

**The Java Developer's Guide to**

# **Asprise JTwain**

**Version 8.2**

Document Version: 8.2

[Last updated on February 3, 2004]

All Rights Reserved. LAB **Asprise!** © 1998-2004.

<http://www.asprise.com/product/jtwain>

# Table of Contents

<b>1. INTRODUCTION.....</b>	<b>3</b>
About TWAIN.....	3
About JTwain.....	3
Components of JTwain.....	3
JTwain SDK Installation.....	4
File Organization .....	4
Development Environment Setup.....	4
Compatibility.....	5
<b>2. IMAGE ACQUISITION WITH JTWAIN.....</b>	<b>6</b>
For the Impatient.....	6
Control Flow of a Typical Image Acquisition Process.....	7
Getting a Source .....	7
Hiding the User Interface.....	9
Setting and Getting Source Capabilities.....	9
Acquiring Images.....	10
Automatic Document Feeding (ADF).....	11
Saving Acquired Images into Files.....	11
Uploading Acquired Images to Web Servers.....	13
About the JTwain Web Demo.....	14
<b>3. LOW LEVEL API PROGRAMMING.....</b>	<b>15</b>
JTwain API Model.....	15
Extending the Source.....	15
TWAIN & JTwain Mapping.....	17
<b>4. ADVANCED TOPICS.....</b>	<b>18</b>
Exception Handling.....	18
Using JTwain in Threads.....	19
Software Packaging and Distribution.....	19
<b>5. IMAGE ACQUISITION UI COMPONENTS.....</b>	<b>20</b>
JImageDialog.....	20
JImageFileChooser.....	24
<b>6. SUPPORT AND PROFESSIONAL SERVICES.....</b>	<b>26</b>
Support Web Site.....	26
Basic Support.....	26
Premium Support Services.....	26
Professional Services.....	26

# 1. INTRODUCTION

## About TWAIN

The TWAIN initiative was originally launched in 1992 by leading industry vendors who recognized a need for a standard software protocol and applications programming interface (API) that regulates communication between software applications and imaging devices (the source of the data). TWAIN defines that standard. Most of scanners and digital cameras on market are TWAIN compatible. For more information, visit: [www.twain.org](http://www.twain.org)

## About JTwain

JTwain is the Java counterpart of TWAIN. It is a TWAIN suite developed by LAB Asprise! since 1998. It is 100% TWAIN (most updated version 1.9) compatible. JTwain 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 8.1 of JTwain.

## Components of JTwain

JTwain comprises two components:

- A native library: **AspriseJTwain.dll**
- Several Java packages:
  - \* **com.asprise.util.jtwain** - main package; contains essential classes to perform image acquisition
  - \* **com.asprise.util.jtwain.lowlevel** – low-level APIs for advanced development of TWAIN applications
  - \* **com.asprise.util.ui** – optional UI components
  - \* **com.asprise.util.jtwain.web** – classes for uploading images to web servers

## JTwain SDK Installation

First, make sure that you have already installed Java runtime version 1.2 or above on your system. Currently, JTwain only support the following OSs: Windows 98, NT, ME, 2000, XP and all Windows Server platforms. Other OSs are planed.

Download a copy of JTwain installation file from <http://www.asprise.com/product/jtwain>. Double click the installation file to launch the installation process. Then follow the instruction to install it.

After the installation, double click *TestJTwain.bat* to test your installation. [Make sure *java* is in the path]. A proper installation results no ERROR messages.

