

**Version 2.60, 2/15/12**
**Revision control**

2.60	Added support for RXTX serial API. Updated support for TabletModelNumber()
2.52	Added support for SigGemColor5.7 signature pad.
2.51	Optimized bean, improved HID USB tablet performance.
2.47	Added getTabletSerialNumber() and getTabletmodelNumber() functions. Added getPointPValue() function. Optimized bean.
2.45	Added basic SE-pad HSB support. Optimized bean.
2.44	Internal processing changes.
2.41	Internal processing changes. Added EncryptionMode 3.
2.40	Switched to com.topaz.sigplus from com.gemtools.sigplus
2.32	Fixed bug in SigImage() method.
2.31	Fixed bug in TabletComTest() method.
2.29	Added getXExtent() and getYExtent() to allow perfectly-sized buffered images returned from sigImage(). Modified to allow justified images to be generated with TabletState == 1. Added getNumberOfStrokes(), getNumPointsForStroke(), getPointXValue(), getPointYValue() for returning signature points one at a time. Improved HID tablet performance. Improved LCD interactive performance in general. Improved performance of the lcdWriteImage() method.
2.24	Removed minor bug from 2_23.jar, to prevent possibility of buffer underflow or overflow.
2.23	Improved speed of text code. Added a text-only mode.
2.20	Performance update, providing serial tablet functionality under Linux/Unix.
2.19	Improved capture performance in conjunction with KeyPadSetSigWindow() method.
2.18	Improved tablet mapping for LCD4X5 tablets.
2.16	Improved HID tablet redraw characteristics.
2.15	Added support for Topaz HSB (USB w/HID driver) tablets. Added setTabletComTest and getTabletComTest methods to determine if tablet is plugged in.
2.10	Improved LCD interaction. Added support for SignatureGem LCD 1X5 tablet. Added serial and model number support. Improved screen refresh of SigPlus.
2.06	Allows USB tablet connectivity. Improvements made to signature rendering.



#### License Agreement and Limited Warranty

**IMPORTANT:** Please read this document before continuing the software load procedure. By loading the software enclosed with this agreement, you are indicating acceptance of the terms of this legal agreement between you (herein called Licensee) and Topaz Systems, Inc. (herein called Topaz). If you do not agree to the terms of this agreement, do not load the enclosed software and promptly return the product.

1. **Limited use License:** Topaz and its suppliers (if any) grant you the right to use the software for use with Topaz Gem-Series tablets only. The software is owned for distribution exclusively by Topaz and is protected by the United States Patent and Trademark laws and international treaties.

2. **Governing Law, Jurisdiction, and Forum:** This agreement is governed by the laws of the State of California, County of Ventura.

3. **YOU MAY:**

(a) Freely use, copy, and distribute the enclosed software only for use with Topaz Gem-Series tablets.

4. **YOU MAY NOT:**

- (a) Use, distribute or modify the Topaz software for use with tablets not supplied by Topaz Systems, Inc.
- (b) Reverse engineer, decompile, or dis-assemble, the software.

5. **LIMITED WARRANTY:**

Topaz Systems warrants to the original buyer only, that the media upon which the Software is recorded is free from defects in workmanship and material under normal use and service for a period of 30 days.

6. **EXCLUSIVE REMEDY:**

Topaz's entire liability and your exclusive remedy shall be, at Topaz's option, either (a) the repair or replacement of the software or (b) the refund of the price that was paid for the software package. The defective software, along with proof of payment, must be returned to Topaz within the terms of the warranty as set forth in this agreement.

7. **LIMITATIONS ON DAMAGES:**

In no event shall Topaz be liable for damages whatsoever, (including without limitation, damages for loss of profits, business interruption, loss of information, or other pecuniary loss) arising out of the use of or inability to use the software even if Topaz or its suppliers, if any, have been of the possibility of such damages. In no event will Topaz's liability for any reason exceed the actual price paid for the license to use the specific program.

8. **NO OTHER WARRANTIES:**

With respect to the software, media, and written material, Topaz and its suppliers, if any, disclaim all warranties, other

than the above Topaz warranty, including but not limited to warranties merchantability or fitness for a particular use.

Topaz does not warranty the software will meet or requirements or that the operations of the software will be uninterrupted or error free.

9. **GOVERNING LAW AND GENERAL PROVISIONS:**

If any part of this agreement is found void and unenforceable, it will not effect the validity of the balance of the agreement,

which shall remain valid and enforceable according to its terms. You agree that the software will not be shipped, transferred, or exported into any country or used in any manner prohibited by the United States Export Administration Act or any other export laws, restrictions or regulations. This agreement shall automatically terminate upon failure of you to comply with its terms. This agreement may only be modified in writing signed by an officer of Topaz Systems, Inc. This agreement will not be governed by the United Nations convention on contracts for the International sale of goods, the application of which is expressly excluded.

