pen ink width for the displayed signature in pixels.

GetDisplayRotate

public bool GetDisplayRotate();

Return Value: Value of rotation on a 360 degree axis for signature display.

GetDisplayRotateSave

public bool GetDisplayRotateSave();

Remarks: Gets state of DisplayRotateSave.Can be used to rotate and then save in rotated format, signatures after capture, if the signature was accidentally taken in a rotated orientation during signature capture. Normally, so change the tablet orientation during capture, only the TabletRotation property is used.

Return Value: TRUE DisplayRotateSave mode = active, FALSE DisplayRotateSave mode = inactive.

GetDisplayTimeStamp

public bool GetDisplayTimeStamp();

Remarks: Gets the current setting for bean to display Time Stamp.

Return Value: True if Time Stamp is displayed, False if not displayed.

GetDisplayTimeStampPosX

public int GetDisplayTimeStampPosX();

Remarks: Gets the current X Position for the start of the Time Stamp String in the signature box.

Return Value: X value in pixels relative to the left edge, if 0 then 5% in from left side of signature box.

GetDisplayTimeStampPosY

public int GetDisplayTimeStampPosY();

Remarks: Gets the current Y Position for the start of the Time Stamp String in the signature box.

Return Value: Y value in pixels relative to the bottom edge, if 0 then 5% in from bottom edge of signature box.

GetDisplayTimeStampSize

public int GetDisplayTimeStampSize();

Remarks: Gets the Y size in pixels of the Time Stamp in the signature box.

Return Value: Time Stamp size in pixels, if 0 then text size is 7.5% of Y size of signature box.

GetDisplayWindowRes

public bool GetDisplayWindowRes();

Remarks: Gets state of DisplayWindowRes.

Return Value: TRUE DisplayWindowRes mod = active, FALSE DisplayWindowRes mode = inactive.

GetEncryptionMode

public int GetEncryptionMode();

Remarks: Returns current EncryptionMode.

Return Value: Numeric value of encryption mode, 0 = no encryption, 1 = medium encryption, 2 = higher security encryption mode.

GetHashCode

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

GetImageAnnotate

public bool GetImageAnnotate();

Remarks: Gets the current setting for bean to display Annotation as it applies to the WriteImageFile method.

Return Value: True if Annotation is displayed, false if not displayed for Image.

GetImageAnnotatePosX

public int GetImageAnnotatePosX();

Remarks: Gets the current X position for the start of the Annotation String in the signature box as it applies to the WriteImageFile method.

Return Value: X position for start of the Annotation string, if 0 then text is positioned 5% in from right side of signature box for Image.

GetImageAnnotatePosY

public int GetImageAnnotatePosY();

Remarks: Gets the current Y position for the start of the Annotation String in the signature box as it applies to the WriteImageFile method.

Return Value: Y position for start of the Annotation string, if 0 then text is positioned 5% in from bottom edge of signature box for Image.

GetImageAnnotateSize

public int GetImageAnnotateSize();

Remarks: Gets the current Y size in pixels of the Annotation start in the signature box as it applies to the WriteImageFile method.

Return Value: Text size of the Annotation in pixels, if 0 then text size is 7.5% of the Y size of the signature box.

GetImageFileFormat

public int GetImageFileFormat();

Remarks: Returns the current setting of the image file format.

Return Value: Integer value of image file type as listed for the SetImageFileFormat method.

GetImagePenWidth

public int GetImagePenWidth();

Remarks: Gets current pen ink width as it applies to the WriteImageFile method.

Return Value: Pen ink width for Image.

GetImageTimeStamp

public bool GetImageTimeStamp();

Remarks: Gets the current setting for bean to display Time Stamp as applies to the WriteImageFile method.

Return Value: True if Time Stamp is displayed, False if not displayed for Image.

GetImageTimeStampPosX

public int GetImageTimeStampPosX();

Remarks: Gets X position of the start of the Time Stamp in the signature box as it applies to the WriteImageFile method.

Return Value: X value in pixels relative to the left edge, if 0 then 5% in from the left side of signature box.

GetImageTimeStampPosY

public int GetImageTimeStampPosY();

Remarks: Gets the current Y position in pixels for the start of the Time Stamp as it applies to the WriteImageFile method.

Return Value: Y value in pixels relative to the left edge, if 0 then 5% in from the bottom edge of signature box.

GetImageTimeStampSize

public int GetImageTimeStampSize ();

Remarks: Gets the Y size in pixels of the Time Stamp in the signature box as it applies to the WriteImageFile method.

Return Value: Time Stamp size in pixels, if 0 then text size is 7.5% of Y size of signature box.

GetImageXSize

public int GetImageXSize();

Remarks: Gets the current width in X pixels of the image as it applies to the WriteImageFile method.

Return Value: Number of X pixels of image.

GetImageYSize

public int GetImageYSize();

Remarks: Gets the current height in Y pixels of the image as it applies to the WriteImageFile method.

Return Value: Number of Y pixels of image height.

GetJustifyMode

public int GetJustifyMode();

Remarks: Gets the current justification mode – how the signature is sized and positioned in the signature box as well as when using the WriteImageFile method.

Return Value: Justification mode, 0-normal no justification, 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).

GetJustifyX

public int GetJustifyX();

Return Value: Justification X buffer size in pixels for display.

GetJustifyY

public int GetJustifyY();

Return Value: Justification Y buffer size in pixels for display.

GetKeyReceipt

public int GetKeyReceipt();

Remarks: Returns a 32 bit value that is uniquely derived from the key, it can be used to verify that a document has not been modified if the Auto key feature was used to generate the key.

Return Value: 32 bit binary receipt.

GetKeyReceiptAscii

public string GetKeyReceiptAscii();

Remarks: Returns the key receipt as an 8 character Ascii hex string.

Return Value: The Ascii string.

GetKeyString

public string GetKeyString();

Remarks: Provides hash of the encryption data in ASCII compatible format.

Return Value: Hash of encryption data.

GetLCDCaptureMode

public int GetLCDCaptureMode();

Remarks: Gets the current LCD Capture Mode for the tablet.

Return Value: Mode the LCD is set to capture signatures in, Mode 0 no LCD commands are sent to the tablet, Mode 1-sets capture mode to be active with Autoerase in the tablet, Mode 2-sets the tablet to persistent ink capture without autoerase, Mode 3-signature ink is displayed inverted on a suitable dark background set using the Graphic functions.