## File Organization

The file organization of JTwain SDK distribution is as follows:

### JTWAIN\_HOME

- +--- **DevelopersGuide.pdf** [Developer's Guide]
- +--- **doc** [Java docs]
  
- +--- **AspriseJTwain.dll** [The sole native library]
- +--- **JTwain.jar** [Contains all JTwain classes]
- +--- **demo.jar** [Contains binary class files and source code for demo programs]
- +--- **demo-src.jar** [Contains the source code for all the demo programs]
- +--- **jid.jar** [Contains optional UI components]
  
- +--- **LaunchDemo.bat** [Launches JTwain demo programs]
- +--- **UI.bat** [Launches UI components Demo]
- +--- **applet.html** [Launches the JTwain Web Demo applet]
  
- +--- **LICENSE-EVALUATION.txt** [License agreement]
- +--- **Purchase.htm** [Click to order JTwain]

## Development Environment Setup

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

**Put JTwain.jar into your class path.**

If you only want to see JTwain demos, then you do not have to perform this step.

For more information, please refer the '*Software Distribution*' section.

## Compatibility

Operating Systems:      **All Windows platforms** are currently supported; other OSs planned.  
Java Runtime:            **Version 1.2 or above.**

## 2. IMAGE ACQUISITION WITH JTWAIN

### For the Impatient

The following code demonstrates the basic usage of JTwain:

```
1.  try {
2.      Source source = SourceManager.instance().getDefaultSource();
3.      source.open();
4.      Image image = source.acquireImage();
5.      ... // Uses image here ...
6.  } catch (Exception e) {
7.      e.printStackTrace();
8.  } finally {
9.      SourceManager.closeSourceManager();
10. }
```

Line 2: *SourceManager* represents TWAIN source manager, which manages and controls all the data(image) sources, eg. scanners, digital cameras, available on the operating system. There can be one and only one *SourceManager* at any time. A default *Source* is obtained by calling *SourceManager*'s *getDefaultSource* method.

Line 3: Opens the *Source*.

Line 4: Acquires an *Image* from the opened *Source*. Now, the *Image* has been acquired, and it can be used as other *Images* in your applications.

Line 9 closes any open *SourceManager* (which closes any open *Source*) regardless whether there are exceptions thrown.

To execute the above lines, you have to import: *com.asprise.util.jtwain.SourceManager* and *com.asprise.util.jtwain.Source*.

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

**Note:** You need at least one source to run the demo programs. If you do not have any sources, download and install the sample TWAIN Source provided by twain.org at:

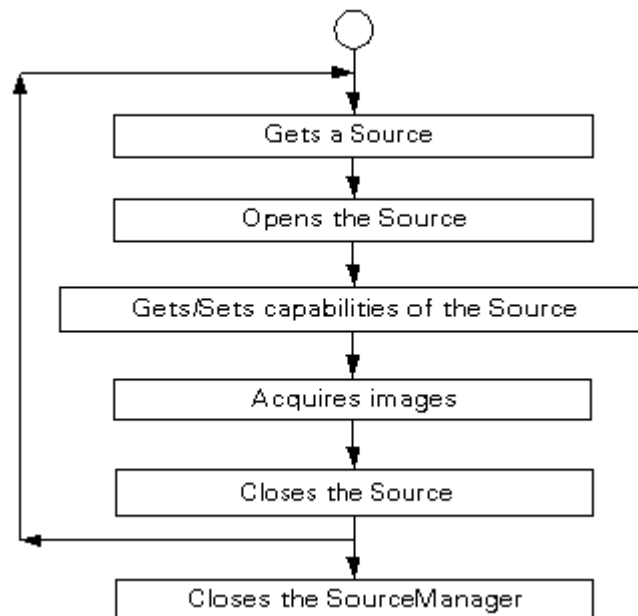
<http://www.twain.org/devfiles/twainkit.exe>

---

**TIP:** Source code for all demo programs are in the file *demo.jar*.

---

# Control Flow of a Typical Image Acquisition Process



*Illustration 1 - Control flow of image acquisition process*

Illustration 1 shows the control flow of a typical image acquisition process. In last section, basic code for image acquisition has been presented. You will notice that this flow clearly illustrates that code, except that:

- Getting and setting capabilities are not used – we will introduce this in later sections.
- In stead of explicitly closing the opened Source, SourceManager is closed – which causes any opened Source close.