**FOR TECHNICAL SUPPORT CONTACT TOPAZ SYSTEMS AT 805 520-8286 OR**

**E-MAIL TO [support@topazsystems.com](mailto:support@topazsystems.com)**

**IMPORTANT NOTE ON THE SIGUSB.DLL FILE:**

Starting with SigPlus2\_40.jar and above, SigUsb.dll v 1.6 is required for all USB tablet operations.

## GENERAL RELEASE NOTES

### Setting the COM Port (or USB, using version 2.06 or later):

Be sure that tablet state is off when selecting com port setTabletState(0)--Com port must be selected first, and then tablet state turned on. Also be sure to turn tablet state off before exiting the application. You will need the Java Communications API installed for your target environment to run an application with the SigPlus Java bean. Please review the readme file in the javacomm folder for details.

With version 2.06 and above, USB tablet connectivity is optional within certain operating systems. Again, the USB port (as with the COM described above) must be selected first, before turning tablet on.

For Windows machines, the Topaz USB drivers (TopazUsb.sys and TopazAuto.inf) must be installed prior to usage. To install, download the latest install from the web site at <http://www.topazsystems.com/Software/topazusb.exe>

In addition, the SigUsb.dll must be placed in the path, or where your environment will find it. This DLL is necessary for USB functionality. **See note on SigUsb.dll at the bottom of page 2.**

To set the port (COM, USB or Topaz "HSB"), please see the method TabletComPort on p. 21)

### Additional Information:

Refer to SigPlus.doc and SigPlus\_Manual.doc for information about the windows ActiveX implementation of this software tool <http://www.topazsystems.com/Software/sigplusdocs.zip>

### Topaz Java Demos:

You will find demos included for the following:

- Interactive Topaz LCD4X3 tablet demo
- Interactive Topaz LCD 1X5 demo
- Simple serial signing demo
- Simple USB signing demo
- Graphic interactive demo

Topaz Java demos have been set up for a Windows environment. Please change as necessary for your own.

If you experience problems trying to use CommAPI in an Applet, the following was tested and works on a Win32 platform:

1. In .java.policy, add the following lines:

```
grant signedBy "YOUR_ALIAS_HERE"
permission java.lang.RuntimePermission "loadLibrary.win32com";
permission java.io.FilePermission "${java.home}\\lib\\ext\\win32com.dll", "read";
permission java.io.FilePermission "${java.home}\\lib\\ext\\x86\\win32com.dll", "read";
permission java.io.FilePermission "${java.home}\\lib\\javax.comm.properties", "read";
permission java.io.FilePermission "${java.home}\\lib\\javax.comm.properties", "delete";
permission java.util.PropertyPermission "java.home", "read";
permission java.util.PropertyPermission "javax.comm.properties", "read";
};
```

(If you prefer, just add permission java.security.AllPermission;) (Don't be afraid about the "delete" permissions - nothing will happen to the file, driver.initialize() checks these permissions...)

2. Before calling any CommAPI routine, insert the following statements. The driver.initialize() statement will emit a "Caught..." message, just as before, but this time you can ignore it!

```
String drivename = "com.sun.comm.Win32Driver"; try
CommDriver driver = (CommDriver) Class.forName(drivename).newInstance();
driver.initialize();
} catch (Throwable th) { /* Discard it */ }
After executing these statements,
CommPortIdentifier.getPortIdentifier("COM1");
won't emit a "name can't be null" message!
```



For more information: from  
samples/porting/CommPortIdentifier.java!

**Important Notice:**

These guidelines or any or all additional documentation or examples do not constitute a warranty about the performance, security, or legal acceptability of SigPlus software or the SigPlus control in any specific use or implementation. To the extent that SigPlus or SigPlus are used to achieve regulatory or other specific objectives within an industry, you must consult competent experts or regulatory officials together with your own plan to achieve your desired business objectives using the Topaz tools.

Properties and Methods for *SigPlus*<sup>™</sup> Java Bean:

**NEW METHODS IN FULL RELEASE (From 2.14)**

The SigPlus Java Bean is designed to mimic the functions and the function names of the SigPlus Active-X control. For a complete description of the functions, please download the Active-X document SigPlus.doc.

Several of the following pages give detailed information for the Java version of SigPlus. The initial production release includes support for the functions listed below.

