The Java Developer's Guide to

Asprise JSANE

Version 2.1

Single Developer Version

Document Version: 2.1

[Last updated on June 1, 2004]

http://www.asprise.com/product/jsane

Table of Contents

1. INTRODUCTION

About SANE

SANE is the de facto standard to access scanners/cameras on Mac, Linux, Unix platforms.

“SANE stands for "Scanner Access Now Easy" and is an application programming interface (API) that provides standardized access to any raster image scanner hardware (flatbed scanner, hand-held scanner, video- and still-cameras, frame-grabbers, etc.). The SANE API is public domain and its discussion and development is open to everybody. The current source code is written for UNIX (including GNU/Linux) and is available under the GNU General Public License (the SANE API is available to proprietary applications and backends as well, however)." visit: http://www.sane-project.org for more details.

About JSANE

JSANE provides SANE API in Java. JSANE enables Java developers to acquire images from scanners and digital cameras easily. Its universal APIs bridge Java and scanners, digital cameras tightly. With more than five years extensive development, LAB Asprise! proudly presents you the long waiting version 2 of JSANE.

Components of JSANE

JSANE is written in 100% pure Java.

Package com.asprise.util.jsane contains all the classes.

JSANE SDK Installation

First, make sure that you have already installed Java runtime version 1.2 or above on your system.

Download a copy of JSANE installation file from http://www.asprise.com/product/jsane. Unzip the installation file.

File Organization

The file organization of JSANE SDK distribution is as follows:

JSANE_HOME
+--- **DevelopersGuide-JSANE.pdf** [Developer's Guide]
+--- **doc** [Java docs]

+--- **jsane.jar** [Contains all JSANE classes]
+--- **demo-src.jar** [Contains the source code for all the demo programs]

+--- **demo.sh** [Launches JSANE demo programs]

+--- **LICENSE-EVALUATION-JSANE.txt**[License agreement]
+--- **Purchase-JSANE.htm** [Click to order JSANE]

Development Environment Setup

After you have installed JSANE, you need to setup your development environment in order to develop Java applications with JSANE. You only have to do one thing:

Put jsane.jar into your class path.

SANE Setup

JSANE makes use of SANE network daemon (**saned**). You need to start the SANE network daemon on the computer that the scanner is attached to (referred as SANE_SERVER). Then, you can access the scanner through JSANE on your computer (referred as SANE_CLIENT). This means, you can access scanners attached to other computers easily with JSANE. Of course, the SANE_SERVER and SANE_CLIENT can be the same computer.

For example, the code below acquires an image from the first device on host 198.98.10.1:

```
1.JSane sane = new JSane("198.98.10.1");
2.JSaneDevice[] devices = sane.getAllDevices();
3.Image image = devices[0].acquireImage();
```

To set up the SANE net daemon on the SANE_SERVER:

1) Check whether SANE is installed on SANE_SERVER by typing: **whereis saned**
2) If it not installed, you need obtain an installation file http://www.sane-project.org/source.html install it on SANE_SERVER.
3) Untar the file: **tar xzvf sane-backends-1.x.xx.tar.gz**

4) **cd sane-backends-1.x.xx**

5) ./configure

6) make

7) make install

8) Now you have SANE installed.

9) Run **scanimage -L** on SANE_SERVER to see whether your scanner/camera is detected.

10) You may need to edit the access configuration file: **/etc/sane.d/net.config** or **/usr/local/etc/sane.d/net.config** (either one, but you should not have both). Put the IP address of the SANE_CLIENT into it.

11) Run **saned -d128** to start the daemon

12) Run your Java programs based on JSANE from the SANE_CLIENT machine.

13) Once a session is terminated, the **saned** program exits. You might set saned as a service. Here is a good guide: http://www.penguin-breeder.org/sane/saned

Compatibility

Operating Systems:	**All platforms**
Java Runtime:	**Version 1.2 or above**.

2. IMAGE ACQUISITION WITH JSANE

There are two ways you can acquire images with JSANE. You can either acquire images through a UI component JSaneDialog or you can program the JSane class and JSaneDevice class directly. The former approach is much easier than the latter one. However, if flexibility is important you should consider the latter approach.

