The Java Developer's Guide to

Asprise OCR SDK 4.0

Prepared by: LAB Asprise!

Nov 2007

www.asprise.com

Table of Contents

1 Introduction

1.1 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.

1.2 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. on multiple platforms – Windows, Linux, Mac OS, Solaris and AIX.

1.3 Features of Asprise OCR

An incomplete list of features offered by Asprise OCR:

- **Highest Level of Accuracy**
  Asprise OCR can easily recognize difficult documents of poor image quality

- **Excellent Format Retention**
  Text layouts on the input documents are preserved;

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

- **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 determine the best setting internally.

- **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. Currently, the following bar code formats are supported:

- CODE 128 (128b, 128C, 128raw)
- EAN 8 EAN 13
- UPC
- code 3 of 9
- code interleaved 2 of 5

1.4 Components of Asprise OCR for Java

Asprise OCR comprises two essential components:

- **A native library**:
  - *AspriseOCR.dll* [on Windows]
  - *libAspriseOCR.so* [on Linux]
  - *libAspriseOCR.jnilib* [on Mac OS]
  - *libAspriseOCR.so* [on Solaris]

- **one Java package**:
  - *com.asprise.util.ocr* [Cross platforms] main package; contains essential classes to perform OCR.

1.5 Compatibility of Asprise OCR SDK

Currently the following OS are supported:

OS	**Evaluation** version availability	**Licensed** version availability
Windows XP 32bit	**Yes** (Java, VB, C, C++, VB.NET & C#)	**Yes** (Java, VB, C, C++, VB.NET & C#)
Windows XP 64bit	**Yes** (Java, VB, C, C++, VB.NET & C#)	**Yes** (Java, VB, C, C++, VB.NET & C#)
Windows Vista 32bit	**Yes** (Java, VB, C, C++, VB.NET & C#)	**Yes** (Java, VB, C, C++, VB.NET & C#)
Windows Vista 64bit	**Yes** (Java, VB, C, C++, VB.NET & C#)	**Yes** (Java, VB, C, C++, VB.NET & C#)
Windows Server 32bit	**Yes** (Java, VB, C, C++, VB.NET & C#)	**Yes** (Java, VB, C, C++, VB.NET & C#)
Windows Server 64bit	**Yes** (Java, VB, C, C++, VB.NET & C#)	**Yes** (Java, VB, C, C++, VB.NET & C#)
MacOS X PowerPC	**Yes** (Java & command line tool)	**Yes** (Java & command line tool)

MacOS X Intel	**Yes** (Java & command line tool)	**Yes** (Java & command line tool)
Linux 32bit	**Yes** (Java & command line tool)	**Yes** (Java & command line tool)
Linux 64bit	**Yes** (Java & command line tool)	**Yes** (Java & command line tool)
Solaris SPARC	**Yes** (Java & command line tool)	**Yes** (Java & command line tool)
Solaris x86	**Yes** (Java & command line tool)	**Yes** (Java & command line tool)
AIX	**No** (Please evaluate on other platforms)	**Yes** (Java & command line tool)
HP-UX	**No** (Please evaluate on other platforms)	**Yes** (Java & command line tool)

1.6 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 SDK from http://www.asprise.com/product/ocr. Simply unzip it to an empty folder. Let refer this folder as OCR_ HOME.

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

```
OCR_HOME
+--- sample-images [folder, containing sample image documents]
+--- javadoc [Java docs]
+--- UI-Eval [UI related libraries, optional]

+--- AspriseOCR.dll / libAspriseOCR.so / libAspriseOCR.jnilib [native
libraries]
+--- aspriseOCR.jar [Contains all Asprise OCR classes]

+--- Asprise OCR SDK v4 Java Developer's Guide.pdf [The developer's
guide]
+--- demo.jar, demo-src.jar [Binary and source code of demo programs]
+--- runDemo1-4.bat/sh [OCR demos on different image documents]
+--- runDemoUI.bat/sh [OCR demo with user-friendly UI]

+--- LICENSE-EVALUATION.txt [License agreement]
+--- HOW-TO-ORDER-ASPRISE-OCR.htm [Click to order Asprise OCR]
```

1.7 Development Environment Setup

After you have obtained and unzipped the Asprise OCR SDK kit, you need to setup your development environment in order to develop Java applications with Asprise OCR. To do so, you need to:

- **Put *aspriseOCR.jar* into your class path, and**
- **Put the native library into the library path**.