GetLifetimeService

Retrieves the current lifetime service object that controls the lifetime policy for this instance.

GetNextControl

Retrieves the next control forward or back in the tab order of child controls.

GetSaveSigInfo

public bool GetSaveSigInfo();

Return Value: True if SigInfo is enabled, False if disabled.

GetService

Returns an object that represents a service provided by the Component or by its Container.

GetSigCompressionMode

public int GetSigCompressionMode();

Remarks: Returns compression mode for signatures.

Return Value: Mode for compression of signature, where 0= no compression, 1= lossless compression with compacted data format, 2-8= compression ratio of signature stored in in .sig file where 2=1KB typ, 4=500 byte typ, and 8=250 byte typ. Topaz Systems does not recommend compressing beyond setting 1 unless size is more important than signature quality.

GetSigImage

public Image GetSigImage();

Remarks: Renders and returns a .NET Image, using ImageX and YSizes and Image pen width. The returned image can use the SaveAs method in the Image class to save the image into a number of supported formats, such as jpg, tif, bmp, png, etc.

Return Value: A .NET Image.

GetSigReceipt

public int GetSigReceipt ();

Remarks: Returns a 32 bit receipt similar to the key receipt. Forms receipt by using the auto key generation algorithm on the signature file and the result can be used to verify that the signature has not been modified.

Return Value: 32 bit binary receipt.

GetSigReceiptAscii

public string GetSigReceiptAscii ();

Remarks: Same as GetKeyReceiptAscii, but for Sig receipt.

Return Value: The Ascii string.

GetSigString

public string GetSigString();

Return Value: SigString as ASCII hex string.

GetStyle

Retrieves the value of the specified control style bit for the control.

GetTabletBaudRate

public int GetTabletBaudRate();

Return Value: Current TabletBaudRate.

GetTabletComPort

public int GetTabletComPort ();

Return Value: Current COM port setting.

GetTabletComTest

public int GetTabletComTest ();

Remarks: Gets current hardware check mode, can be used to determine if tablet is connected or which port tablet is connected to.

Return Value: Current hardware check mode, True if active, False if not active.

GetTabletFilterPoints

public int GetTabletFilterPoints ();

Return Value: Current TabletFilterPoints

GetTabletLogicalXSize

public int GetTabletLogicalXSize();

Return Value: Current horizontal values used in representing signatures in Logical Tablet Coordinates.

GetTabletLogicalYSize

public int GetTabletLogicalYSize();

Return Value: Current vertical values used in representing signatures in Logical Tablet Coordinates.

GetTabletResolution

public int GetTabletResolution();

Return Value: Current TabletResolution.

GetTabletRotation

public int GetTabletRotation();

Remarks: Gets the current orientation on a 360 degree axis for display of tablet data. The data in the sig representation is stored in the native tablet orientation.

Return Value: Current tablet orientation.

GetTabletState

public int GetTabletState();

Remarks: Indicates capture state of the tablet.

Return Value: Value of 1 enables the component to access the selected COM or USB port and access the tablet for signature capture, 0 disables the tablet for capture.

GetTabletTimingAdvance

public int GetTabletTimingAdvance();

Return Value: Current TabletTimingAdvance

GetTabletType

public int GetTabletType();

Remarks: Gets TabletType value.

Return Value: Integer value of TabletType. See SetTabletType.

GetTabletXStart

public int GetTabletXStart () ;

Return Value: Current X position in Logical Tablet Coordinates of the upper left hand corner of the component signature box.

GetTabletXStop

public int GetTabletXStop();

Remarks: Gets the X pos in Logical Tablet Coordinates of the right most X pixel.

Return Value: The X pos of the right most pixel.

GetTabletYStart

public int GetTabletYStart();

Remarks: Gets the current Y pos in Logical Tablet Coordinates of the top most X pixel.

Return Value: The Y pos of the top most pixel.

GetTabletYStop

public int GetTabletYStop();

Remarks: Gets the current Y pos in Logical Tablet Coordinates of the bottom most X pixel.

Return Value: The Y pos of the bottom most pixel.

GetTimeStamp

public string GetTimeStamp();

Remarks: Gets the current Time Stamp string for the signature.

Return Value: ASCII new line character.

GetTopLevel

Determines if the control is a top-level control.

GetType

Gets the Type of the current instance.

GiveFeedback

Occurs during a drag operation.

GotFocus

Occurs when the control receives focus.

Handle (inherited from Control)

Gets the window handle that the control is bound to.

HandleCreated

Occurs when a handle is created for the control.

HandleDestroyed

Occurs when the control's handle is in the process of being destroyed.

HasChildren (inherited from Control)

Gets a value indicating whether the control contains one or more child controls.

Height (inherited from Control)

Gets or sets the height of the control.

HelpRequested

Occurs when the user requests help for a control.

Hide

Conceals the control from the user.

ImeMode (inherited from Control)

Gets or sets the Input Method Editor (IME) mode of the control.

ImeModeChanged

Occurs when the ImeMode property has changed.

ImportSigFile

```
public bool ImportSigFile(
  string FileName
);
```

Remarks: Clears the current signature, read in a signature file in the Topaz vector file format, and display it.

Parameters: FileName – Contains the path and filename that is to be read from.

Return Value: True if successful, False if not successful.

InitializeLifetimeService

Obtains a lifetime service object to control the lifetime policy for this instance.

InitLayout

Called after the control has been added to another container.

Invalidate

Overloaded. Invalidates the specified region of the control (adds it to the control's update region, which is the area that will be repainted at the next paint operation), and causes a paint message to be sent to the control.

Invalidated

Occurs when a control’s display requires redrawing.

Invoke

Overloaded. Executes the specified delegate, on the thread that owns the control's underlying window handle, with the specified list of arguments.

InvokeGotFocus

Raises the GotFocus event for the specified control.

InvokeLostFocus

Raises the LostFocus event for the specified control.

InvokeOnClick

Raises the Click event for the specified control.

InvokePaint

Raises the Paint event for the specified control.

InvokePaintBackground

Raises the PaintBackground event for the specified control.

InvokeRequired (inherited from Control)

Gets a value indicating whether the caller must call an invoke method when making method calls to the control because the caller is on a different thread than the one the control was created on.

IsAccessible (inherited from Control)

Gest or sets a value indicating whether the control is visible to accessibility applications.