NOTE: We strongly encourage you to purchase the ADVANCED DEVERLOPER license or FULL SOURCE SITE license version, which exposes all the API functions – maximum flexibility.

Acquiring Images with JSaneDialog

The following code demonstrates the basic usage of JSane:

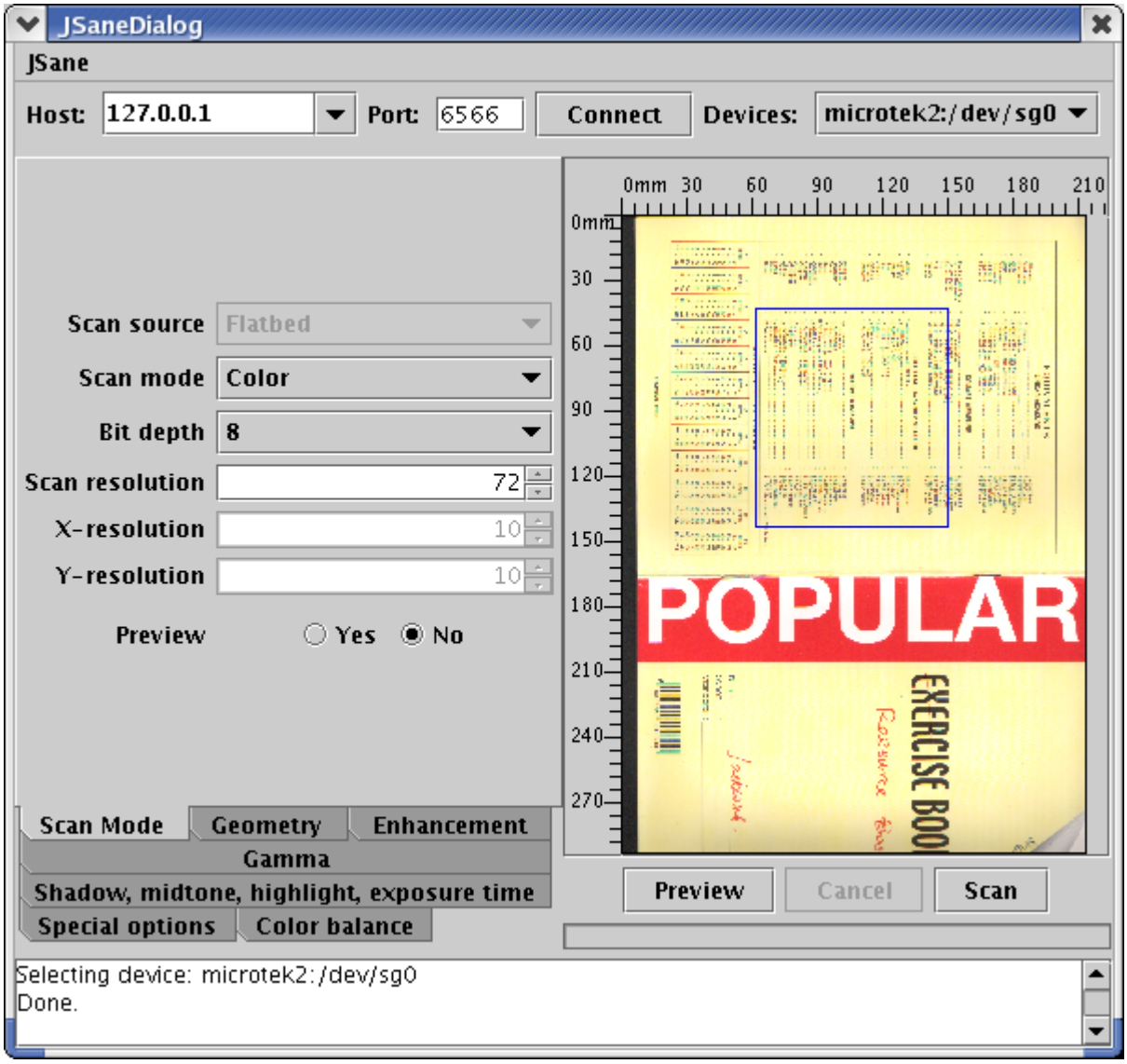
```
1.  JSaneDialog dialog = new JsaneDialog(
2.                              JSaneDialog.CP_START_SANED_LOCALHOST,
3.                              frame, "JSaneDialog", true, null);
4.  Image image = dialog.openDialog();
```

