How-To Guide

SigPlus Java Image Demo

Table of Contents

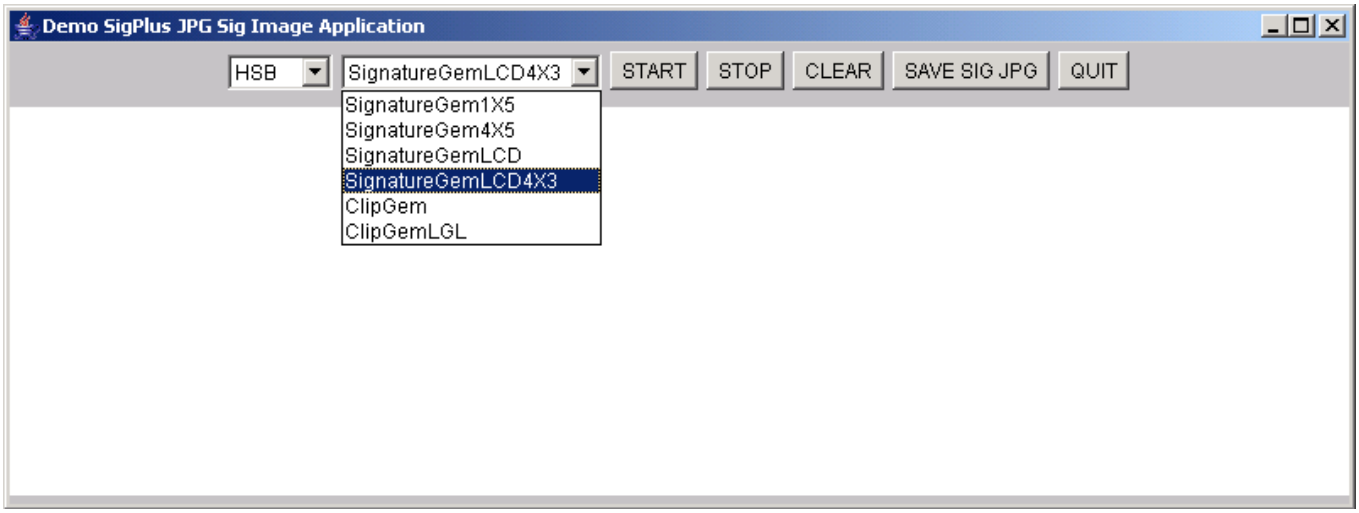
Overview

Welcome to the SigPlus Image Demo Guide. This will walk you through our demo "SigPlusImgDemo.java", available from the Topaz website.

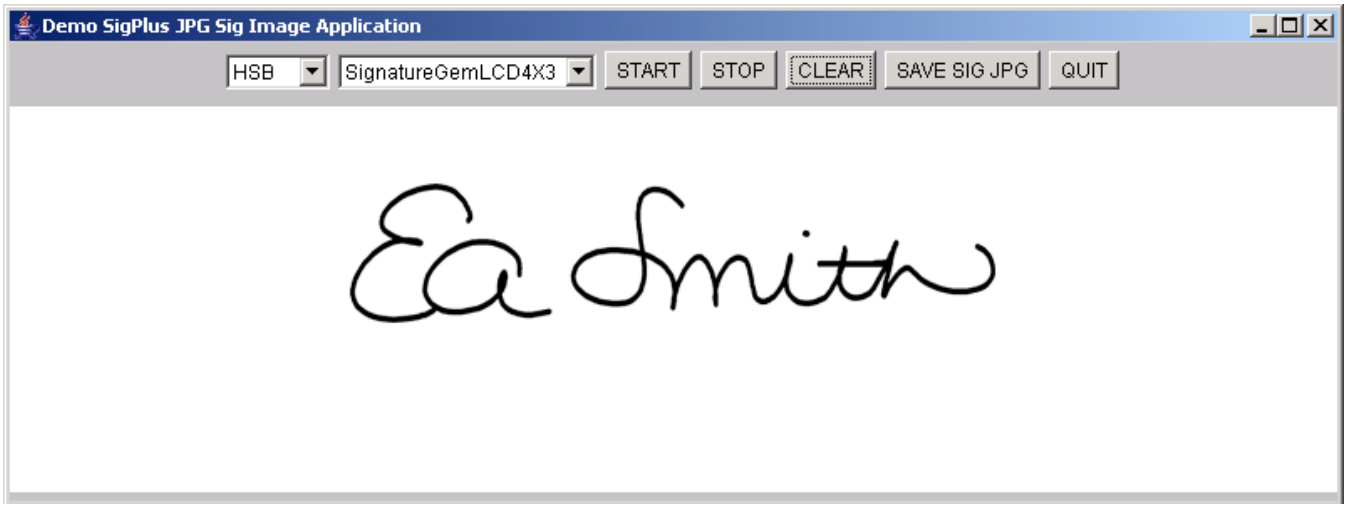
Download at: **www.topazsystems.com/sigplusprojava.html**

This application will show Java developers how to integrate SigPlus technology into their own applications, and how to export a signature image as an image file. Begin by opening "SigPlusImgDemo.java" in a Java runtime environment. Set the connection type to correspond with the connection type you are using. We used an HSB tablet for the purposes of this demo documentation.