IsDisposed (inherited from Control)

Gets a value indicating whether the control has been disposed of.

IsHandleCreated (inherited from Control)

Gets a value indicating whether the control has a handle associated with it.

IsInputChar

Determines if a character is an input character that the control recognizes.

IsInputKey

Determines whether the specified key is a regular input key or a special key that requires preprocessing.

KeyDown

Occurs when a key is pressed while the control has focus.

KeyPadAddHotSpot

```
public void KeyPadAddHotSpot(
  short KeyCode,
  short CoordToUse,
  short XPos,
  short YPos,
  short XSize,
  short YSize
);
```

Remarks: Defines in software the location of a tablet HotSpot which is used by the developer to detect user pen taps.

Parameters:
KeyCode-Integer value defining the HotSpot.
CoordToUse-Coordinate system used for this HotSpot.
XPos-Location (upper left- 0,0)
YPos-Same
XSize-X size in pixels.
YSize-Y size in pixels.

KeyPadClearHotSpotList

```
public void KeyPadClearHotSpotList();
```

Remarks: Clears the controls internal list of HotSpots created using KeyPadAddHotSpot.

KeyPadQueryHotSpot

```
public short KeyPadQueryHotSpot(
  short KeyCode
);
```

Remarks: Queries whether the specified HotSpot has been tapped by the user. Returns a true if the control contains data that is within the definition of the keyCode on the tablet.

Parameters: KeyCode-Mapped Logical Tablet Coordinates.

Return Value: Number of points within the KeyCode definition.

KeyPress

Occurs when a key is pressed while the control has focus.

KeyUp

Occurs when a key is released while the control has focus.

Layout

Occurs when a control should reposition its child controls.

LCDClear

```
public void LCDClear();
```

Remarks: Erases the LCD display easily by calling LCDRefresh to do it.

LCDRefresh

```
public bool LCDRefresh(
  int Mode,
  int XPos,
  int YPos,
  int XSize,
  int YSize
);
```

Remarks: Sends tablet a refresh command with 4 possible modes. Mode 0-Clear, display is cleared at the specified location. Mode 1-Complement, complements display at the specified location. Mode 2-WriteOpaque, transfers contents of the background memory to the LCD display, overwriting the content of the LCD display. Mode 3-WriteTransparent, transfers contents of the background memory in the tablet to the LCD display and combined in the contents of the LCD display.

Parameters:
Mode-Defined as above (0-4)
XPos-Location in LCD Coordinates (upper left-0,0)
YPos-Same
XSize-X size in LCD pixels
YSize-Y size in LCD pixels

Return Value: True if checksum received and verified, False if no or incorrect checksum received from tablet.

LCDSendCmdString

```
public int LCDSendCmdString(
  string CmdStr,
  int ReturnCount,
  string Result,
  int TimeOut
);
```

Remarks: Method used to send commands to Topaz LCD Tablets. *NOT CURRENTLY FOR DEVELOPER USE.*

LCDSendGraphic

```
public bool LCDSendGraphic(
  int Dest,
  int Mode,
  int XPos,
  int YPos,
  Bitmap BitmapData
);
```

Remarks: This writes an image to the LCD by taking a .NET Drawing::Bitmap as the source for the image.

Parameters:
Dest- 0=Foreground,1=Background memory in tablet
Mode-0-3 as defined in LCDWriteString
XPos-Location in LCD coordinates (upper left- 0,0)
YPos-Same
BitmapData-Source for rendered image

Return Value: True if successful, False if not.

LCDSetWindow

```
public bool LCDSetWindow(
  int XPos,
  int YPos,
  int XSize,
  int YSize
);
```

Remarks: Sets a signature window that restricts the ink of the SigPlus object to said window on the LCD itself (see SetLCDCaptureMode).

Parameters:
XPos – Location in LCD coordinates (upper left – 0,0)
YPos – Same
XSize – X size in LCD pixels
YSize – Y size in LCD pixels

Return Value: True if checksum received and verified, False if no or incorrect checksum received from tablet.

LCDStringHeight

```
public int LCDStringHeight(
  Font DrawFont,
  string Str
);
```

Remarks: Takes a string and a .NET font and hand back how tall the string is in pixels.

Parameters:
DrawFont - .NET font
Str – String

LCDStringWidth

```
public int LCDStringWidth(
  Font DrawFont,
  string Str
);
```

Remarks: Takes a string and a .NET font and hand back how wide the string is in pixels.

Parameters:
DrawFont - .NET font
Str – String

LCDWriteString

```
public bool LCDWriteString(
  int Dest,
  int Mode,
  int XPos,
  int YPos,
  Font DrawFont,
  string Str
);
```

Remarks: Used to write the image data to the LCD Display. The data is written at the location specified by the combination of Dest, XPos, and YPos. The Mode determines how the data is written.

Mode 0 - Clear: The Display is cleared at the specified location.
Mode 1 - Complement: The Display is complemented at the specified location.

Mode 2 - WriteOpaque: The contents of the background memory in the tablet are transferred to the LCD display, overwriting the contents of the LCD display.
Mode 3 - WriteTransparent: The contents of the background memory in the tablet are combined with and transferred to the visible LCD memory

Parameters:
Dest-0 = Foreground, 1 = Background memory in tablet
Mode-0, 1, 2, 3 as defined above
XPos-Location in LCD coords to draw at
YPos-Same
DrawFont-Not currently implemented, pass a 0
Str-ASCII hex string value.

Return Value: True if checksum received and verified, False if no or incorrect checksum.

Leave

Occurs when the input focus leaves the control.

Left (inherited from Control)

Gets or sets the x-coordinate of the control's left edge in pixels.

Location (inherited from Control)

Gets or sets the coordinates of the upper-left corner of the control relative to the upper-left corner of its container.

LocationChanged

Occurs when the Location property value has changed.

LostFocus

Occurs when the control loses focus.

MemberwiseClone

Overloaded. Releases the unmanaged resources used by the Control and optionally releases the managed resources.

MouseDown

Occurs when the mouse pointer is over the control and a mouse button is pressed.

MouseEnter

Occurs when the mouse pointer enters the control.

MouseHover

Occurs when the mouse pointer hovers over the control.

MouseLeave

Occurs when the mouse pointer leaves the control.

MouseMove

Occurs when the mouse pointer is moved over the control.

MouseUp

Occurs when the mouse pointer is over the control and a mouse button is released.