Line 1: Opens the dialog modally. The user can use the dialog to acquire image. The initial connection policy **JSaneDialog.CP_START_SANED_LOCALHOST** means that when the dialog initializes, JSane try to connection to the daemon on the localhost. If the daemon is not on, JSane try to activate with the command JsaneDialog.JSANECommand (default value is "**saned -d0**", you can modify it before creating JSaneDialog objects. If you have two sane installations on the same machine, you definitely need to replace the saned with its absolute path.).

Complete list of initial connection policies are:

- ***CP_DONOT_CONNECT*** - No initial connection;
- ***CP_LOCALHOST*** - Connect to locathost(default port) daemon when the dialog initializes.
- ***CP_START_SANED_LOCALHOST*** - Connect to locathost(default port) daemon when the dialog initializes. If the daemon is not started, JSane tries to start it.
- ***CP_FIRST_ENTRY*** - Connect to the first auto-saved host entry. If there is no such entry, the localhost is used.

Line 4: The **openDialog()** method will not return until an image is acquired or the user cancels the acquisition. The dialog interface is shown in the figure below.



The image acquiring process takes long time. However, you do not have to create new threads to execute the above function, because JSaneDialog uses a separate thread to loading images.

For a complete sample application, please refer to *JSaneDialogSampleApp.java*.

What if you want the JSaneDialog to be modeless and you need acquire more than one images with the same dialog? Fortunately, JSaneDialog allows you to open it modelessly and specify the image acquisition behavior with a JSaneDialogListener. For example, the code below acquires two images with a dialog:

```
1.  int imageCount = 0;
2.  JSaneDialogListener listener = new JSaneDialogListener() {
3.     public boolean imageAcquired(Image image) {
4.          if(image != null)
5.             imageCount ++;
6.
```

```
7.          // uses the image object here ..
8.
9.         if(imageCount == 2)
10.            return true;
11.         else
12.            return false;
13.     }
14.
15.      public void dialogDispose() {
16.      }
17.   };
18.
19.   JSaneDialog dialog = new JsaneDialog(JSaneDialog.CP_DONOT_CONNECT,
20.                                frame, "JSaneDialog", false, null)
21.   dialog.setListener(listener);
22.   dialog.pack();
23.   dialog.show();
```

Lines 4 – 19: Implements a JSaneDialogListener. The **imageAcquired** method is invoked when an image is acquired. If you want more images to be acquired, you return *false*. Otherwise, you return *true* to indicate that no more images are expected and the JSaneDialog should be disposed. The **dialogDispose** method is called when the JSaneDialog is disposed.

Line 21: Creates the JSaneDialog

Line 23: Sets the dialog listener with the listener we created in Lines 4 – 19.

Acquiring Images with JSaneDevice

First, you need a valid JSaneDevice object from a JSane instance. This is the only way to obtain a JSaneDevice.

For example, the code below selects the first device on the sane daemon host:

```
1.JSane sane = new JSane("198.98.10.1");
2.JSaneDevice[] devices = sane.getAllDevices();
3.Image image = devices[0].acquireImage();
```

Line 1: Opens the connection to host 198.98.10.1.
Line 2: Queries all the devices available on the sane daemon host
Line 3: Selects the first device

Alternatively, you can use the **getDeviceByName** function of the JSane class to obtain a JSaneDevice object through the device name:

public JSaneDevice getDeviceByName(java.lang.String name)

There are two modes to acquire images: block mode and non-block mode.

The acquireImage() function of the JSaneDevice class can be used in block mode:

public Image acquireImage()

The **acquireImage()** function does not return until an image is acquired or an error occurs. You might need to use a new thread when calling this function.

The **acquireImage(ImageObserver observer)** function of the JSaneDevice class should be used if non-block mode is desired.

public void acquireImage(java.awt.image.ImageObserver observer)

The **acquireImage(ImageObserver observer)** function returns immediately. The image is acquired through a new thread. Once the image is fully acquired, the specified image observer will be notified. So you should NOT put this in a separate thread.