## Getting a Source

Source, or data source, is an abstraction of an image source – which can be a scanner, a digital camera or an image database.

There are many ways to get a Source from the SourceManager:

### 1) Gets the default Source

```
1. Source source = SourceManager.instance().getDefaultSource();
```

SourceManager will return the default Source or null if no data source exists.

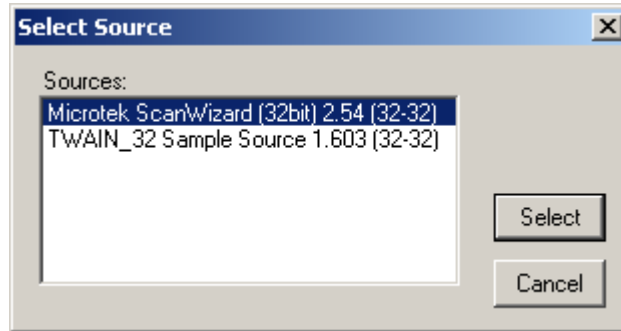
If you do not have any TWAIN compatible scanner or digital camera, you can try the sample

TWAIN Source provided by twain.org at: <http://www.twain.org/devfiles/twainkit.exe>

## 2) Lets the user select a Source

```
1. Source source = SourceManager.instance().selectSourceUI();
```

A dialog will pop up to enable the user to select a Source. Run *DemoSelectSourceDialog.bat* to see the demo. You will see the following pop up (the contents may be different):



## 3) Selects source by its name

```
1. Source source = SourceManager.instance().selectSource("Big Camera");
```

A Source will be returned if there is a Source with the specified name, null otherwise.

## 4) Get all the Sources available

```
1. Source sources[] = SourceManager.getAllSources();
```

An array containing all the data sources will be returned.

## Validating a Source:

```
1. Source validateSource = SourceManager.selectSource(source);
```

If the *source* is a valid Source, a new Source object representing the same data Source is returned possible with more detailed information; otherwise, null will be returned.



# Hiding the User Interface

## Hiding the “Select Source” UI:

Instead of using method (2) in *Getting a Source*, try to use other methods, like (1), (3), (4).

## Hiding the scanner/digital camera's acquisition UI:

Before performing image acquisition, a Source should be called on the following method:

```
1. if(source.isUIEnabled())  
2.     source.setUIEnabled(false);
```

## Hiding the indicators' UI:

Some Sources may pop up indicator dialogs to show the progress of image acquisition. To turn them off:

```
1. source.setIndicators(false);
```

The above line set the source's capability CAP\_INDICATORS to off. To learn more about capabilities, see next section.

Run *DemoHiddenUI.bat* to see hidden-UI image acquisition from the default Source.

# Setting and Getting Source Capabilities

One of TWAIN's benefits is it allows applications to easily interact with a variety of acquisition devices. Developers of applications need to be aware of a Source's capabilities and may influence the capabilities that the Source offers to the application's users. To do this, the application can perform capability negotiation. Some sample capabilities are: CAP\_XFERCOUNT – indicating number of images the application is willing to accept, ICAP\_XRESOLUTION - X-axis resolution of the Source.

When all the UIs have been hidden, setting and getting the Source's capabilities are extremely important. Setting and getting source capabilities should be done after the source has been opened and before performing the image acquisition. Thanks to JTWAIN, you do not have to go into dirty details of those capabilities. We have implemented proper operations for every capability that required by TWAIN 1.9 specification.