MouseWheel

Occurs when the mouse wheel moves while the control has focus.

Move

Occurs when the control is moved.

Name (inherited from Control)

Gets or sets the name of the control.

NumberOfTabletPoints

public int NumberOfTabletPoints();

Remarks: Returns the total number of points in the current signature, can be used to detect if a signature is present or not.

Return Value: Decimal value of number of points in the signature.

OnBackColorChanged

Raises the BackColorChanged event.

OnBackgroundImageChanged

Raises the BackgroundImageChanged event.

OnBindingContextChanged

Raises the BindingContextChanged event.

OnCausesValidationChanged

Raises the CausesValidationChanged event.

OnChangeUICues

Raises the ChangeUICues event.

OnClick

Raises the Click event.

OnContextMenuChanged

Raises the ContextMenuChanged event.

OnControlAdded

Raises the ControlAdded event.

OnControlRemoved

Raises the ControlRemoved event.

OnCreateControl

Raises the CreateControl event.

OnCursorChanged

Raises the CursorChanged event.

OnDockChanged

Raises the DockChanged event.

OnDoubleClick

Raises the DoubleClick event.

OnDragDrop

Raises the DragDrop event.

OnDragEnter

Raises the DragEnter event.

OnDragLeave

Raises the DragLeave event.

OnDragOver

Raises the DragOver event.

OnEnabledChanged

Raises the EnabledChanged event.

OnEnter
Raises the Enter event.

OnFontChanged
Raises the FontChanged event.

OnForeColorChanged
Raises the ForeColorChanged event.

OnGiveFeedback
Raises the GiveFeedback event.

OnGotFocus
Raises the GotFocus event.

OnHandleCreated
Raises the HandleCreated event.

OnHandleDestroyed
Raises the HandleDestroyed event.

OnHelpRequested
Raises the HelpRequested event.

OnImeModeChanged
Raises the ImeModeChanged event.

OnInvalidated
Raises the Invalidated event.

OnKeyDown
Raises the KeyDown event.

OnKeyPress
Raises the KeyPress event.

OnKeyUp
Raises the KeyUp event.

OnLayout
Raises the Layout event.

OnLeave
Raises the Leave event.

OnLocationChanged
Raises the LocationChanged event.

OnLostFocus
Raises the LostFocus event.

OnMouseDown
Raises the MouseDown event.

OnMouseEnter
Raises the MouseEnter event.

OnMouseHover
Raises the MouseHover event.

OnMouseMove

Raises the MouseMove event.

OnMouseUp

Raises the MouseUp event.

OnMouseWheel

Raises the MouseWheel event.

OnMove

Raises the Move event.

OnNotifyMessage

Notifies the control of Windows messages.

OnPaint

```
protected override void OnPaint(
  PaintEventArgs EvArgs
);
```

OnPaintBackground

Paints the background of the control.

OnParentBackColorChanged

Raises the BackColorChanged event when the BackColor property values of the control's container changes.

OnParentBackgroundImageChanged

Raises the BackgroundImageChanged event when the BackgroundImage property value of the control's container changes.

OnParentBindingContextChanged

Raises the BindingContextChanged event when the BindingContext property value of the control's container changes.

OnParentChanged

Raises the ParentChanged event.

OnParentEnabledChanged

Raises the EnabledChanged event when the Enabled property value of the control's container changes.

OnParentFontChanged

Raises the FontChanged event when the Font property value of the control's container changes.

OnParentForeColorChanged

Raises the ForeColorChanged event when the ForeColor property value of the control's container changes.

OnParentRightToLeftChanged

Raises the RightToLeftChanged event when the RightToLeft property value of the control's container changes.

OnParentVisibleChanged

Raises the VisibleChanged event when the Visible property value of the control's container changes.

OnQueryContinueDrag

Raises the QueryContinueDrag event.

OnResize

Raises the Resize event.

OnRightToLeftChanged
Raises the RightToLeftChanged event.

OnSizeChanged
Raises the SizeChanged event.

OnStyleChanged
Raises the StyleChanged event.

OnSystemColorsChanged
Raises the SystemColorsChanged event.

OnTabIndexChanged
Raises the TabIndexChanged event.

OnTabStopChanged
Raises the TabStopChanged event.

OnTextChanged
Raises the TextChanged event.

OnTimedEvent
protected void OnTimedEvent(
object *Source*,
ElapsedEventArgs *E*
);

OnValidated
Raises the Validated event.

OnValidating
Raises the Validating event.

OnVisibleChanged
Raises the VisibleChanged event.

Paint
Occurs when the control is redrawn.

Parent (inherited from Control)
Gets or sets the parent container of the control.

ParentChanged
Occurs when the Parent property value changes

PerformLayout
Overloaded. Forces the control to apply layout logic to all its child controls.

PointToClient
Computes the location of the specified screen point into client coordinates.

PointToScreen
Computes the location of the specified client point into screen coordinates.

PreProcessMessage
Preprocesses input messages within the message loop before they are dispatched.

ProcessCmdKey
Processes a command key.

ProcessDialogChar
Processes a dialog character.

ProcessDialogKey
Processes a dialog key.

ProcessKeyEventArgs
Processes a key message and generates the appropriate control events.

ProcessKeyMessage
Processes a keyboard message.

ProcessKeyPreview
Previews a keyboard message.

ProcessMnemonic
Processes a mnemonic character.

ProductName (inherited from Control)
Gets the product name of the assembly containing the control.

ProductVersion (inherited from Control)
Gets the version of the assembly containing the control.

QueryAccessibilityHelp
Occurs when AccessibleObject is providing help to accessibility applications

QueryContinueDrag
Occurs during a drag-and-drop operation and allows the drag source to determine whether the drag-and-drop operation should be canceled.

RecreateHandle
Forces the recreation of the handle for the control.

RecreatingHandle (inherited from Control)
Gets a value indicating whether the control is currently re-creating its handle.

RectangleToClient
Computes the size and location of the specified screen rectangle in client coordinates.

RectangleToScreen
Computes the size and location of the specified client rectangle in screen coordinates.

Refresh
Forces the control to invalidate its client area and immediately redraw itself and any child controls.

Region (inherited from Control)
Gets or sets the window region associated with the control.

ResetBackColor
Resets the BackColor property to its default value.

ResetBindings
Resets the DataBindings property to its default value.