Now, ensure the "Tablet Model" you are using corresponds to the tablet type selected by the demo application. For the purposes of this demo, we used a SignatureGem LCD 4X3 tablet.



After configuring the application for use with your signature capture device, click “START” to activate the tablet, and sign your name. Your signature will be displayed in the window. To erase your signature and start over, push “CLEAR”. However, if you do push “CLEAR”, be sure to push “START” again before resigning. If you would like to turn the pad off, push the “STOP” button. When you are ready to save, click “SAVE SIG JPG” to save your signature as a .JPG image file.



The JPG will be saved as “sig.jpg,” and will be stored in your Local Disk (C:). Of course, you can change to location and filename in your own application. Press “QUIT” when you are finished to exit the application.

The Code

Next, we will look over the Java code used to control this application. The code below creates a form for the “SigPlusImgDemo” using the gridbag class and the panel class. Buttons are defined to later call upon button events.

```
public static void main( String Args[] )
        {
        SigPlusImgDemo demo = new SigPlusImgDemo();
        demo.setSize(800,300);
        demo.setVisible(true);
demo.setBackground(Color.lightGray);
        }

public SigPlusImgDemo()
        {
GridBagLayout gbl = new GridBagLayout();
        GridBagConstraints gc = new GridBagConstraints();
        setLayout(gbl);
        Panel controlPanel = new Panel();
        setConstraints(controlPanel, gbl, gc, 0, 0,
        GridBagConstraints.REMAINDER, 1, 0, 0,
```

```
        GridBagConstraints.CENTER, GridBagConstraints.NONE,0, 0, 0, 0);
        add(controlPanel, gc);

controlPanel.add(connectionChoice);
controlPanel.add(connectionTablet);

Button startButton = new Button("START");
controlPanel.add(startButton);

Button stopButton = new Button("STOP");
controlPanel.add(stopButton);

Button clearButton = new Button("CLEAR");
controlPanel.add(clearButton);

Button saveJpgButton = new Button("SAVE SIG JPG");
controlPanel.add(saveJpgButton);

Button okButton = new Button("QUIT");
controlPanel.add(okButton);
```

The code below initializes the com API. This is necessary even with HSB pads because Java expects to see the com initialized. The com driver class is only necessary while in Windows. It then uses the ClassLoader class to create a new instance of SigPlus.

```
initConnection();
        String drivername = "com.sun.comm.Win32Driver";
        try
                {
                CommDriver driver = (CommDriver)

Class.forName(drivername).newInstance();

                driver.initialize();
                }
        catch (Throwable th)
                {
                /* Discard it */
                }
        try
                {
                ClassLoader cl =
                (com.topaz.sigplus.SigPlus.class).getClassLoader();
                sigObj = (SigPlus)Beans.instantiate( cl, "com.topaz.sigplus.SigPlus" );
        setConstraints(sigObj, gbl, gc, 0, 1,
        GridBagConstraints.REMAINDER, 1, 1, 1, GridBagConstraints.CENTER,
        GridBagConstraints.BOTH, 5, 0, 5, 0); add(sigObj, gc);
        sigObj.setSize(100,100);
sigObj.clearTablet();
        setTitle( "Demo SigPlus JPG Sig Image Application" );
```

When the “OK” button, which it titled as the “QUIT” button, is pushed, the tablet is turned off and the application is closed.

```
okButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                sigObj.setTabletState(0);
                        System.exit(0); } });
```

When the “START” button is pushed, the tablet is activated to accept signatures.

```
startButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                sigObj.setTabletState(0);
                sigObj.setTabletState(1);
        }
});
```

When the “STOP” button is pushed, the tablet is turned off and no longer accepts any signatures. However, unlike the “QUIT” button, the application still runs.

```
stopButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                sigObj.setTabletState(0);
        }
});
```

When the “CLEAR” button is pushed, the signature is cleared from the signature field.

```
clearButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                        System.out.println(sigObj.getKeyReceipt());
                sigObj.clearTablet();
        } });
```