The following is a sample implementation of an ImageObserver:

```
1.public class MyObserver implements ImageObserver {
2.
3.         ...
4.
5.         public boolean imageUpdate(
6.                   final Image img,
7.                   int infoflags,
8.                   int x,
9.                   int y,
10.                  int w,
11.                  int h) {
12.
13.                  if ((infoflags & ImageObserver.ERROR) != 0) {
14.                            // failed to acquire the image.
15.
16.                  } else {
17.                            // the image is acquired successfully.
18.                            // you can use it now.
19.                  }
20.        }
21.}
```

To cancel the current image acquisition process, you can call the **cancelAcquisition** method:

public void cancelAcquisition()

The ADVANCED license and SITE SOURCE license enable you to monitor the progress of the image acquisition process and provide many other features.

Setting and Getting Control Options

The Single Developer license version provides very basic control option operations. You can get and set value for a control option with the following functions of the JSaneDevice:

public java.lang.String getOptionValue(java.lang.String optionName)

public void setOptionValue(java.lang.String optionName, java.lang.String value)

For example, the following code set the scan resolution:

device.setOptionValue("resolution", "300"); // sets the resolution to 300.

Well know options are:

Option name	Remarks
resolution	The resolution at which an image should be acquired.
tl-x ***tl-y*** ***br-x*** ***br-y***	These four options define the scan area. *tl-x* specifies the top-left x coordinate value; *tl-y* specifies the top-left y coordinate value; *br-x* specifies the bottom right x coordinate value; *br-y* specifies the bottom right y coordinate value.

The ADVANCED license version and FULL SOURCE SITE license version allow you to query all the available control options and their valid values, and provide very sophisticated control option settings.

Automatic Document Feeding (ADF)

You need an ADF enabled scanner and JSANE ADVANCED license or FULL SOURCE SITE license.

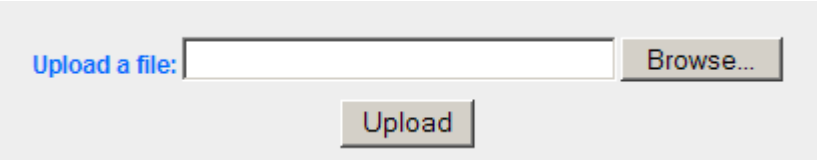
Uploading Acquired Images to Web Servers

This feature is available on JTwain. However, if you are our JSANE ADVANCED licensee or FULL SOURCE licensee , we will provide this feature at no charge if you need it.

For the demo of JTwain acquisition and uploading in action, please visit:

http://asprise.com/product/jtwain/applet/

A typical uploading web page is shown below:



The user browses a file and clicks the upload button to upload the selected file. The HTML code behind it is:

```
1.<form method='post' enctype='multipart/form-data'
2.        action='/product/jtwain/applet/fileupload.php?method=upload'>
3.
4.        <input type='file' name='file[]' style='width: 300'>
5.        <input type='hidden' name='testParam' value='testValue'>
6.        <input type='submit' value='Upload'></td>
7.
8.</form>
```

With JTwain, you can use the code below to perform the exact same uploading action:

```
1. // Acquire image first
2. ...
3. File acquiredImage = source.saveLastAcquiredImageIntoTemporaryFile();
4.
5.FileUploader fileUploader = new FileUploader();
6.
7.// fileUploader.setProxyHost(proxyHost);
8.// fileUploader.setProxyPort(port);
9.
10.Properties extraParameters = new Properties();
11.extraParameters = new Properties();
12.extraParameters.put("testParam", "testValue");
13.
14.fileUploader.upload(
15. "http://asprise.com/product/jtwain/applet/fileupload.php?method=upload"
  ,
16. "file[]",
17. "scanned.jpg", acquiredImage, extraParameters);
18. // If no exception thrown, upload succeeds.
```

About the Web Demo

On the first tab "JTwain Configuration", you can check your JTwain dll version or install the JTwain dll file to your system.

If the JTwain version is fine, you can proceed to the second tab to acquired images;

After an image has been acquired, you can use the third tab to upload it to your web server.