ResetCursor
Resets the Cursor property to its default value.

ResetFont
Resets the Font property to its default value.

ResetForeColor
Resets the ForeColor property to its default value.

ResetImeMode
Resets the ImeMode property to its default value.

ResetRightToLeft
Resets the RightToLeft property to its default value.

ResetText
Resets the Text property to its default value.

Resize
Occurs when the control is resized.

ResizeRedraw
Gets or sets a value indicating whether the control redraws itself when resized.

ResumeLayout
Overloaded. Resumes normal layout logic.

Right (inherited from Control)
Gets the distance between the right edge of the control and the left edge of its container.

RightToLeft (inherited from Control)
Gets or sets a value indicating if control's elements are aligned to support locales using right-to-left fonts.

RightToLeftChanged
Occurs when the RightToLeft property values changes.

RtlTranslateAlignment
Overloaded. Converts the specified HorizontalAlignment to the appropriate HorizontalAlignment to support right-to-left text.

RtlTranslateContent
Converts the specified ContentAlignment to the appropriate ContentAlignment to support right-to-left text.

RtlTranslateHorizontal
Converts the specified HorizontalAlignment to the appropriate HorizontalAlignment to support right-to-left text.

RtlTranslateLeftRight
Converts the specified LeftRightAlignment to the appropriate LeftRightAlignment to support right-to-left text.

Scale
Overloaded. Scales the control and any child controls to the specified ratio.

ScaleCore
Performs the work of scaling the entire control and any child controls.

Select
Overloaded. Activates a child control. Optionally specifies the direction in the tab order to select the control from.

SelectNextControl
Activates the next control.

SendToBack
Sends the control to the back of the z-order.

SetAnnotate

```
public void SetAnnotate(
  string Annotate
);
```

Remarks: Sets Annotation string.

Parameters: Annotate – ASCII line character.

SetAutoKeyData

```
public void SetAutoKeyData(
  string KeyData
);
```

Remarks: Adds data to the auto key generation function. If called with file name (and path) when AutoKeyStart has not been initialized, this command will generate AutoKey data from a file rather than adding data via string. Used with AutoKeyStart and AutoKeyFinish methods, but called as a property.

Parameters: String containing the data, to be added to the key generation.

SetBounds

Overloaded. Sets the bounds of the control to the specified location and size.

SetBoundsCord

Performs the work of setting the specified bounds of this control.

SetClientSizeCore

Sets the size of the client area of the control.

SetDisplayAnnotate

```
public void SetDisplayAnnotate(
  bool DisplayAnnotate
);
```

Remarks: Sets the bean to display the Annotation string.

Parameters: DisplayAnnotate – Bean to display Annotation string.

SetDisplayAnnotateData

```
public void SetDisplayAnnotateData(
  int XPos,
  int YPos,
  int Size
);
```

SetDisplayAnnotatePosX

```
public void SetDisplayAnnotatePosX(
  int XPos
);
```

Remarks: Sets the X position in pixels of the start of the Annotation String in the signature box.

Parameters: XPOS – X position for the start of the Annotation String to be set.

SetDisplayAnnotatePosY

```
public void SetDisplayAnnotatePosY(
  int YPos
);
```

Remarks: Sets the Y position in pixels of the start of the Annotation String in the signature box.

Parameters: YPOS – Y position for the start of the Annotation String to be set.

SetDisplayAnnotateSize

```
public void SetDisplayAnnotateSize(
  int Size
);
```

Remarks: Sets the Y size of the Annotation start of the Time Stamp in the signature box.

Parameters: DisplayAnnotationSize – Y Size of Annotation text in pixels

SetDisplayMode

```
public void SetDisplayMode(
  int Mode
);
```

Remarks: *NOT CURRENTLY IMPLEMENTED.*

SetDisplayPenWidth

```
public void SetDisplayPenWidth(
  int PenWidth
);
```

Remarks: Sets pen ink width for the displayed signature in pixels.

Parameters: PenWidth – Pen width for the displayed signature in pixels.

SetDisplayRotate

```
public void SetDisplayRotate(
  bool Mode
);
```

Remarks: Sets mode allowing signature rotation in the control after capture for Display only, does not save the .sig info rotated.

Parameters: DisplayRotation - Orientation for display of signature after capture.

SetDisplayRotateSave

```
public void SetDisplayRotateSave(
  bool __unnamed000
);
```

Remarks: Sets mode allowing signature rotation and the save of signature in rotated format after capture. Does not save the .sig file rotated. Display rotation only. The rotation value is set by TabletRotation. Note: This is the preferred way of setting TabletMode = add 768 Can be used to rotate and then save in rotated format, signatures after capture, if the signature was accidentally taken in a rotated orientation during signature capture. Normally, to change the tablet orientation during capture, only the TabletRotation property is used.

Parameters: unnamed000-TRUE DisplayRotateSave mode = active, FALSE DisplayRotateSave mode = inactive

SetDisplayTimeStamp

```
public void SetDisplayTimeStamp(
  bool DisplayTimeStamp
);
```

Remarks: Sets the bean to display the developer provided Time Stamp string.

Parameters: DisplayTimeStamp-Component to be set to display Time Stamp.

SetDisplayTimeStampData

```
public void SetDisplayTimeStampData(
  int XPos,
```

```
  int YPos,
  int Size
);
```

SetDisplayTimeStampPosX

```
public void SetDisplayTimeStampPosX(
  int XPos
);
```

Remarks: Sets the X position in pixels of the start of the Time Stamp in the signature box.

Parameters: XPos-X position to be set for start of display of Time Stamp.

SetDisplayTimeStampPosY

```
public void SetDisplayTimeStampPosY(
  int YPos
);
```

Remarks: Sets the Y position in pixels of the start of the Time Stamp in the signature box.

Parameters: YPos-Y position to be set for start of display of Time Stamp.

SetDisplayTimeStampSize

```
public void SetDisplayTimeStampSize(
  int Size
);
```

Remarks: Sets the Y size in pixels of the Time Stamp in the signature box.

Parameters: Size – Size of Time Stamp to set.

SetDisplayWindowRes

```
public void SetDisplayWindowRes(
  bool Mode
);
```

Remarks: Sets mode which renders signatures in lower (screen) resolution for compatibility in printing directly from VB. *MUST BE USED WHEN PRINTING DIRECTLY FROM A VISUAL BASIC FORM.*

Parameters: Mode -TRUE DisplayWindowRes mode = active, FALSE DisplayWindowRes mode = inactive

SetEncryptionMode

```
public void SetEncryptionMode(
  int EncryptionMode
);
```

Remarks: Sets EncryptionMode.