**public methods**

```

public Image sigImage()

public boolean exportSigFile( String fileName )

public boolean importSigFile( String fileName )

public void clearTablet( )

public boolean getTabletInvisible()

public void setTabletInvisible( boolean newValue )

public void keyPadSetSigWindow( int coords, int xPos, int yPos, int xSize, int ySize )

public void keyPadClearSigWindow( int inside )

public void keyPadAddHotSpot( int keyCode, int coords, int xPos, int yPos, int xSize, int ySize )

public void keyPadClearHotSpotList( )

public int keyPadQueryHotSpot( int keyCode )

public int getLCCaptureMode( )

public void setLCCaptureMode( int newMode )

public void setLCDTabletMap( int newLCDType, int newLCDXSize, int newLCDYSize,
                             int newLCDXStart, int newLCDYStart, int newLCDXStop, int newLCDYStop )

public boolean lcdSetWindow( int xPos, int yPos, int xSize, int ySize )

public boolean lcdWriteImage( int dest, int mode, int xPos, int yPos, int xSize, int ySize, Image
                              imageData )

public boolean lcdRefresh( int mode, int xPos, int yPos, int xSize, int ySize )

public int getNumberOfStrokes()

public int getNumPointsForStroke()

public int getPointXValue()

public int getPointYValue()

public int getXExtent()

public int getYExtent()

```

```
public String getTimeStamp()

public void setTimeStamp( String newValue )

public String getAnnotation()

public void setAnnotation( String newValue )

public String getSigString()

public void setSigString( String sigString )
public int getTabletState()

public void setTabletState( int newValue )

public int getTabletLogicalXSize()

public int getTabletLogicalYSize()

public int getTabletXStart()

public void setTabletXStart( int newValue )

public int getTabletXStop()

public void setTabletXStop( int newValue )

public int getTabletYStart()

public void setTabletYStart( int newValue )

public int getTabletYStop()

public void setTabletYStop( int newValue )

public int getTabletFilterPoints()

public void setTabletFilterPoints( int newValue )

public int getTabletTimingAdvance()

public void setTabletTimingAdvance( int newValue )

public int getTabletBaudRate()

public void setTabletBaudRate( int newValue )

public int getTabletResolution( )

public void setTabletResolution( int newValue )

public int getTabletRotation()

public void setTabletRotation( int newValue )

public String getTabletComPort()

public void setTabletComPort( String newValue )

public boolean getTabletComTest()
```

```

public void setTabletComTest( boolean newValue )

public int getTabletFormat()

public void setTabletFormat( int newValue )

public String getTabletModel()

public void setTabletModel( String newValue )

public boolean getTabletClippingMode()

public void setTabletClippingMode( boolean newValue )

public int getTabletLCDType()

public void setTabletLCDType( int newValue )

public int getTabletLCDXSize()

public void setTabletLCDXSize( int newValue )

public int getTabletLCDYSize()

public void setTabletLCDYSize( int newValue )

public int getTabletLCDXStart()

public void setTabletLCDXStart( int newValue )

public int getTabletLCDXStop()

public void setTabletLCDXStop( int newValue )

public int getTabletLCDYStart()

public void setTabletLCDYStart( int newValue )

public int getTabletLCDYStop()

public void setTabletLCDYStop( int newValue )

```

#### **// Display properties**

```

public float getDisplayPenWidth()

public void setDisplayPenWidth( float newValue )

public boolean getDisplayTransparentMode()

public void setDisplayTransparentMode( boolean newValue )

public int getDisplayRotation()

public void setDisplayRotation( int newValue )

public int getDisplayJustifyX()

public void setDisplayJustifyX( int newValue )

public int getDisplayJustifyY()

```

```

public void setDisplayJustifyY( int newValue )
public int getDisplayJustifyMode()
public void setDisplayDisplayMode( int newValue )
public int getDisplayTimeStampX()
public void setDisplayTimeStampX( int newValue )
public int getDisplayTimeStampY()
public void setDisplayTimeStampY( int newValue )
public int getDisplayTimeStampSize()
public void setDisplayTimeStampSize( int newValue )
public int getDisplayAnnotationX()
public void setDisplayAnnotationX( int newValue )
public int getDisplayAnnotationY()
public void setDisplayAnnotationY( int newValue )
public int getDisplayAnnotationSize()
public void setDisplayAnnotationSize( int newValue )
public boolean getDisplayTimeStamp()
public void setDisplayTimeStamp( boolean newValue )
public boolean getDisplayAnnotation()
public void setDisplayAnnotation( boolean newValue )

```

#### **// Image properties**

```

public int getImageXSize()
public void setImageXSize( int newValue )
public int getImageYSize()
public void setImageYSize( int newValue )
public float getImagePenWidth()
public void setImagePenWidth( float newValue )
public boolean getImageTransparentMode()
public void setImageTransparentMode( boolean newValue )