In last section, CAP\_INDICATORS has been set by calling method *setIndicators*. Here, we use

CAP\_XFERCOUNT as an example to explain setting and getting capabilities in details:

Method in Source	Remarks
<i>getTransferCount</i>	Return the Capability (CAP_XFERCOUNT)'s valid value(s) including current and default values. [Corresponding to TWAIN: MSG_GET]
<i>getCurrentTransferCount</i>	Get the Capability's current value. [Corresponding to TWAIN: MSG_GETCURRENT]
<i>getDefaultTransferCount</i>	Get the Capability's preferred default value (Source specific). Corresponding to TWAIN: MSG_GETDEFAULT]
<i>resetTransferCount</i>	Change a Capability's current value to its TWAIN-defined default. [Corresponding to TWAIN: MSG_RESET]
<i>setTransferCount</i>	Change a Capability's current and/or available value(s). [Corresponding to TWAIN: MSG_SET]

For a complete list of all capabilities, please refer the TWAIN specification at:

[http://www.twain.org/docs/Spec1\\_9\\_197.pdf](http://www.twain.org/docs/Spec1_9_197.pdf)

Demo program *DemoGetCapabilities* prints all the capabilities that the selected Source supports. Double click *DemoGetCapabilities.bat* to run it.

## Acquiring Images

Acquire an image using JTwain is as easy as:

```
1. java.awt.Image image = source.acquireImage();
```

To load the image completely:

```
2. MediaTracker tracker = new MediaTracker(this);
3. tracker.addImage(this.image, 1);
4.
5. try {
6.     tracker.waitForAll();
7. }catch(Exception e) {
8.     e.printStackTrace();
9. }
```

Then you can proceed to acquire another image. For a complete sample application, see *DemoSelectSourceDialog* – *DemoSelectSourceDialog.java* and *ImageDisplayer.java*.

## Automatic Document Feeding (ADF)

JTwain makes it easy to perform automatic document feeding. The following code illustrates an automatic document feeding process:

```
1. source.open();
2. source.setUIEnabled(true);
3.
4. int counter = 1;
5. do {
6.     Image image = source.acquireImage();
7.     // Uses image here ...
8. }while(source.hasMoreImages());
```

If the data Source supports ADF, the user can enable ADF and set number of documents intending to acquire. Run DemoADF.bat to see how ADF acquisition works.

To perform ADF without UI, replace the above code Line 2 with the following code:

```
1. source.setUIEnabled(false);
2. source.setFeederEnabled(true);
3. source.setAutoFeed(true);
4. source.setTransferCount(3);
```

Modify *DemoADF.java* to run ADF without UI.

## Saving Acquired Images into Files

From version 8.2, you can using the following built-in image saving functions:

```
public InputStream outputLastAcquiredImageAsJPEG()

public File saveLastAcquiredImageIntoTemporaryFile()

public void saveLastAcquiredImageIntoFile(String destination)

public void saveLastAcquiredImageIntoFile(File destination)
```

Sample code:

```
1. public class DemoSaveJPEG extends JTwainDemoCode{
2.
3.     public DemoSaveJPEG() {
4.
5.         try {
6.
7.             Source source = SourceManager.instance().getDefaultSource();
```

```

8.
9.     if(source == null) {
10.         error("There is no (default) source on the system!");
11.         return;
12.     }
13.
14.     source.open();
15.
16.     Image image = source.acquireImage();
17.
18.     ImageDisplayer imageDisplayer =
19.         new ImageDisplayer("DemoSimple", image);
20.
21.     FileDialog fileDialog = new FileDialog(imageDisplayer.getFrame(),
    "Save the image acquired into a file: ", FileDialog.SAVE);
22.     fileDialog.show();
23.
24.     source.saveLastAcquiredImageIntoFile(new
25.         File(fileDialog.getDirectory(), fileDialog.getFile()));
26.
27.     source.close();
28.
29.     }catch(Exception e) {
30.         exception(e);
31.     }finally{
32.         SourceManager.closeSourceManager();
33.     }
34.}
35.
36. public static void main(String[] args) {
37.     new DemoSaveJPEG();
38. }
39.}

```

The code above can run on any JRE with version 1.2 and above. Additionally, if you are using Java version 1.4 or about, you can use the ImageIO class to write the acquired images into various formats. For example,

```

1.import java.awt.*;
2.import java.awt.image.*;
3.import java.io.*;
4.import javax.imageio.*;
5.
6.import com.asprise.util.jtwain.*;
7.
8....
9.
10.try {
11.
12.    Source source = SourceManager.instance().getDefaultSource();
13.    source.open();
14.
15.    BufferedImage image = source.acquireImage(); // Acquire the image
16.
17.    // Save the image as a PNG file
18.    ImageIO.write(image, "png", new File("C:\\test.png"));
19.
20.    // Save it as a JPEG file
21.    ImageIO.write(image, "jpg", new File("C:\\test.jpg"));
22.