Parameters: EncryptionMode-0= no encryption, 1= medium encryption, 2=higher security encryption mode.

SetImageAnnotate

```
public void SetImageAnnotate(
  bool ImageAnnotate
);
```

Remarks: Sets the bean to display the Annotation string as it applies to the WriteImageFile method.

Parameters: ImageAnnotate-Bean to be set to display Annotation string in the Image.

SetImageAnnotateData

```
public void SetImageAnnotateData(
  int ImageAnnotatePosX,
  int ImageAnnotatePosY,
  int ImageAnnotateSize
);
```

Remarks: Sets display screen info for Annotate string. The default is the lower right corner, at ~8% of the screen high.

Parameters:
ImageAnnotatePosX - X Location to display Annotate string at in signature display window using Logical Tablet Coordinates.
ImageAnnotatePosY - Y Location to display Annotate string at in signature display window using Logical Tablet Coordinates.
ImageAnnotateSize - Size to display Annotate string, in logical tablet coordinate height.

SetImageAnnotatePosX

```
public void SetImageAnnotatePosX(
  int ImageAnnotatePosX
);
```

Remarks: Sets the X position in pixels of the start of the Annotation String in the signature box as it applies to the WriteImageFile method.

Parameters: ImageAnnotatePosX - X position for the start of the Annotation String to be set for Image.

SetImageAnnotatePosY

```
public void SetImageAnnotatePosY (
  int ImageAnnotatePosY
);
```

Remarks: Sets the Y position in pixels of the start of the Annotation String in the signature box as it applies to the WriteImageFile method.

Parameters: ImageAnnotatePosY - Y position for the start of the Annotation String to be set for Image.

SetImageAnnotateSize

```
public void SetImageAnnotateSize (
  int ImageAnnotateSize
);
```

Remarks: Sets the Y size of the Annotation start in the signature box as it applies to the WriteImageFile method.

Parameters: ImageAnnotateSize - Y size of Annotation text in pixels for Image.

SetImageFileFormat

```
public void SetImageFileFormat(
  int FileFormat
);
```

Remarks: Sets the current format to use for Image files. The default is .BMP. The file extension is not assumed in the WriteImageFile function. Any extension can be specified, but it should match the specified file format. Note that writing an image file is completely different from a .sig file. An image file is just a standard image format of what is seen in the control and cannot be encrypted or decrypted by the SigPlus

control. The .sig file format does not store an image, but rather uses a unique method of preserving the original signature data from the tablet. To create a metafile with a transparent background use a trio of instructions to set TabletOpaque = False, then WriteImageFile, the TabletOpaque = True. For all other image files, TabletOpaque must be true when the image file is written.

Parameters:
FileFormat-File format for Image files.
0=Compressed BMP (default) must have .bmp ext.
1=Uncompressed BMP must have .bmp ext.
2=Mono. BMP must have .bmp ext.
3=JPG Q=20 must have .jpg ext.
4=JPG Q=100 must have .jpg ext.
5=Uncompressed TIF must have .tif ext.
6=Compressed TIF must have .tif ext.
7=WMF (windows metafile) must have .wmf ext.
8=EMF (enhanced metafile) must have .emf ext.
9=TIF (1-bit) must have .tif ext.
10=TIF (1-bit inverted) must have .tif ext.

SetImagePenWidth

```
public void SetImagePenWidth(
  int ImagePenWidth
);
```

Remarks: Sets pen ink width as it applies the WriteImageFile method.

Parameters: ImagePenWidth-Pen ink width for Image.

SetImageTimeStamp

```
public void SetImageTimeStamp(
  bool ImageTimeStamp
);
```

Remarks: Sets the component to display the Time Stamp as it applies to the WriteImageFile method.

Parameters: ImageTimeStamp – Component to be set to display developer provided Time Stamp.

SetImageTimeStampData

```
public void SetImageTimeStampData(
  int ImageTimeStampPosX,
  int ImageTimeStampPosY,
  int ImageTimeStampSize
);
```

Remarks: Sets display screen info for Time and Date stamp. The default is the lower left corner, at ~8% of the screen height.

Parameters:
ImageTimeStampPosX-X, Location to display TimeStamp string at in signature display window using Logical Tablet Coordinates.

ImageTimeStampPosY-Y Location to display TimeStamp string at in signature display window using Logical Tablet Coordinates.

ImageTimeStampSize-Size to display TimeStamp string, in logical tablet coordinates high.

SetImageTimeStampPosX

```
public void SetImageTimeStampPosX(
  int ImageTimeStampPosX
);
```

Remarks: Sets the X position in pixels of the start of the Time Stamp as it applies to the WriteImageFile method.

Parameters: ImageTimeStampPosX – X position to be set for start of Time Stamp for Image.

SetImageTimeStampPosY

```
public void SetImageTimeStampPosY(
  int ImageTimeStampPosY
);
```

Remarks: Sets the Y position in pixels of the start of the Time Stamp as it applies to the WriteImageFile method.

Parameters: ImageTimeStampPosY – Y position to be set for start of Time Stamp for Image.

SetImageTimeStampSize

```
public void SetImageTimeStampSize(
  int ImageTimeStampSize
);
```

Remarks: Set the Y size in pixels of the Time Stamp in the signature box as it applies to the WriteImageFile method.

Parameters: ImageTimeStampSize – Size of Time Stamp to set for Image

SetImageXSize

```
public void SetImageXSize(
  int ImageXSize
);
```

Remarks: Set the number of X pixels in the image provided by the WriteImageFile method.

Parameters: ImageXSize – Size in X pixels of the Image width.

SetImageYSize

```
public void SetImageYSize(
  int ImageYSize
);
```

Remarks: Set the number of Y pixels in the image height provided by the WriteImageFile method.

Parameters: ImageYSize – Size in Y pixels of the Image height.

SetJustifyMode

```
public void SetJustifyMode(
  int JustifyMode
);
```

Remarks: Sets the current justification mode- how the signature is sized and positioned in the signature box as well as when using the WriteImageFile method.

Parameters: JustifyMode - Justification mode, 0-normal no justification, 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).

