DEPLOYMENT GUIDE FOR JAVA

V1.0

Last updated on January 2006

Table Of Contents

1 Putting *.class files into Jars

Jar files are the recommended java binary code distribution format. This section will guide you to put *.class files into jar file.

1.1 Organizing your directory

It is important to organize your project directory properly first. Let's say you have the following Java class:

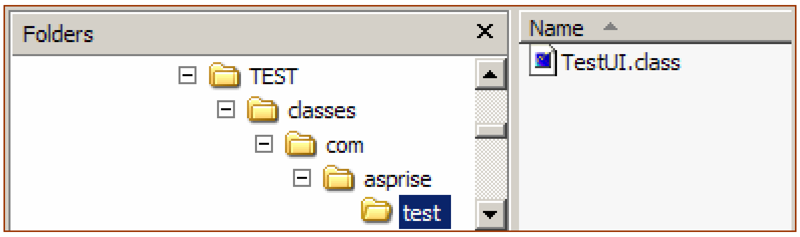
```
package com.asprise.test;
...
public class TestUI extends JApplet {

public TestUI() {
    getContentPane().setBackground(Color.pink);
    getContentPane().setLayout(new BorderLayout());
    JLabel label = new JLabel("Testing Message.");
    label.setHorizontalAlignment(JLabel.CENTER);
    getContentPane().add(label, BorderLayout.CENTER);
...
}


public static void main(String[] args) {
    JFrame frame = new JFrame("TestUI");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(new TestUI());
    frame.setSize(400, 300);
    frame.setVisible(true);
}
}
```

The class name is *TestUI* and it is in package ***com.asprise.test***.

For this class you should organize the directory as following:



[Assuming *TEST* is your root folder. The absolute path of TEST is *E:\eclipse-301\workspace\TEST*]

1.2 Creating a Jar File

Now, you change directory to the *classes* folder, and type the following command to put all the files in the *classes* directory into a jar file named *program.jar*.[1]

```
E:\eclipse-301\workspace\TEST\classes>jar cvf program.jar *
```

Now, all the files in the classes directory has been put into: *E:\eclipse-301\workspace\TEST\classes\program.jar*

Now, move the *program.jar* to the *E:\eclipse-301\workspace\TEST folder.*

```
E:\eclipse-301\workspace\TEST\classes>move program.jar ..
```

Note: If your class is in the default package (i.e., you did not specify any package information in the source code), you should put it in the *classes* folder. Otherwise, you should create the corresponding package directory under the *classes* folder and put the class file into the package directory as shown above.

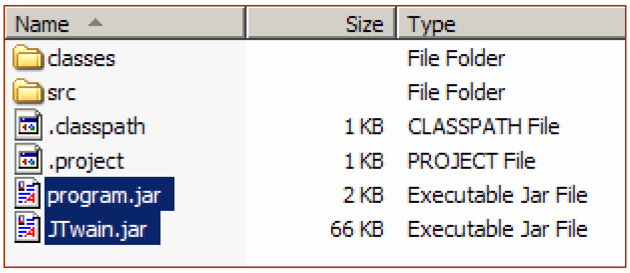
[1] In this document, we use Microsoft Windows as the hosting OS. You can easily convert the commands onto other platforms, like Mac OSX, Linux, etc.

2 Creating Signed Applets

Only signed applets can be granted with all the permissions. To enable an applet to access dll files (e.g., JTwain) or other native services, you need to sign the all the jar files required by the applet with the same certificate. This section provides a step-by-step guide.

First, if you have .class files, make sure you put them into a jar as shown in the first chapter.

In our sample project, we have a jar named *program.jar* containing our program binary class files. We also have a library file named *JTwain.jar*:



Name	Size	Type
classes		File Folder
src		File Folder
.classpath	1 KB	CLASSPATH File
.project	1 KB	PROJECT File
program.jar	2 KB	Executable Jar File
JTwain.jar	66 KB	Executable Jar File

2.1 Creating a Certificate

Before you can sign jar files, you need a certificate. If you already have one, you can skip this procedure.

First, change directory to the root directory of the project:
E:\eclipse-301\workspace\TEST

Run the following command:

```
E:\eclipse-301\workspace\TEST>keytool -genkey -dname "cn=YOUR NAME,
ou=ORG UNIT, o=COMPANY, c=US" -alias test -keypass testpass -validity 999
-keystore test -storepass testpass
```

A file named *test* containing the certificate is generated under the *TEST* folder.

2.2 Signing Jar Files

Use the following command to sign each jar file:

```
E:\eclipse-301\workspace\TEST>jarsigner -keystore test -storepass
testpass -keypass testpass program.jar test

E:\eclipse-301\workspace\TEST>jarsigner -keystore test -storepass
testpass -keypass testpass JTwain.jar test

...
```

Now, all the jar files have been signed. You can launch the applet with a proper HTML page.