```

```

23.}catch(Exception e) {
24.    e.printStackTrace();
25.}finally{
26.    SourceManager.closeSourceManager();
27.}

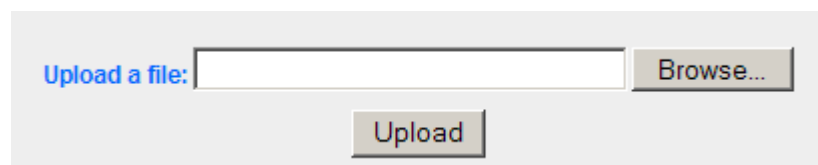
```

## Uploading Acquired Images to Web Servers

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 JTwain 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

### JTwain API Model

JTwain has provided all TWAIN specification v1.9 mandatory capabilities negotiation operations. Besides them, some useful optional capabilities negotiation operations have been implemented too. If you need to deal with some less frequently used capabilities or you prefer to have total control on capabilities negotiation, the good news that JTwain exposes low level APIs which you can use to accomplish your tasks.

### Extending the Source

Suppose you need to do ICAP\_BITDEPTHREDUCTION capability negotiation. (ICAP\_BITDEPTHREDUCTION is the capability controlling Reduction Method the Source should use to reduce the bit depth of the data.)

On page 9-425 of Twain specification v1.9:

#### Values

<i>Type:</i>	TW_UINT16	
<i>Default Value:</i>	No Default	
<i>Allowed Values:</i>	TWBR_THRESHOLD	0
	TWBR_HALFTONES	1
	TWBR_CUSTHALFTONE	2
	TWBR_DIFFUSION	3
<i>Container for MSG_GET:</i>	TW_ENUMERATION	
	TW_ONEVALUE	
<i>Container for MSG_SET:</i>	TW_ENUMERATION	
	TW_ONEVALUE	

Now, you can extend the Source class to add this capability handling.

```
1. public class MySource extends com.asprise.util.jtwain.Source  
2. {
```

```

3.  public MySource(Source source)
4.  {
5.      if(source == null ||
6.          (source.getSourceName() == null && source.getIdentity() == null))
7.          throw new IllegalArgumentException("source should not be null!");
8.
9.      this.sourceName = source.getSourceName();
10.     this.identity = source.getIdentity();
11. }
12.
13. // Get value(s).
14. public int[] getBitDepthReduction()
15.     throws InvalidStateException, OperationException
16. {
17.     Object ret = getCapability(MSG_GET, ICAP_BITDEPTHREDUCTION,
18.                               TWON_DONOTCARE16);
19.     return getIntArray(ret);
20. }
21.
22. // Set one value.
23. public void setBitDepthReduction(int value)
24.     throws InvalidStateException, OperationException
25. {
26.     ValueContainerOneValue c = new ValueContainerOneValue();
27.     ItemTypeInteger item = new ItemTypeInteger(TWTY_UINT16, value);
28.     c.setItem(item);
29.     setCapability(ICAP_BITDEPTHREDUCTION, c);
30. }
31.
32. // Set multiple values.
33. public void setBitDepthReduction(int[] values)
34.     throws InvalidStateException, OperationException
35. {
36.     if(values == null || values.length == 1)
37.         throw new IllegalArgumentException("Empty values!");
38.
39.     ValueContainerEnumeration c = new ValueContainerEnumeration();
40.     for(int i=0; i<values.length; i++)
41.         c.pushItem(new ItemTypeInteger(TWTY_UINT16, values[i]));
42.
43.     setCapability(ICAP_BITDEPTHREDUCTION, c);
44. }
45.
46. // Sample use.
47. public static void main(String args[]) throws Exception
48. {
49.     Source source = SourceManager.instance().selectSourceUI();
50.     source.open();
51.
52.     MySource mine = new MySource(source);
53.     mine.setBitDepthReduction(1);
54.     Image image = mine.acquireImage();
55.
56.     ...
57. }
58.