SetJustifyX

public void SetJustifyX(
  int *JustifyX*
);

Remarks: Sets the buffer size in Logical Tablet Coordinates of "dead space" of left and right edge of SigPlus object if JustifyMode is 1-5. This method functions for both the signature box as well as when using the WriteImageFile method.

Parameters: JustifyX-Justification X buffer size in pixels to be set.

SetJustifyY

public void SetJustifyY(
  int *JustifyY*
);

Remarks: Sets the buffer size in Logical Tablet Coordinates of "dead space" of top and bottom edge of SigPlus object if JustifyMode is 1-5. This method functions for both the signature box as well as when using the WriteImageFile method.

Parameters: JustifyY-Justification Y buffer size in pixels to be set.

SetKeyString

public void SetKeyString(
  string *KeyString*
);

Remarks: Sets the Key String into the SigPlus component.

Parameters: KeyString - Hash of the data used to encrypt/decrypt the signature, key internally generated by SigPlus.

SetLCDCaptureMode

public void SetLCDCaptureMode(
  int *CaptureMode*
);

Remarks: Sets the current LCD Capture Mode for the tablet.

Parameters:
CaptureMode-Mode the LCD is set to capture signatures in,
Mode 0=no LCD commands are sent to the tablet
Mode 1=sets capture mode to be active with Autoerase in the tablet
Mode 2=sets the tablet to persistent ink capture without autoerase
Mode 3=signature ink is displayed inverted on a suitable dark background set using the Graphic functions.

SetSaveSigInfo

public void SetSaveSigInfo(
  bool *SaveSigInfo*
);

Remarks: Enables/disables the saving of TimeStamp and Annotate data in the signature.

Parameters: SaveSigInfo – If True then SigInfo will be saved (default), if False then the info will not be saved.

SetSigCompressionMode

public void SetSigCompressionMode(
  int *CompressionMode*
);

Remarks: Sets the current compression mode for signatures.

Parameters: CompressionMode-Mode for compression of signature, where 0= no compression, 1= lossless compression with compacted data format, 2-8= compression ratio of signature stored in in .sig file where 2=1KB typ, 4=500 byte typ, and 8=250 byte typ. Topaz Systems does not recommend compressing beyond setting 1 unless size is more important than signature quality.

SetSigString

public void SetSigString(
  string *SigString*
);

Remarks: Puts signature into the component.

Parameters: SigString – Signature in ASCII format

SetSigWindow

public void SetSigWindow(
  short *Coords*,
  short *NewXPos*,
  short *NewYPos*,
  short *NewXSize*,
  short *NewYSize*
);

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.

Parameters:
Coords-Coordinate system used for this hot spot, 0 = Logical tablet coordinates, 1 = LCD Coordinates.
NewXPos-Location in logical tablet coordinates (upper left - 0,0).
NewYPos-Same
NewXSize-XSize in logical tablet pixels
NewYSize-YSize in logical tablet pixels

SetStyle

Sets the specified style bit to the specified value.

SetTabletBaudRate

public void SetTabletBaudRate(
  int *BaudRate*
);

Remarks: Sets TabletBaudRate, an internal property associated with tablet model.

Parameters: BaudRate – internal tablet property.

SetTabletComPort

public void SetTabletComPort(
  int *Port*
);

Remarks: Sets the COM port to use using a string. The SigPlus.NET component does not lock up a port as is the case with mouse-type drivers. Only set COM port when tablet state is OFF.

Parameters: Port-

SetTabletComTest

public void SetTabletComTest(
  int *ComTest*
);

Remarks: Sets hardware check mode. When this mode is active and Topaz tablet plugged into selected COM port (or USB) TabletState can be set to 1(ON). If tablet cannot be set to 1.

Parameters: ComTest-Hardware check mode.

SetTabletFilterPoints

public void SetTabletFilterPoints(
  int *Points*
);

Remarks: Sets the TabletFilterPoints, an internal property associated with tablet model.

Parameters: Points – Internal tablet property.

SetTabletLogicalXSize

public void SetTabletLogicalXSize(
  int *XSize*
);

Remarks: Sets the range of horizontal values to be used in representing signatures. This has no relation to the displayed, image file, or tablet sizes. This is the X-range used for the Topaz vector format, and the internally used format.

Parameters: XSize – Integer value of Tablet logical size. Default is 2150.

SetTabletLogicalYSize

public void SetTabletLogicalYSize(
  int Y*Size*
);

Remarks: Sets the range of vertical values to be used in representing signatures. This has no relation to the displayed, image file, or tablet sizes. This is the Y-range used for the Topaz vector format, and the internally used format.

Parameters: YSize – Integer value of Tablet logical size. Default is 1400.

SetTabletResolution

public void SetTabletResolution(
  int *Resolution*
);

Remarks: Sets TabletResolution, an internal property associated with tablet model and set by the TabletModel property, based on hardware tablet resolution is 410 dpi, (excluding ClipGem which is 275 dpi) but can be changed at the risk of affecting signature capture.

Parameters: Resolution – internal tablet property.

SetTabletRotation

```
public void SetTabletRotation(
  int Rotation
);
```

Remarks: Sets the orientation on a 360 degree axis for display of tablet data. The data in the sig representation is stored in the native tablet orientation.

Parameters: Rotation – Display orientation, allowed values are 0, 90, 180, 270.

SetTabletState

```
public void SetTabletState(
  int State
);
```

Remarks: Enables tablet to access the COM or USB port to capture signatures or not.

Parameters: State-setting to 1 enables the tablet to capture signatures as above, setting to 0 disables signature capture.

SetTabletTimingAdvance

```
public void SetTabletTimingAdvance(
  int Advance
);
```

Remarks: Sets the TabletTimingAdvance, an internal property associated with tablet model.

Parameters: Advance – internal tablet property.

SetTabletType

```
public void SetTabletType(
  int TabletType
);
```

Remarks: Determines if the tablet will accept data from a com port, WinTab driver, USB driver or other method of data input. If WinTab support is not available, it will not do anything when in the active state. Conversely, if WinTab is present and the mode is not correct, the tablet will also do nothing when active, because the WinTab driver takes exclusive possession of the Com Port. This is the preferred way of setting TabletMode for tablet input.