The code below saves the signature to JPG format. First the tablet is turned off, and the image is formatted. Variable W and H are assigned respectively to the Width and Height of the image. The filepath is set to the “C:Sig1.jpg”, the JPEG is created, and it is then saved to the filepath.

```
saveJpgButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
        try {
                sigObj.setTabletState(0);
        sigObj.setImageJustifyMode(5);
        sigObj.setImagePenWidth(10);
        sigObj.setImageXSize(1000);
        sigObj.setImageYSize(350);
        BufferedImage sigImage = sigObj.sigImage();
        int w = sigImage.getWidth(null);
```

```
int h = sigImage.getHeight(null);
int[] pixels = new int[(w * h) * 2];
sigImage.setRGB(0, 0, 0, 0, pixels, 0, 0);
FileOutputStream fos = new
FileOutputStream("c:\\sig.jpg");
        JPEGImageEncoder jpeg =
        JPEGCodec.createJPEGEncoder(fos);
jpeg.encode(sigImage);
fos.close();

}
catch (Throwable th) {
        th.printStackTrace();
}

}
        });
```

This section of code sets the tablet model to match the selection from the drop down menu. If you use a SignatureGem LCD 4X3, it must be set to SignatureGem LCD 4X3 New. Otherwise, the choice matches the entry from the drop down list.

```
connectionTablet.addItemListener(new ItemListener(){
        public void itemStateChanged(ItemEvent e){
    if(connectionTablet.getSelectedItem() != "SignatureGemLCD4X3"){
sigObj.setTabletModel(connectionTablet.getSelectedItem());
        }
        else{
sigObj.setTabletModel("SignatureGemLCD4X3New"); //properly set up LCD4X3
        }
                }
        });
```

The following code sets the connection type to match the selection from the drop down menu. If you select HSB, it must be set to “HID1”. Otherwise, the choice matches the entry from the drop down list.

```
connectionChoice.addItemListener(new ItemListener(){
        public void itemStateChanged(ItemEvent e){
                if(connectionChoice.getSelectedItem() != "HSB"){
        sigObj.setTabletComPort(connectionChoice.getSelectedItem());
                }
                else{
        sigObj.setTabletComPort("HID1"); //properly set up HSB tablet
                }
        }
});
```

When you prompt the window to close, this ensures the tablet is off. There are three SigPlus event handlers in SigPlus that are currently unimplemented. These are not available for the developer to use.

```
addWindowListener( new WindowAdapter()
        {
        public void windowClosing( WindowEvent we )
                {
                sigObj.setTabletState( 0 );
                System.exit( 0 );
                }
        public void windowClosed( WindowEvent we )
                {
                System.exit( 0 );
                }
        } );
sigObj.addSigPlusListener( new SigPlusListener()
        {
public void handleTabletTimerEvent( SigPlusEvent0 evt )
                {
                }
public void handleNewTabletData( SigPlusEvent0 evt )
                {
                }
public void handleKeyPadData( SigPlusEvent0 evt )
                {
                }
        } );
```

This next section selects the defaults. A thread called eventThread is then created to run the application.

```
show();
        sigObj.setTabletModel("SignatureGem1X5");
        sigObj.setTabletComPort("COM1");
eventThread = new Thread(this);
eventThread.start();
        }
catch ( Exception e )
        {
        return;
        }
                }
public void run()
{
try
        {
        while ( true )
                {
                  Thread.sleep(100);
                 }
        }
catch (InterruptedException e)
```

```
        {
        }
    }
```

The code below creates and defines the items of the drop down menu for connection type choices.

```
        TextField txtPath = new TextField("C:\\test.sig", 30);

        Choice connectionChoice = new Choice();          protected
String[] connections =
            {
              "COM1",
              "COM2",
              "COM3",
              "COM4",
              "USB",
              "HSB",
            };
```

The code below creates and defines the items in the drop down menu for tablet type choices.

```
Choice connectionTablet = new Choice();                  protected String[] tablets =
            {
                    "SignatureGem1X5",
                    "SignatureGem4X5",
                    "SignatureGemLCD",
                    "SignatureGemLCD4X3", "ClipGem",
                    "ClipGemLGL",

            };
```

The following code specifies how many choices there will be for the menus above.

```
        private void initConnection()
{
        for(int i = 0; i < connections.length; i++)
{
                connectionChoice.add(connections[i]);
            }
            for(int i = 0; i < tablets.length; i++)
            {
                connectionTablet.add(tablets[i]);
            }
}
```

Because this demo will not create a biometric or an encrypted signature, but instead creates a .jpeg image, the document is not legally binding. This is a demo only and should be used as a blueprint for creating your own applications.

Dependencies and Files Necessary to Run This Demo

Jar files must be included in the CLASSPATH

1. SigPlus2_xx.jar (at the creation of this document the current version was 2_45)
2. Comm.jar (This file will vary depending on OS, you will need the correct java com API for your OS.)

Additional Files

1. (Win) – Win32com.dll must reside in the System32 folder
2. (HSB in Win) – SigUsb.dll must reside in the System32 folder