1.7.1 Windows

On Windows, the system path is part of the library path. So you can put the DLL into any of the folder in the system path, e.g., C:/Windows/System32.

1.7.2 Linux/Solaris/Mac OS X/Other Unix

On Linux/Solaris/Mac OS, LD_LIBRARY_PATH is part of the library path. Execute this command before launch the JVM: **export LD_LIBRARY_PATH=*OCR_HOME*:$LD_LIBRARY_PATH** to add the OCR_HOME to the library path. For detailed example, see runDemo1.bat/sh.

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

2 OCR with Asprise OCR in Java

2.1 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.*

2.2 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.

2.3 Advanced Usages

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

2.3.1 Recognizes characters (letters and numbers) from the specified image

```
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.
```

2.3.2 Recognizes characters from part of the image

In some cases, you might not want to OCR the whole image. In that case, you can OCR on part of the image only:

```
1    BufferedImage image = ImageIO.read(file);
2
3    image = image.getSubimage(0, 0, 200, 100);
4    String s = new OCR().recognizeEverything(image);
```

The above code OCR the top left part of the image with width 200 pixels and height 100 pixels.

Reference (copied from javadoc of *java.awt.image.BufferredImage*):

```
public BufferedImage getSubimage (int x, int y, int w, int h)

    Returns a subimage defined by a specified rectangular region. The
returned BufferedImage shares the same data array as the original
image.

    Parameters:
        x - the X coordinate of the upper-left corner of the
specified rectangular region
        y - the Y coordinate of the upper-left corner of the
specified rectangular region
        w - the width of the specified rectangular region
        h - the height of the specified rectangular region
    Returns:
        a BufferedImage that is the subimage of this BufferedImage.
    Throws:
        RasterFormatException - if the specified area is not
contained within this BufferedImage.
```

2.3.3 Recognizes a barcode from the specified image

```
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
```

2.3.4 Recognizes one or more barcode from the specified image

```
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}
```

2.3.5 Recognizes characters and barcodes from the specified image

```
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 (will be introduced soon). So the output for Figure 1 will be:

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

2.3.6 Recognizes characters and barcodes from the specified image

```
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.
```

2.3.7 Sets the OCR native library path explicitly

```
OCR.setLibraryPath("C:/tmp/AspriseOCR.dll");
```

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

The default OCR native library path is null, in which case the system will load the native library from the library path.

3 Advanced Topics

3.1 Using Asprise OCR in Threads

This library is thread-safe.

3.2 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 main 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 the native library *AspriseOCR.dll* (depends on *ILU.dll & DevIL.dll*, you need to copy them into the target computer's library path too.) for Windows, *libAspriseOCR.so* for Linux, *libAspriseOCR.jnilib* for Mac OS, and *libAspriseOCR.so* for Solaris. There are a few ways to 'install' the native libraries:

- Insert the directory path containing the native library to the system path (for Windows) or to the LD_LIBRARY_PATH variable (for other OS), or
- Copy the native library to jre/bin directory – 'install' the library to the JVM

4 OCR TIFF and PDF Files

4.1 Perform OCR on TIFF Images

With Asprise TIFF library, you can perform OCR on TIFF images.

First, download the Asprise TIFF library from http://asprise.com/product/javatiff/. Unzip the Java TIFF kit and add aspriseTIFF.jar to the classpath.

Sample code:

```
1   OCR ocr = new OCR();
2   TIFFReader reader = new TIFFReader("C:\\my.tiff"); // loads the
      TIFF file.
3
4   for(int i=0; i<reader.countPages(); i++) { // for each page
5     System.out.println("Page #" + i);
6     RenderedImage image = reader.getPage(i);
7     System.out.println("OCR result:\n" + ocr.recognizeAll(image));
8   }
```

Line 4: reader.countPages() returns the total number of pages of the TIFF file;

Line 6: reder.getPage() returns the specified individual page;

Line 7: performs OCR on the page.

***Note**: Asprise TIFF library is not included in Asprise OCR package. You need to order it separately from http://asprise.com/product/javatiff/order.php*

4.2 Perform OCR on PDF files

With Asprise TIFF library, you can perform OCR on PDF files.

First, download the Asprise PDF library from http://asprise.com/product/javapdf. Unzip the Java PDF kit and add AspriseJavaPDF.jar to the classpath.

Sample code:

```
9   OCR ocr = new OCR();
10  PDFReader reader = new PDFReader(new File("my.pdf"));
11   reader.open(); // open the file.
12   int pages = reader.getNumberOfPages();
```

```
13
14   for(int i=0; i<pages; i++) {
15    BufferedImage image = reader.getPageAsImage(i);
16    System.out.println("OCR result:\n" + ocr.recognizeAll(image));
17  }
18
19  reader.close(); // finally, close the file.
```

Line 2: creates a PDF reader for the given PDF file;

Line 4: reader.getNumberOfPages() returns the total number pages contained in the PDF file;

Line 7: reader.getPageAsImage() reads the specified page into an image;

4.2.1 PDF text extraction

If the PDF file is text based, you should use PDF text extraction instead of OCR as text extraction is much faster and accurate in most cases. PDF text extraction requires Asprise PDF library.

Sample code:

```
1   OCR ocr = new OCR();
2   PDFReader reader = new PDFReader(new File("my.pdf"));
3    reader.open(); // open the file.
4    int pages = reader.getNumberOfPages();
5
6    for(int i=0; i<pages; i++) {
7     String txt = reader.extractTextFromPage(i);
8     System.out.println("Text result:\n" + txt);
9   }
10
11  reader.close(); // finally, close the file.
```

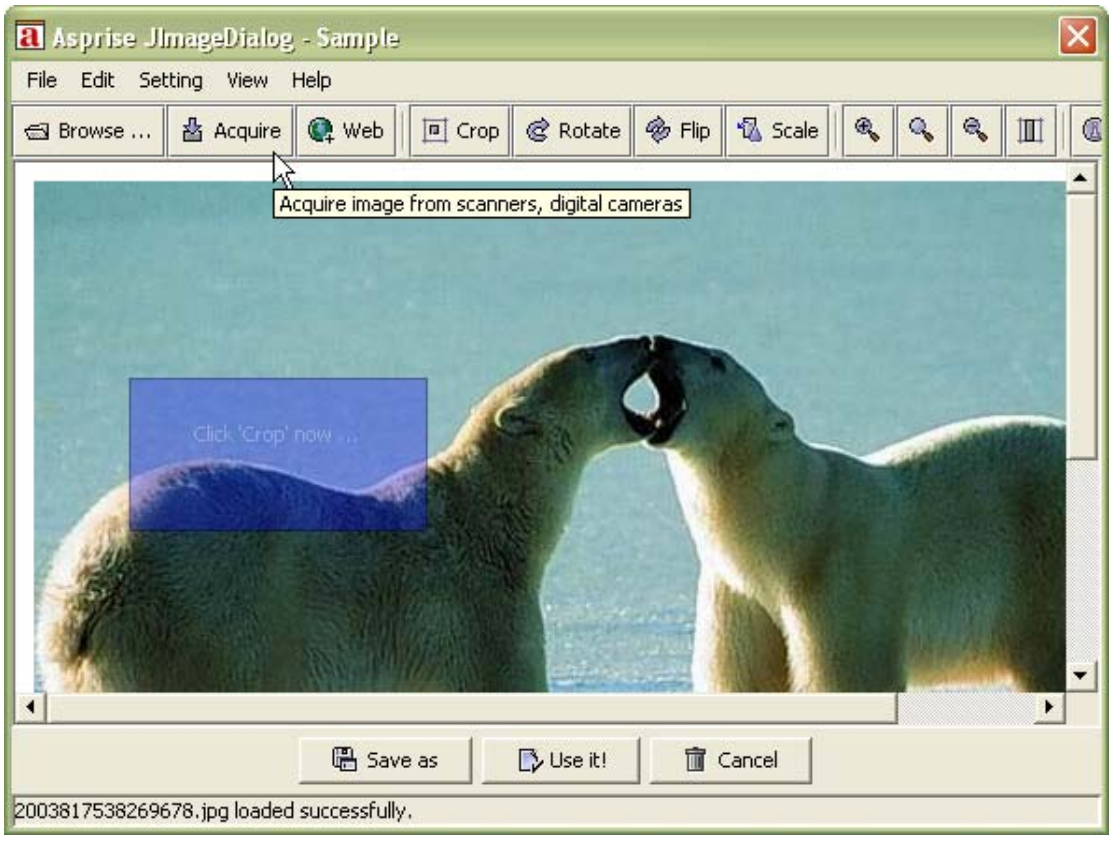
***Note**: Asprise PDF library is not included in Asprise OCR package. You need to order it separately from http://asprise.com/product/javapdf/order.php*

5 Image Acquisition 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.

5.1 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 (i.e. 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.

5.1.1 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.

5.1.2 Sample Uses

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:

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 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*

5.1.3 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 – e.g. 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!.

5.1.4 Compatibility

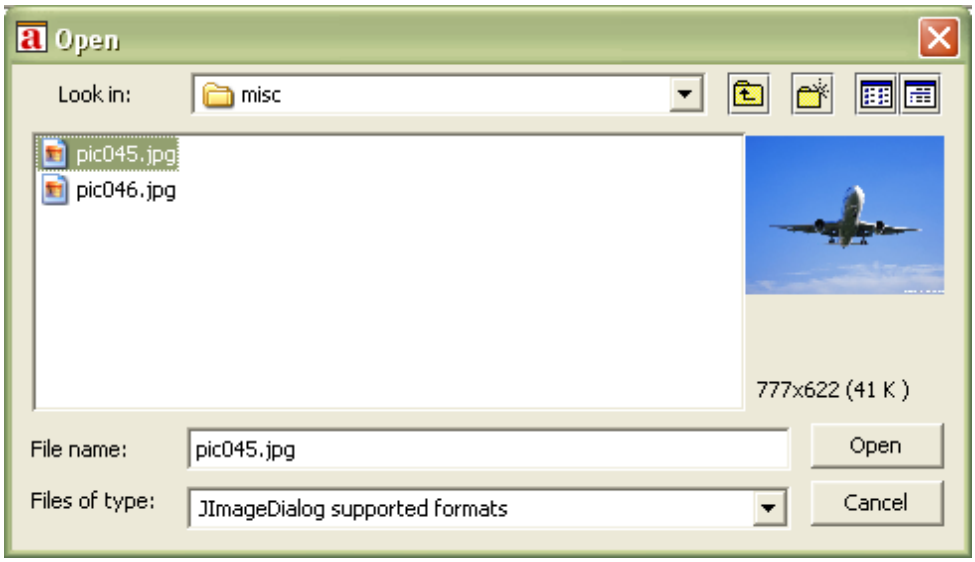
All Java runtimes with version 1.3 or above.

5.1.5 Software Packaging and Distribution

Mandatory: *jid.jar, JTwain.jar*

5.2 JImageFileChooser

JImageFileChooser is 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.

5.2.1 Sample Use

```
1   JFileChooser fc = new JImageFileChooser(lastDirectory);
2   fc.addChoosableFileFilter(JImageFileChooser.getImageFileFilter());
3   int returnVal = fc.showOpenDialog(frame);
4   ...
```

Line 1 creates the image file chooser;

Line 2 set the file filter.

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

5.2.2 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!.

5.2.3 Compatibility

All operating systems; All Java runtimes with version 1.2 or above.

5.2.4 Software Packaging and Distribution

Mandatory: **jid.jar**

6 Support and Professional Services

6.1 Support Web Site

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

6.2 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 solved quickly.

6.3 Premium Support Services + Updates

For premium support service subscribers: your requests will be of our top priority and your email will be replied within 24 business hours. You entitle to source code/binary update.