Parameters:
TabletType-Default is 0
0=Normal mode. When tablet is activated it will accept input from the selected com port.
1=WinTab mode, when the tablet is activated, it will accept data from the Topaz WinTab driver only.
2=USB mode, when the tablet is activated, it will accept data from the Topaz USB driver
3=TracGemPOST signature format, Transparent Mode (CTRL T)
4=Older-model SigLite touch tablet format, an obsolete mode
6=HSB tablet (USB mode for tablets using HID driver)

SetTabletXStart

```
public void SetTabletXStart(
  int XStart
);
```

Remarks: Sets the X position in Logical Tablet Coordinates of the upper left hand corner of the bean signature box.

Parameters: XStart – X coordinate of the upper left corner of the signature box.

SetTabletXStop

```
public void SetTabletXStop(
  int XStop
);
```

Remarks: Sets the X position in Logical Tablet Coordinates of the lower right hand corner of the bean signature box.

Parameters: XStop – X coordinate for the lower right corner of the signature box.

SetTabletYStart

```
public void SetTabletYStart(
  int YStart
);
```

Remarks: Sets the Y position in Logical Tablet Coordinates of the upper left hand corner of the bean signature box.

Parameters: YStart – Y coordinate for the upper left corner of the signature box.

SetTabletYStop

```
public void SetTabletYStop(
  int YStop
);
```

Remarks: Sets the Y position in Logical Tablet Coordinates of the lower right hand corner of the bean signature box.

Parameters: YStop – Y coordinate for the lower right corner of the signature box.

SetTimeStamp

```
public void SetTimeStamp(
  string TimeStamp
);
```

Remarks: Sets the TimeStamp string for the signature.

Parameters: TimeStamp – ASCII new line character.

SetTopLevel

Sets the control as the top-level control.

SetUseAmbientColors

```
public void SetUseAmbientColors(
  bool UseAmbientColors
);
```

Remarks: Must be enabled to allow the ForeColor and BackColor of the object to be modified.

SetVisibleCore

Sets the control to the specified visible state.

Show

Displays the control to the user.

ShowFocusCues

Gets a value indicating whether the control should display focus rectangles.

ShowKeyboardCues

Gets a value indicating whether the control should display keyboard shortcuts.

SigPlusNET Constructor

Initializes a new instance of the SigPlusNET class.

```
public SigPlusNET () ;
```

Site (inherited from Control)

Gets or sets the site of the control.

Size (inherited from Control)

Gets or sets the height and width of the control.

SizeChanged

Occurs when the Size property value changes.

Sleep

```
public void Sleep(
  uint TimeInMs
);
```

StyleChanged

Occurs when the control style changes.

SuspendLayout

Temporarily suspends the layout logic for the control.

SystemColorsChanged

Occurs when the system colors change.

TabIndex (inherited from Control)

Gets or sets the tab order of the control within its container.

TabIndexChanged

Occurs when the TabIndex property value changes.

TabletConnectQuery()

```
public bool TabletConnectQuery()
```

Note: TabletConnectQuery() can only be used in a local environment. It cannot be used in a Terminal Server or Citrix environment.

Return Value: Boolean indicating if signature pad is connected. Uses TabletType value to determine what signature pad connection type to use.

TabletModelNumber()

public int TabletModelNumber();

Remarks: Please note: SetTabletState(1) must be successfully set before TabletModelNumber() can return a value. TabletModelNumber() returns a value corresponding to a particular Topaz tablet model, used to detect whether the signature pad in question is connected.

Parameters: None.

Return: Model Number. The following list shows the return from TabletModelNumber() and the corresponding Topaz tablet model.

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

To differentiate "SE" pads from one another (see 58 above), use TabletSerialNumber(), and the following values correspond to these "SE" pads:

550 = TLBK766SE
551 = TLBK462SE
553 or 557 = TLBK755SE or TLBK750SE

See TabletSerialNumber() below for further details.

TabletSerialNumber()

public long TabletSerialNumber();

Remarks: Please note: SetTabletState(1) must be successfully set before TabletMSerialNumber() can return a value. Given the use of these Topaz tablet models:

T-L(BK)462
T-LBK57GC
T-LBK43LC

TabletSerialNumber() returns a unique value corresponding to a particular Topaz tablet. This value can be used to differentiate one signature tablet device from another of the same tablet model type (given the 3 models listed above).

Additionally, given a Topaz 'SE' tablet model (one of the following):

TLBK462SE
TLBK766SE
TLBK755SE
TLBK750SE

TabletSerialNumber() is used as a secondary model number identifier for determining which specific 'SE' tablet is connected. Given the TabletModelNumber() function, a return of 58 indicates that an 'SE' signature pad is connected. At this point, TabletSerialNumber() can now be used to further identify which specific 'SE' tablet is connected. The following TabletSerialNumber() returns correspond to these 'SE' tablet models:

550 = TLBK766SE
551 = TLBK462SE
553 or 557 = TLBK755SE or TLBK750SE

No other Topaz tablet models are able to return a TabletSerialNumber() value for particular use.

Parameters: None.

Return: Serial or further refined Model Number depending upon usage.

TabStop (inherited from Control)
Gets or sets the value indicating whether the user can give the focus to this control using the TAB key.

TabStopChanged
Occurs when the TabStop property value changes.

Tag (inherited from Control)
Gets or sets the object that contains data about the control.

Text (inherited from Control)
Gets or sets the text associated with this control.

TextChanged
Occurs when the Text property value changes.

Top (inherited from Control)
Gets or sets the y-coordinate of the control's top edge of pixels.

TopLevelControl (inherited from Control)
Gets the parent control that is not parented by another Windows Form control. Typically, this is the outermost Form that the control is contained in.

Update
Causes the control to redraw the invalidated regions within its client area.

UpdateBounds
Overloaded. Updates the bounds of the control with the current size and location.

UpdateStyles
Forces the assigned styles to be reapplied to the control

UpdateZOrder
Updates the control in its parent's z-order.

Validated
Occurs when the control is finished validating

Validating
Occurs when the control is validating.

Visible (inherited from Control)
Gets or sets a value indicating whether the control is displayed.

VisibleChanged
Occurs when the Visible property value changes.

Width (inherited from Control)
Gets or sets the width of the control

WndProc
Processes Windows messages.

WriteImageFile

```
public bool WriteImageFile(
  string FileName
);
```

Remarks: Scheduled for removal. WriteImageFile() should not be used. Instead, please refer to the GetSigImage() function which returns the signature as a System.Drawing.Image

Return Value: N/A

Parameters: Filename – N/A