```

Line 17 *getCapability* returns the capability enquiry result. Some examples: if the container is of the type *ValueContainerOneValue* and item type is *TWTY\_UINT16*, then a Long will be



returned; For a single value container, the returned object may be of the following types: Long, Double, String in case of multi-value containers, the returned object may be of the following type: Long[], Double[], String[].

## TWAIN & JTwain Mapping

### Containers

<b><i>TWAIN</i></b>	<b><i>JTWAIN</i></b>
TW_ONEVALUE	ValueContainerOneValue
TW_ARRAY	ValueContainerArray
TW_ENUMERATION	ValueContainerEnumeration
TW_RANGE	ValueContainerRange

### Item Types

<b><i>TWAIN</i></b>	<b><i>JTWAIN</i></b>
TW_BOOL	ItemTypeInteger(TWTY_BOOL, <i>value[1 or 0]</i> )
TW_INT8	ItemTypeInteger(TWTY_INT8, <i>value</i> )
TW_INT16	ItemTypeInteger(TWTY_INT16, <i>value</i> )
TW_UINT32	ItemTypeInteger(TWTY_UINT32, <i>value</i> )
TW_FIX32	ItemTypeFix32( <i>value</i> )
TW_STR32	ItemTypeString(TWTY_STR32, <i>value</i> )
TW_STR1024	ItemTypeString(TWTY_STR1024, <i>value</i> )
TW_FRAME	ItemTypeFrame

## 4. ADVANCED TOPICS

### Exception Handling

According to TWAIN specification, a data Source does not have to support all the capabilities. Furthermore, there are some poorly designed TWAIN Sources which are not TWAIN-compatible. Exceptions could be thrown everywhere especially in the middle of getting/setting capabilities.

Asprise JTwain package has been carefully designed to relieve developers from handling complicated multiple exceptions. By default, a minimum set of exceptions will be thrown. To enable a Source to throw all possible exceptions, one can enable or disable this minimum exception option by executing the following code:

```
1. source.setMinimumExceptionEnabled(true); // Only a minimum set of
   exceptions will be thrown. By default, minimum exception is enabled.

2. source.setMinimumExceptionEnabled(false); // Throw as many as possible
   exceptions.
```

The “minimum set of exceptions” does not include exceptions thrown during capability getting/setting. Most of critical exceptions are included in this minimum exception set.

Note that although minimum exception option can be turned on, the developer still has to write code to catch exceptions – there could be some exceptions to be thrown, although just a minimum set.

A typical exception handling example:

```
1. try {
2.     Source source = SourceManager.instance().getDefaultSource();
3.     source.open();
4.     source.setXResolution(999); // Invalid value!
5.
6.     Image image = source.acquireImage();
7.
8.     new ImageDisplay("DemoSimple", image);
9.
10. } catch (Exception e) {
11.     e.printStackTrace();
12. } finally {
13.     SourceManager.closeSourceManager();
14. }
```

Note that line 4 set the capability ICAP\_XRESOLUTION to value 999 – very likely an invalid value (typical resolution values are 72, 300, 600, etc). Even if the Source does not support this

capability or it rejects the value setting request, no exception will be thrown – because minimum exception option is on by default. The image will be acquired using Source's default resolution.

If this capability is very important, one can turn off minimum exception option by inserting the following line before line 4:

```
|1. source.setMinimumExceptionEnabled(false)|
```

Browse any of the demo programs to see exception handling at work.

