The Java Developer's Guide to

Asprise OCR SDK

Version 2.1

Document Version: 2.1

[Last updated on September 7, 2004]

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

Table of Contents

1. INTRODUCTION

About OCR

OCR (Optical Character Recognition) is the technology that allows you to transform images (e.g., images scanned from paper documents) into editable text-based computer files.

About Asprise OCR SDK

Embedded a high performance OCR engine, Asprise OCR SDK is OCR software development kit that can be used with Java, .Net, Delphi, Visual Basic (VB), Borland C, etc.

Features of Asprise OCR

An incomplete list of features offered by Asprise OCR:

1. **Highest Level of Accuracy**
   Asprise OCR can easily recognize difficult documents of poor image quality;

2. **Excellent Format Retention**
   Text layouts on the input documents are perserved;

3. **High Speed**
   Asprise OCR uses optimized OCR engine to perform excellent recognition in very short time;

4. **Ease of Use**
   We strive to make the developer's life easier. Complex parameter configurations are removed from Asprise OCR SDK. You only have to supply the image document. Asprise OCR can intelligently determines the best setting internally.

5. **Barcode Recognition**
   Beside characters (letters and numbers), Asprise OCR can recognize almost every kind of bar code. You can choose to recognize barcode or characters or both.

Components of Asprise OCR for Java

AspriseOCR comprises two essential components:

- A native library: **AspriseOCR.dll** (for windows only) / **libAspriseOCR.so** (*nix).
- one Java package:
  * **com.asprise.util.ocr** - main package; contains essential classes to perform OCR.

Asprise OCR for Java Installation

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

Download a copy of Asprise OCR installation file from http://www.asprise.com/product/ocr. Double click the installation file to launch the installation process. Then follow the instruction to install it.

File Organization

The file organization of Asprise OCR SDK distribution is as follows:

ASPRISE_OCR_SDK_HOME
+--- **images** [folder, containing sample image documents]
+--- **javadoc** [Java docs]
+--- **UI-eval** [UI related libraries, optional]

+--- **AspriseOCR.dll / libAspriseOCR.so** [The sole native library]
+--- **aspriseOCR.jar** [Contains all Asprise OCR classes]

+--- **DevelopersGuidetoAspriseOCR-Java.pdf** [The developer's guide]

+--- **runDemo1-4.bat/sh** [OCR demos on different image documents]
+--- **runDemoUI.bat/sh** [OCR demo with userfriendly UI]

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

Development Environment Setup

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

Put aspriseOCR.jar into your class path.

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

Compatibility

Operating Systems:	**Windows platforms and Linux/Unix.**
Java Runtime:	**Version 1.2 or above**.

2. PERFORMING OCR WITH ASPRISE OCR IN JAVA

Jump Start

The following code demonstrates the basic usage of Asprise OCR:

```
1. import com.asprise.util.ocr.OCR
2. ...
3.
4. // loads the image.
5. BufferedImage image = ImageIO.read(new File("ocr.gif"));
6.
7. // recognizes both characters and barcodes
8. String s = new OCR().recognizeAll(image);
9.
10. // prints the results.
11. System.out.println("RESULTS: \n"+ s);
```

Line 1: Imports the main OCR class;

Line 5: Loads the source image. Later, we are going to perform OCR for this image. You can load it from an existing file, or you can acquire it from a scanner through JTwain for Windows (http://asprise.com/product/jtwain) or JSane for Unix/Linux (http://asprise.com/product/jsane).

Line 8: All the OCR work is done here. The *recognizeAll* method of the *com.asprise.util.ocr.OCR* class recognizes all the characters and barcodes from the image and output them as a string.

Line 11 Prints the result of the OCR. Of course, you can perform advanced operations such as editing, spelling check on the results.

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

TIP: Source code for all demo programs are in the file samples-src.jar.

Input and Output

For the simple OCR program in above section, if the input is the figure below:

Figure 1

Then the output will be:

```
[123456789012]
Asprise OCR
Speed. Accuracy.
```

The first line is extracted from the barcode. Note the content of the barcode is enclosed in a '[]' pair.

The second and third lines come from the text regions in the image.