Specify your extra parameters if any in the text field next to "Extra Params" in the following format:

```
param1=value1;param2=value2;param3=value3
```

If your have purchased a license from us, you can request a full copy of the source code of the applet at no cost.

3. LOW LEVEL API PROGRAMMING

FOR ADVANCED LICENSE VERSION AND FULL SOURCE LICENSE VERSION ONLY.

4. IMAGE ACQUISITION UI COMPONENTS

**** Note: This is an optional feature. You need to purchase a separate license before using it.*

JImageDialog

JImageDialog is an image acquisition UI component that allows the user to load images and to perform basic image editing tasks. If you are developing some applications that require the user to select/edit/input images, then *JImageDialog* will make your life extremely easy – and more importantly, the user experience will be improved dramatically.

Let say you want to build an album application, the user is required to supply photos(ie. images). You put a button on your panel. When the user click the button, *JImageDialog* is brought up – now the user can select existing pictures files from his or her computer or acquire images from digital cameras or scanners. And the user can edit images before putting it into the album.

The following figure is the screen snapshot of *JImageDialog*.

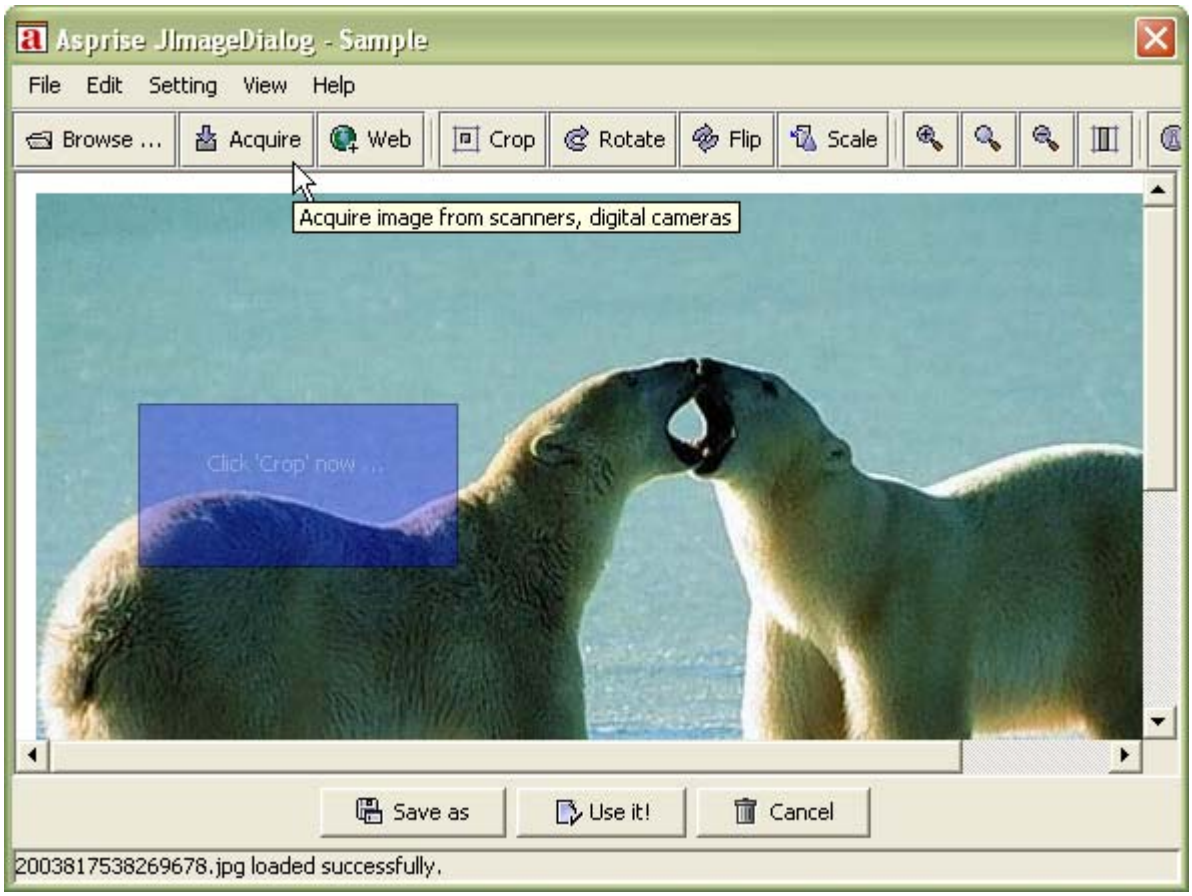


Illustration 1 JImageDialog

Advantages

- Multiple image sources supported: local computer, digital cameras, scanners and the web
- Multiple image formats: read and write BMP, PNG, JPG, GIF, PCT, PSD and many other formats
- Platform/Virtual machine independent: Any platform, any Java virtual machine (version 1.3 or above)
- Powerful features: rotation, flipping, scaling, clipping, etc.
- User friendly as well as developer friendly

The user can load images from local computer or the web, he or she can also acquire images from digital cammeras and scanners. After the image has been loaded, the user can rotate, clip, flip, and scale the image. The image has been loaded and edited, the user can save the image or select the image - which will be used in your applications.

Sample Uses

1. Modal (synchronous) mode

```
1. JImageDialog dialog = new JImageDialog(frame, "Sample", true); // Modal
  dialog
2. BufferedImage image = dialog.showDialog();
3. ...
```

Line 1 constructs the image dialog.

Line 2 brings up the image dialog and waiting for user's selection/acquisition.

Besides using JImageDialog in synchronous mode, you can also use it in:

2. Asynchronous mode

```
1. public class JImageDialogSample extends JPanel implements