```



```
public int getImageRotation()
public void setImageRotation( int newValue )
public int getImageJustifyX()
public void setImageJustifyX( int newValue )
public int getImageJustifyY()
public void setImageJustifyY( int newValue )
public int getImageJustifyMode()
public void setImageJustifyMode( int newValue )
public int getImageTimeStampX()
public void setImageTimeStampX( int newValue )
public int getImageTimeStampY()
public void setImageTimeStampY( int newValue )
public int getImageTimeStampSize()
public void setImageTimeStampSize( int newValue )
public int getImageAnnotationX()
public void setImageAnnotationX( int newValue )
public int getImageAnnotationY()
public void setImageAnnotationY( int newValue )
public int getImageAnnotationSize()
public void setImageAnnotationSize( int newValue )
public boolean getImageTimeStamp()
public void setImageTimeStamp( boolean newValue )
public boolean getImageAnnotation()
public void setImageAnnotation( boolean newValue )
public void autoKeyData(String)
public void autoKeyStart( )
public void autoKeyFinish( )
public int getEncryptionMode()
public String getKeyString()
```



```
public String getSigPlusVersion()
{
    public int getSigReceipt( )
    {
        public String getSigReceiptAscii( )
        {
            public int numberOfTabletPoints( )

        public void setEncryptionMode( int encryptionMode )

        sig.setSaveSigInfo( boolean );

        public void setSigCompressionMode( int compMode )
```

### DETAILED DESCRIPTION OF SELECTED METHODS

For complete description of all methods, see SigPlus.doc for the SigPlus Active-X control

**public void keyPadSetSigWindow( int coords, int xPos, int yPos, int xSize, int ySize )**

Function: 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 will only operate on points inside the window.  
SigString only operates on points inside of the window.

Arguments:      Integers:

XPos	Location in logical tablet coordinates (upper left - 0,0)
YPos	Same
XSize	XSize in logical tablet pixels
YSize	YSize in logical tablet pixels

Return Value:    Void

Remarks:

This behavior is enabled by setting the start and stop values to a non-zero value. The window defaults to (0,0,0,0), disabled at startup. The window can be enabled at one spot, re-enabled at another and so on, 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.

method

**public void keyPadClearSigWindow( int inside )**

if inside = 0, then signature data is erased (inside sig window)  
if inside = 1, the points outside of sig window are cleared

method

**public void keyPadAddHotSpot( int keyCode, int coords, int xPos, int yPos, int xSize, int ySize )**

Function: Defines in software the location of a tablet hotspot in logical tablet coordinates. The coordinates of the HotSpot are defined in logical tablet coordinates with (0,0) being the upper left-most pixel. To use LCD coordinates, the LCD coordinate map is set using setLcdTabletMap.

Arguments:      Integers:

KeyCode	Integer value defining the HotSpot
Coords	Coordinate system used for this hot spot 0 = Logical tablet coordinates 1 = LCD Coordinates.
XPos	Location (upper left - 0,0)
YPos	Same
XSize	XSize in pixels
YSize	YSize in pixels

Return Value:    Void

Remarks:

method

**public void keyPadClearHotSpotList( )**

KeyPadClearHotSpotList()

Function: This method clears the controls internal list of hotspots, created using KeyPadAddHotSpot.

Arguments: None

Return Value: None.

Remarks:

method

**public int keyPadQueryHotSpot( int keyCode )**

Function: This method queries the data points currently in the control against the logical tablet coordinates mapped by KeyCode. This method returns a true if the control contains data that is within the definition of the KeyCode area on the tablet.

Arguments: Integer

Return Value: Integer, the number of points within the KeyCode definition.

Remarks:

property

**public int getLCDCaptureMode( )****public void setLCDCaptureMode( int newMode )**

Function: This property sets the current LCD Mode for the tablet, the tablet is put into the mode as well.

Mode 0 – No LCD Tablet. No LCD commands are sent to the tablet

Mode 1 - Capture Default. CTRL-D is sent to the tablet, which clears the tablet and sets capture mode to be active with Autoerase in the tablet.

Mode 2 - Capture Ink CTRL-T is sent to the tablet, putting the tablet in persistent ink capture mode where the tablet does not automatically clear the display.

Mode 3 - Capture Ink Inverted: CTRL-I is sent to the tablet, where signature ink is displayed inverted against a suitable dark background set using the Graphic functions. Autoerase in the tablet is disabled.

Remarks:

If TabletState is TRUE, the mode command is sent to the tablet immediately. If Tabletstate = false, then this inking mode command is sent when the tablet state is next set to true. When LCDWrite functions, or LCDRefresh are called, and tablet state is TRUE, the mode will automatically be set after completing the Write or Refresh function.

```
public void setLCDTabletMap( int newLCDType, int newLCDXSize, int newLCDYSize,
    int newLCDXStart, int newLCDYStart, int newLCDXStop, int newLCDYStop )
```

Function: Used to override the default values for the LCD parameters at run time.