Advanced Usages

This section covers the essential functions offered by the com.asprise.util.ocr.OCR class:

1) **Recognizes characters (letters and numbers) from the specified image**

```
1. String s = new OCR().recognizeCharacters(image);
```

The *recognizeCharacters* method only returns the text characters. Barcodes are skipped. So the output for *Figure 1* will be:

```
Asprise OCR
Speed. Accuracy.
```

2) **Recognizes a barcode from the specified image**

```
1. String s = new OCR().recognizeBarcode(image);
```

The *recognizeBarcode* method recognizes a barcode from the specified image. If there is more than one barcode, only the first barcode will be recognized. If you need to recognize more than one, you should use the *recognizeBarcodes* method. So the output for *Figure 1* will be:

```
123456789012
```

3) Recognizes one or more barcode from the specified image

```
1. Vector barcodes = new OCR().recognizeBarcodes(image);
```

The *recognizeBarcodes* method recognizes one or more barcode from the specified image. For your convenience, if there is only one barcode on the image or you only want the first one, you can use the *recognizeBarcode* method. So the output for *Figure 1* will be:

```
Vector (size=1): {element #0: 123456789012}
```

4) Recognizes characters and barcodes from the specified image

```
1. Vector barcodes = new OCR().recognizeEverything(image);
```

The *recognizeEverything* method recognizes characters and barcodes from the specified image. The content of a recognized bar code will be enclosed in a '[]' pair. If you need customize the prefix and suffix for barcodes, use the other function with the same name. So the output for *Figure 1* will be:

```
[123456789012]
Asprise OCR
Speed. Accuracy.
```

5) Recognizes characters and barcodes from the specified image

```
1. Vector barcodes = new OCR().recognizeEverything(image, "<", ">");
```

The *recognizeEverything* method recognizes characters and barcodes from the specified image. This function allows you to customize the prefix and suffix for recognized barcodes. With the above line, the output for *Figure 1* will be:

```
<123456789012>
Asprise OCR
Speed. Accuracy.
```

*) Setting native library path

By default, Asprise OCR loads the associated native library from the default directories, such as system directories. If you'd like Asprise OCR to load the native library from a specified path, you can use this method.

The default library path is *null*, in which case the system will load the native library from default locations, such as system directory.

3. ADVANCED TOPICS

Using Asprise OCR in Threads

The library is thread-safe.

Software Packaging and Distribution

So you have successfully developed your Java applications with Asprise OCR. 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/ocr**

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

- Copy the native library to the same directory where aspriseOCR.*jar* locates (Win32 only), or
- Insert the directory path containing the native library to the LD_LIBRARY_PATH variable
- 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:\AspriseOCR.dll, before calling *new OCR()*, call: *OCR.setLibraryPath(" C:\AspriseOCR.dll ")*;

4. IMAGE ACQUISITION UI COMPONENTS

The image acquisition UI components are not part of Asprise OCR library. However, based on our customers' experience, if you need to build a front-end for OCR, they are invaluable and could save you a lot of time. Otherwise, you may skip this chapter.

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.