  JImageDialogListener {
2.          ...
3.   BufferedImage image;
4.
5.   // Displays selected image if any.
6.   public void paintComponent(Graphics g) {
7.       super.paintComponent(g); // Paint background.
8.       if(image != null)
9.           g.drawImage(image, 0, 0, null);
10.   }
11.
12.   // Sets image and refreshes the panel.
13.         public void setImage(BufferedImage image) {
14.       this.image = image;
15.       setPreferredSize(getPreferredSize());
16.       revalidate();
17.       repaint();
18.   }
19.
20.   // Methods in JImageDialogListener
21.   // When the user presses cancel button, this method will be called.
22.   public void onCancel() {
23.       setImage(null);
24.   }
25.
26.   // When the user presses the selection button, will be invoked.
27.   public void onImageSet(BufferedImage image) {
28.       setImage(image);
29.   }
30. }
```

```
31.
32. ...
33. JImageDialogSample imagePanel = new JImageDialogSample();
34.
35. JImageDialog dialog = new JImageDialog();
36. dialog.addImageDialogListener(imagePanel);
37. dialog.showDialog();
```

Line 1-30 implements a JimageDialogListener.

Line 33 constructs the listener.

Line 35 constructs the dialog.

Line 36 registers the listener the the dialog

Line 37 brings up the dialog

When the user acquires an image and selects it, JimageDialog's listeners will be notified. In this case, *imagePanel.onImageSet(BufferedImage image)* will be called and thus the panel will display the selected image. If the user cancels the selection, *onCancel()* will be called instead.

Sample application: *com.asprise.util.ui.JImageDialogSample*

Supported Image Formats

The following table shows image formats supported by *JImageDialog:*

Formats	File extensions	READ	WRITE
Adobe Photoshop	*.psd	Y	Y
Bitmap, Windows/OS2	*.bmp, *.dib	Y	Y
Cursor	*.cur	Y	
Graphics Interchange Format	*.gif	Y	
Icon	*.ico	Y	
JPEG	*.jpg, *.jpeg	Y	Y
Macintosh PICT Format	*.pict, *.pct	Y	Y
PCX Format	*.pcx	Y	Y
Portable Network Graphics	*.png	Y	Y
Sun Raster Format	*.ras	Y	
Tag Image File Format	*.tif, *tiff	Y	

Formats	File extensions	READ	WRITE
Targa	*.tga	Y	Y
X Bitmap	*.xbm	Y	Y
X PixMap	*.xpm	Y	Y

On any Java platforms (version 1.3 or above), *JImageDialog* supports the above formats (using its own library to read/write image files). *JImageDialog* intelligently selects the best way to read or write files – eg. on Java 1.4, it may invoke *ImageIO* to see whether a file can be read or writen; if the *ImageIO* can do the job then *JImageDialog* will let it do; otherwise, *JImageDialog* will use its own library to access the file.