Arguments:      Integers:

LCDType	Specifies LCD type and format, 0 for 240x128,
LCDXSize	X Size of LCD display, in pixels
LCDYSize	Y Size of LCD display, in pixels
LCDXStart	X Pos in logical tablet coordinates of LCD
LCDYStart	Y Pos in logical tablet coordinates of LCD
LCDXStop	X Pos in logical tablet coordinates of LCD
LCDYStop	Y Pos in logical tablet coordinates of LCD

Return Value:    Void

Remarks:

```
public boolean lcdSetWindow( int xPos, int yPos, int xSize, int ySize )
```

Function: This function sets the tablet so that the LCD display will be showing ink only in a restricted area when data is input with a pen in LCDCaptureMode = 2 and = 3 inking modes. Returning the tablet to default state (such as using LCDCaptureMode = 1) will reset these values.

Arguments:      Integers:

XPos	Location in LCD coordinates (upper left - 0,0)
YPos	Same
XSize	XSize in LCD pixels
YSize	YSize in LCD pixels

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

Remarks:        Do not send a command with XSize or Ysize = 0

```
public boolean lcdWriteImage( int dest, int mode, int xPos, int yPos, int xSize, int ySize,
    Image imageData )
```

Function: Used to write windows bitmap 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

Arguments:      Integers:

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
XSize	Width in LCD pixels
YSize	Height in LCD pixels

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

Remarks: Note that this function only can occur on 8 LCD-pixel boundaries on the LCD tablet unit.

**public boolean lcdRefresh( int mode, int xPos, int yPos, int xSize, int ySize )**

Function: The tablet is sent a refresh command with 4 possible modes

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 transferred to the LCD display, and combined with the contents of the LCD display.

Arguments:      Integers:

Mode	0, 1, 2, 3 as defined above
XPos	Location in LCD coordinates (upper left - 0,0)
YPos	Same
XSize	XSize in LCD pixels
YSize	YSize in LCD pixels

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

Remarks: Note that this function only can occur on 8 LCD-pixel boundaries on the LCD tablet unit.

## **METHODS FOR SIGPLUS JAVA BEAN:**

### **Implemented Methods:**

public long getSigReceipt( )

Function:

Returns a 32 bit receipt similar to the key receipt. The receipt is formed, by using the auto key generation algorithm on the signature file data. The result can be used to verify that the signature has not been modified

Argument:

none

Return Value:

long 32 bit binary receipt.

public String getSigReceiptAscii(short Strokeldx, short PointIdx)

Function:

Same as GetKeyReceiptAscii, but for Sig receipt.

Argument:

none

Return Value:

String The Ascii string

public String GetKeyString()

Function:

Provides from the control in Ascii compatible format, the encryption key.

Argument:

none

Return Value:

String KeyString

Remarks:

Can be used to move keys from one instance of the control to another.

public short numberOfTabletPoints()

Function:

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

Argument:

none

Return Value:

short decimal value of number of points in the signature.

Remarks:

A signature should consist of at least 200 points to be considered valid, as this represents approximately 1 second of active signature time.

public void autoKeyFinish( )

Function:

Completes the auto key generation function. After this call, the key is ready to be used in saving an encrypted file.

Argument:

none

Return Value

void

Remarks:

public void autoKeyStart( )

Function:

Initializes the automatic key generation function.

Argument:

none

Return Value:

void

Remarks:

The automatic key generation function will derive a key from the data fed to it via autoKeyData(String). AutoKeyStart() is called to initialize the operation, then autoKeyData(String) is called repeatedly to input more data to the key generation function. When all of the data has been added, then autoKeyFinish() must be called to complete key generation. This feature can be used to produce a key that is uniquely derived from the input data, and can be used to verify that the data has not changed.

public void autoKeyData( String)

Function:

Adds data to the auto key generation function.

Argument:

String

Return Value:

void

Remarks:

Used with autoKeyStart() and autoKeyFinish() methods.

public Image sigImage()

Returns a Java buffered Image of the current signature, rendered using the Image properties described in the appropriate section.

public void clearTablet()

Function:

Causes the signature to clear from the object.

Argument:

none.

Return Value:

none

Remarks:

public Boolean exportSigFile (string filename)

Function:

The control will write out a signature file in the Topaz image-free raw tablet data vector file format (.sig extension).

Argument:

FileName is a string, containing the path and filename to write to.

Return Value:

bool TRUE if successful, FALSE if not successful

Remarks:

The full Filename must be provided, including the .sig extension.

public Boolean importSigFile (String filename)

Function:

The control will Clear the current signature, read in a signature file in the Topaz vector file format, and display it.

Argument:

FileName is a string, containing the path and filename to read from.

Return Value:

bool TRUE if successful, FALSE if not successful

Remarks: The full Filename must be provided including the .sig extension.



public String getSigPlusVersion ()

Function:

Returns the current version of the SigPlus JAR file as a String

Argument:

None.