If the user cancels scanning during single image acquisition, Source's *acquireImage* will throw a *JTwainException* exception.

## Using JTwain in Threads

The TWAIN is not thread-safe, its Java counterpart, JTwain inherits the same property.

Once a Source has been opened, it has to be used in the same thread until it is closed. If a Source is used in application's main thread, a fatal error in native method will be generated if AWT or Swing event handling thread tries to access it.

To avoid fatal errors, always call *SourceManager.instance().close()* at the end of image acquisition.

## Software Packaging and Distribution

So you have successfully developed your Java applications with JTwain. It's time to distribute your programs to end users. First, make sure you are an authorized licensee registered with LAB Asprise!. To purchase a license, please visit: <http://www.asprise.com/product/jtwain>

There are two files about JTwain you need to distribute along with your own binary code. One is *JTwain.jar*, which is like any other java library, you can just copy it and put it in the class path. The other one is *AspriseJTwain.dll*, the native library. There are many ways to 'install' this dll file, you can:

- Copy the native library to the same directory where JTwain.jar locates, or
- Add the native library to the PATH variable, or
- Copy the native library to **jre/bin** directory – 'install' the library to the JVM, or
- Copy the native library to a specific location, eg. C:\AspriseJTwain.dll, before calling *SourceManager.instance()*, call: *SourceManager.setLibraryPath(" C:\AspriseJTwain.dll ")*;

**Note:** JTwain SDK installer copies the native dll library to Windows system directory during the installation.

## 5. IMAGE ACQUISITION UI COMPONENTS

### 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 2 *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 cameras 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](http://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	
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 written; 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

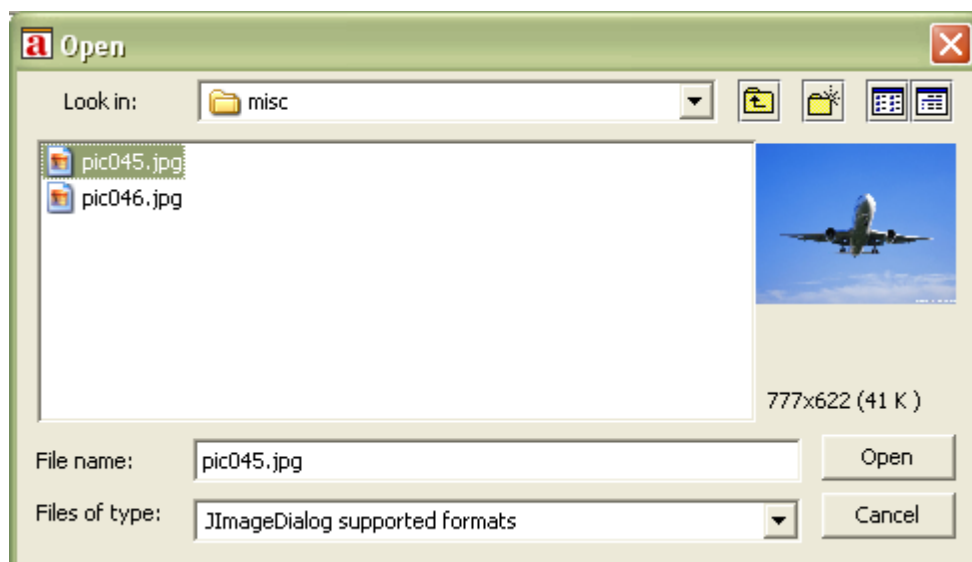
Mandatory: **jid.jar, JTwain.jar**

Optional: **AspriseJTwain.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**

## 6. SUPPORT AND PROFESSIONAL SERVICES

### Support Web Site

<http://www.asprise.com/product/jtwain>

### Basic Support

Our team provides basic support for general Asprise JTwain developers. Email your technical questions to [support@asprise.com](mailto: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](mailto:info@asprise.com)

# # #