Note: You can only read/write image files from the *JImageDialog* UI component with unlicensed image acquisition UI component package. If you want to access image files from your Java code and/or to perform other advanced operations, you need to obtain an affordable license from LAB Asprise!.

Compatibility

All operating systems;

All Java runtimes with **version 1.3 or above**.

Software Packaging and Distribution

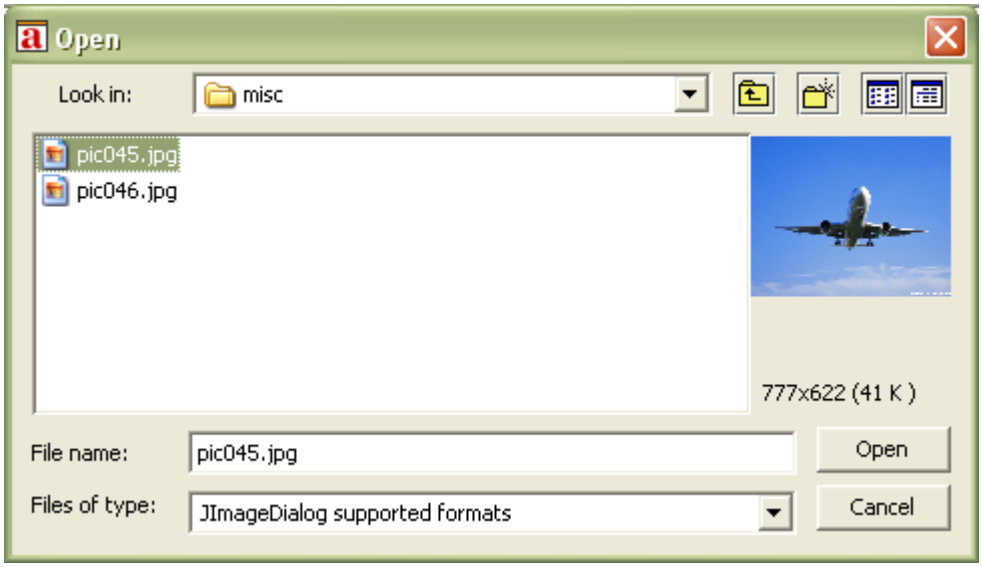
Mandatory: **jid.jar, JTwain.jar**

Optional: **ApsriseJTwain.dll** [supports digital cameras and scanners]

JImageFileChooser

An extended JFileChooser that supports image preview and image information extraction.

When the user clicks an image file, its preview and associated information will be displayed to assist the user to select the proper image.



Sample Use:

```
1.JFileChooser fc = new JImageFileChooser(lastDirectory);
2.fc.addChoosableFileFilter(JImageFileChooser.getImageFileFilter());
3.int returnVal = fc.showOpenDialog(frame);
  ...
```

Line 1 creates the image file chooser;

Line 2 set the file filter.

You can use it as normal JFileChooser, although it improves the user experience greatly.

Supported Image Formats

Please refer to *Supported Image Formats* in *JImageDialog* section.

Note: You can only preview image files from the *JImageFileChooser* UI component with unlicensed image acquisition UI component package. If you want to read/write image files from your Java code with the package and/or to perform other advanced operations, you need to obtain an affordable license from LAB Asprise!.

Compatibility

All operating systems;

All Java runtimes with **version 1.2 or above**.

Software Packaging and Distribution

Mandatory: **jid.jar**

5. SUPPORT AND PROFESSIONAL SERVICES

Support Web Site

http://www.asprise.com/product/jsane

Basic Support

Our team provides basic support for general Asprise JSANE developers. Email your technical questions to support@asprise.com (Our team may reject your improper enquiries without any reply. To avoid this, here is a little piece of:)

Advice: You are strongly recommended to subscribe our premium support service in order to get your problems sloved quickly.

Premium Support Services

Free one year premium support services subscription with every license purchased. You may optionally extend premium support services after your subscription expires. More details will be included in the email sent to you when you purchase licenses.

Professional Services

Our team are ready to help you to develop various applications, components. Please send your query to info@asprise.com

# # #