Return Value:

String Current Version

public void setSaveSigInfo( boolean )

Function:

Enables/Disables the saving of TimeStamp and Annotate data in the signature

Argument:

BOOL Value for SaveSigInfo

TRUE The SigInfo will be saved (default)

FALSE The SigInfo will not be saved

Return Value:

none

public short getNumberOfStrokes()

Function: Returns the total number of strokes in the current signature. Can be used to detect if a signature is present, or not.

Argument:

none

Return Value:

short decimal value of number of strokes in the signature.

public short getNumPointsForStroke( StrokeNumber )

Function:

Returns the total number of points in the specified stroke.

Argument:

short StrokeNumber is the number of the stroke to inquire about. Ranges from 0 to NumberOfStrokes - 1

Return Value:

short decimal value of number of points in the stroke.

public short getPointXValue(short Strokeldx, short PointIdx)

Function:

Returns the X coordinate value for the specified point. The value is in LogicalTablet Coordinates.

Argument:

short Strokeldx, the index of the stroke for the point desired.

short PointIdx, the index of the point in the stroke.

Return Value:

short decimal value of the x coordinate for the point.

public short getPointYValue(short Strokeldx, short PointIdx)

Function:

Returns the Y coordinate value for the specified point. The value is in LogicalTablet Coordinates.

Argument:

short Strokeldx, the index of the stroke for the point desired.

short PointIdx, the index of the point in the stroke.

Return Value:

short decimal value of the y coordinate for the point.

public short getXExtent()

Function:

Returns the exact width in px of the current signature.

Argument:



none  
Return Value:  
short px value of the height of the current signature

public short getYExtent()

Function:  
Returns the exact height in px of the current signature.

Argument:

none  
Return Value:  
short px value of the width of the current signature



## **“PROPERTIES” FOR SIGPLUS JAVA BEAN.** **METHODS WITH GET/SET OPTIONS**

### General Properties:

### Implemented properties

**All of the implemented properties below have both get and set methods, except where noted. For example:**

## Property: String TabletModel

**includes:**

## setTabletModel(String TabletModel)

## getTabletModel()

### Tablet Setup Parameters

These parameters are associated with the tablet configuration. All of them are set when TabletModel below is set.

## String TabletModel

Default = "SignatureGem1X5"

Must have any of the following values:

"SignatureGem1X5"

"SignatureGemLCD1X5"

"SignatureGem4X5"

"ClipGem"

"ClipGem19200"

"ClipGemLGL"

"SigLite1X5"

"SignatureGemLCD4X3"

"MicroGem4X5"

"MicroGemLCD"

"TracGem"

"PaperGem"

Use "SignatureGem1 X5"

Use "SignatureGemLCD4X3New"

Currently not supported

Currently not supported

Currently not supported

Currently not supported

<b>int TabletFormat</b>	Currently not implemented
-------------------------	---------------------------

<b>int TabletLCDType</b>	Currently not implemented
--------------------------	---------------------------

Specifies LCD type and format, 0 for 240x128, 1 for 128x64

<b>int TabletLCDXSize</b>	Currently not implemented
---------------------------	---------------------------

X Size of LCD display, in pixels

<b>int TabletLCDYSize</b>	Currently not implemented
---------------------------	---------------------------

Y Size of LCD display, in pixels

<b>int TabletLCDXStart</b>	Currently not implemented
----------------------------	---------------------------

X Pos in logical tablet coordinates of first LCD pixel, on left side.

## General properties

Preceded by get/set prefix.

## String TimeStamp

Sets the TimeStamp string for the signature, can be any string

ASCII new line character (LF) will break the text into multiple lines. Note that the

TimeStamp string is left justified in the signature box. Limited to 512 characters.

**String Annotation**

ASCII new line character (LF) will break the text into multiple lines. Note that the Annotation string is right justified in the signature box. Limited to 512 characters.

**String SigString**

get = prefix to take the signature string out of the bean.  
set = prefix to put a signature into the bean.

**int TabletState**

Setting to 1 enables the bean to access the selected com port and access the tablet for signature capture, setting to 0 disables the tablet for capture.  
When read, TabletState indicates the capture state of the tablet.  
Only one tablet control can be active on a given serial port at any point in time.

**int SigCompressionMode**

Sets the current compression mode for signatures

Argument:

short integer value as follows:

- |     |  |
|-----|--|
| 0   | No compression (default)   |
| 1   | Lossless compression with compacted data format.   |
| 2-8 | Compression ratio of signature stored in .sig file<br>"2" = 1KB typ. "4" = 500 byte typ. "8" = 250 byte typ. |

Return Value:

none

Remarks:

When loading a signature, the compression mode must be set to the same value that was used when the .sig file was created.

**int Mode EncryptionMode**

Sets Encryption mode. This function is used to set the encryption mode used for signatures.