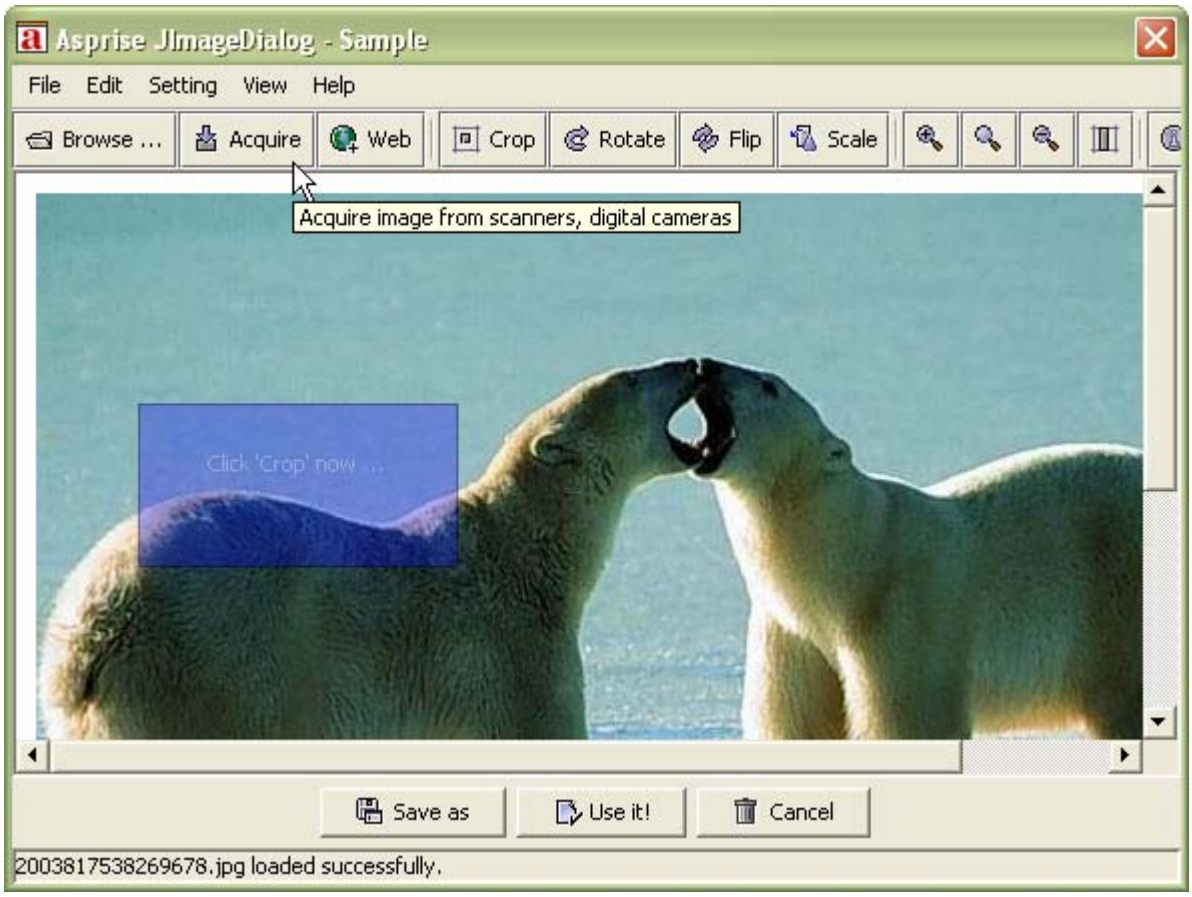


Illustration 1 JImageDialog

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.

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	
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**.

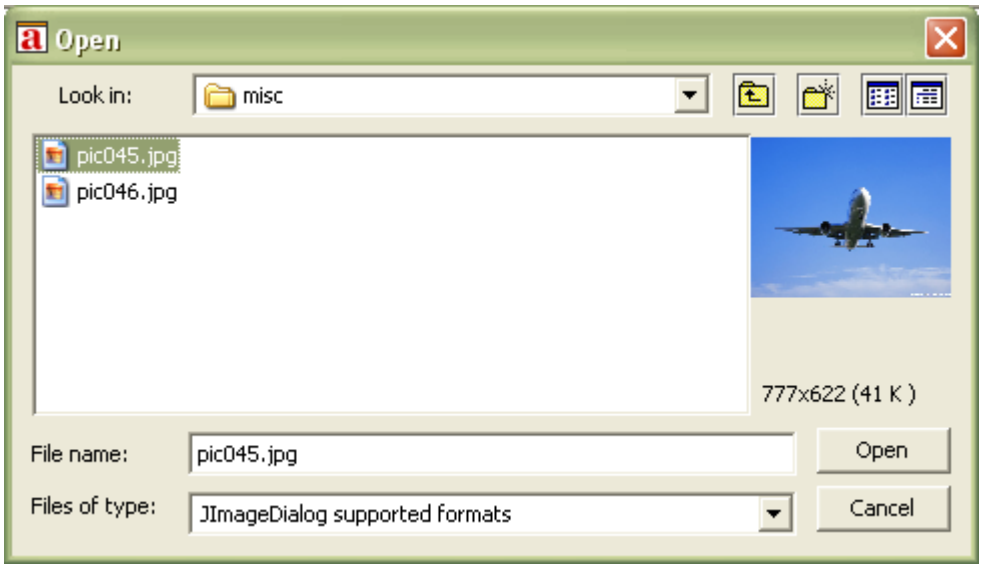
Software Packaging and Distribution

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

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/ocr

Basic Support

Our team provides basic support for general Asprise OCR 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

# # #