Argument:

- | short | Mode   |
|-------|--|
| 0     | Clear text mode  |
| 1     | 40-bit DES. If a longer key is set, only 40 bits used        |
| 2     | Higher security encryption mode                              |
| 3     | Same as Mode2, added feature of locking timestamp/annotation |

Return Value:

none

Remarks:

If Encryption mode  $\geq 2$ , the key can only be set if there are no points in the signature (ClearTablet). When the signature has points while in encryption mode 2, the encryption mode cannot be changed to another mode unless the points are cleared.

- int TabletLCDXStop** Currently not implemented  
X Pos in logical tablet coordinates of last LCD pixel, on right side.
- int TabletLCDYStart** Currently not implemented  
Y Pos in logical tablet coordinates of first LCD pixel, at top
- int TabletLCDYStop** Currently not implemented  
X Pos in logical tablet coordinates of last LCD pixel, on bottom.
- int TabletLogicalXSize** No setter  
Sets the Range of horizontal values to be used in representing signatures in tablet coordinates. This has no relation to the displayed, image file, or tablet sizes. Be default, this value is set to TabletXStop minus TabletXStart.
- int TabletLogicalYSize** No setter  
Sets the Range of vertical values to be used in representing signatures in tablet coordinates. This has no relation to the displayed, image file, or tablet sizes. Be default, this value is set to TabletYStop minus TabletYStart.
- int TabletXStart** Automatically adjusts LogicalXSize when set  
Sets the X position in tablet coordinates, of the upper left hand corner of the bean signature box. This is automatically set by using the TabletModel property.  
This property can be used to position the horizontal starting point of the signature box in tablet space to a particular spot on the tablet, in tablet coordinates which increase from left to right.
- int TabletXStop** Automatically adjusts LogicalXSize when set  
Sets the X position in tablet coordinates, of the lower right hand corner of the bean signature box. This is automatically set by using the TabletModel property.  
This property can be used to position the horizontal ending point of the signature box in tablet space to a particular spot on the tablet, in tablet coordinates which increase from left to right.
- int TabletYStart** Automatically adjusts LogicalYSize when set  
Sets the Y position in tablet coordinates, of the upper left hand corner of the bean signature box. This is automatically set by using the TabletModel property .  
This property can be used to position the vertical starting point of the signature box in tablet space to a particular spot on the tablet, in tablet coordinates which increase from top to bottom.
- int TabletYStop** Automatically adjusts LogicalYSize when set  
Sets the Y position in tablet coordinates, of the lower right hand corner of the bean signature box. This is automatically set by using the TabletModel property.  
This property can be used to position the vertical ending point of the signature box in tablet space to a particular spot on the tablet, in tablet coordinates which increase from top to bottom.
- int TabletTimingAdvance**  
Internal property associated with tablet model and set by the TabletModel property.
- int TabletFilterPoints**  
Internal property associated with tablet model and set by the TabletModel property.
- int TabletBaudRate**  
Internal property associated with tablet model and set by the TabletModel property.
- int TabletResolution**  
Internal property associated with tablet model and set by the TabletModel property.

### Tablet Connection and Configuration Properties

- int TabletRotation** Default = 0  
Sets the orientation for display of tablet data. The data in the sig representation is stored in the native tablet orientation. Allowed values are 0, 90, 180, 270
- String TabletComPort** Default = COM1  
Sets the COM port to use using a String. Tablet should be plugged into this port. The SigPlus™ Java bean does not lock up a port as is the case with mouse-type drivers. The port is only used when tablet is selected on. Take care to only set COM port when tablet state is OFF. For USB connection (be sure you are using version 2.06 or above) please use the String value "USB1". For HSB connection (be sure you are using version 2.12 or above) please use the String value "HID1"
- boolean TabletComTest** Default = False  
Sets Hardware check mode. When this mode is active, if Topaz tablet is plugged into selected COM port (or USB), TabletState can be set to 1. If tablet is not plugged in, TabletState cannot be set to 1. Used with code to determine if tablet is connected, or which port tablet is connected to.
- boolean TabletClippingMode** Default = False  
Sets mode where signature points are not reported if the points are drawn outside the XStart/Stop, and YStart/Stop window. When active, pen down writing outside the Start/Stop window will return zero points as the number of tablet points.
- boolean TabletInvisible** Currently not implemented  
Sets mode where control becomes invisible and does not draw visibly.

### Display properties

These properties affect the way the signature is drawn on the display

**int DisplayPenWidth** Default = 1  
Sets pen width for the displayed signature, in pixels.

**boolean DisplayTransparentMode** Default = False.  
Controls whether the signature is displayed on an opaque background, or transparently.

**int DisplayRotation** Default = 0.  
Sets mode allowing signature rotation in the control after capture. Does not save the .sig information rotated. Display rotation only. The rotation value is set by TabletRotation. Can be used to rotate signatures after capture, if the signature was accidentally taken in a rotation orientation during signature capture. Normally, to change the tablet orientation during capture, the TabletRotation property is used.

**int DisplayJustifyX** Default = 0.  
Sets Justification X coordinate in Logical Tablet coordinates. When Display JustifyMode = 1-5, sets X border (left and right) distance.

**int DisplayJustifyY** Default = 0  
Sets Justification Y coordinate in Logical Tablet coordinates. When Display JustifyMode = 1-5, sets Y border (top and bottom) distance.

**int DisplayJustifyMode** Default = 0  
Sets Justification mode – how the signature is sized and positioned in the signature box

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-right corner)
4	Justify and zoom signature (lower-left corner)
5	Justify and zoom signature (center of control)

When using JustifyMode 1-5, a border area around the signature limits can be set in logical tablet coordinates using the JustifyX and JustifyY values.

**int DisplayTimeStampX** Default = 0  
Sets the Xposition, in pixels of the start of the Time Stamp in the signature box.  
If 0, then text is positioned 5% in from left side of signature box  
If not zero, then text is positioned the number of pixels specified, relative to the left edge of the signature box.

**int DisplayTimeStampY** Default = 0  
Sets the Y position, in pixels of the start of the Time Stamp in the signature box.  
If 0, then text is positioned 5% in from bottom of signature box  
If not zero, then text is positioned the number of pixels specified, relative to the top edge of the signature box.

**int DisplayTimeStampSize** Default = 0  
Sets the Y size, in pixels of the Time Stamp in the signature box.  
If 0, then text size is 7.5% of Y size of the signature box.  
If not zero, then text size is the value in pixels

- int DisplayAnnotationX** Default = 0  
Sets the X position, in pixels of the start of the Annotation String in the signature box.  
If 0, then text is positioned 5% in from right side of signature box  
If not zero, then text is positioned the number of pixels specified, relative to the right edge of the signature box.
- int DisplayAnnotationY** Default = 0  
Sets the Y position, in pixels of the start of the Annotation String in the signature box.  
If 0, then text is positioned 5% in from bottom side of signature box  
If not zero, then text is positioned the number of pixels specified, relative to the top edge of the signature box.
- int DisplayAnnotationSize** Default = 0  
Sets the Y size, in pixels of the Annotation start of the Time Stamp in the signature box.  
If 0, then text size is 7.5% of Y size of the signature box.  
If not zero, then text size is the value in pixels
- boolean DisplayTimeStamp** Default = False.  
Sets the bean to display the time stamp
- boolean DisplayAnnotation** Default = False.  
Sets the bean to display the annotation string.





### Image properties

These properties affect how the signature is drawn for the public `Image sigImage()` method. Many of the following properties share a related `Display`, which have the same functionality, but are applied to the image being generated rather than the display of the signature within the object.

<b>int ImageXSize</b>	Default = TabletLogicalXSize
Sets the number of X-pixels in the image provided by the public Image sigImage() method	

<b>int ImageYSize</b>	Default = TabletLogicalYSize
Sets the number of Y-pixels in the image provided by the public Image sigImage() method	

**float ImagePenWidth**  
Same as DisplayPenWidth, except applies to the public Image sigImage() method.

**boolean ImageTransparentMode**  
Same as DisplayTransparentMode, except applies to the public Image sigImage() method.

**int ImageRotation**  
Same as DisplayRotation, except applies to the public Image sigImage() method.

**int ImageJustifyX**  
Same as DisplayJustifyX, except applies to the public Image sigImage() method.

**int ImageJustifyY**  
Same as DisplayJustifyY, except applies to the public Image sigImage() method.

**int ImageJustifyMode**  
Same as DisplayJustifyMode, except applies to the public Image sigImage() method.

**int ImageTimeStampX**  
Same as DisplayTimeStampX, except applies to the public Image sigImage() method.

**int ImageTimeStampY**  
Same as DisplayTimeStampY, except applies to the public Image sigImage() method.

**int ImageTimeStampSize**  
Same as DisplayTimeStampSize, except applies to the public Image sigImage() method.

**int ImageAnnotationSize**  
Same as DisplayAnnotationSize, except applies to the public Image sigImage() method.

**int ImageAnnotationX**  
Same as DisplayAnnotationX, except applies to the public Image sigImage() method.

**int ImageAnnotationY**  
Same as DisplayAnnotationY, except applies to the public Image sigImage() method.

**boolean ImageTimeStamp**  
Same as DisplayTimeStamp, except applies to the public Image sigImage() method.

**boolean ImageAnnotation**  
Same as DisplayAnnotation, except applies to the public Image sigImage() method.