2.3 Launching the Applet

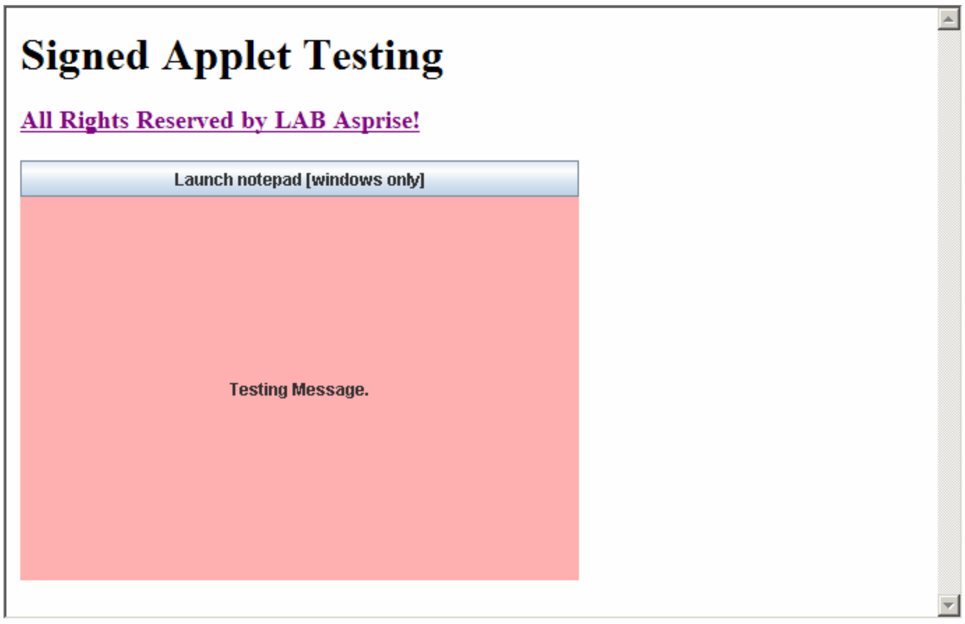
Now, you can use HTML code like the following to invoke the applet:

```
<html>
<head>
<title>TestUI</title>
</head>
<body>
<h1>Signed Applet Testing</h1>
<h3><a href="http://asprise.com">All Rights Reserved by LAB
 Asprise!</a></h3>

<applet code="com.asprise.test.TestUI.class" codebase="."
 archive="program.jar, JTwain.jar" width="400" height="300">
Oops, Your browser does not support Java applet!
</applet>

</body>
</html>
```

The screenshot:


Signed Applet Testing

All Rights Reserved by LAB Asprise!

3 Our Products, Our Services

Popular products offered by LAB Asprise! include:

3.1 Asprise OCR

http://asprise.com/product/ocr
Embedded with a high performance OCR (optical character recognition) engine, Asprise OCR SDK library for Java, VB.NET, CSharp.NET, VC++, VB6.0, C, C++, Delphi on Windows, Mac, Linux and Solaris, enables you to equip your applications with OCR ability easily.

3.2 JTWAIN

http://asprise.com/product/jtwain
Acquiring images from any kinds of digital cameras, scanners with Java! With JTwain, you can access, control any kinds of digital cameras and scanners from Java. Image acquisition has been made extremely easy by this JTwain package.

3.3 JSANE

http://asprise.com/product/jsane
SANE is the de facto standard to access scanners/cameras on AIX, BeOS, Darwin, FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS2, Solaris, Unixware, Unix platforms. JSANE provides SANE access APIs 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.

3.4 Java Image Acquisition/Editor UI Components

http://asprise.com/product/jid
Embed Image Acquisition UI Components into Your Java Application to Achieve Better User Experience. If you are developing some applications that require the user to select/edit/input images, then Image Acquisition UI Components (The UI Components) will make your life extremely easy - and more importantly, the user experience will be improved dramatically. Currently the following components are available: *JImageDialog* - an image acquisition UI component that allows the user to load images and to perform basic image editing tasks. *JFileChooser* - An extended JFileChooser that supports image preview and image information extraction.

3.5 Java TIFF Reader/Writer Library

http://asprise.com/product/javatiff
Asprise offers TIFF writer and reader library as valued add-on to our flagship products – Asprise OCR & JTwain. Tagged Image File Format (abbreviated TIFF) is a file format for mainly storing raster images. With Asprise Java TIFF library, you can easily create, manipulate (read and write), disassemble TIFF files easily.

3.6 Support Service

You can subscribe our support service for any of the products listed above. For more details, please visit the individual product homepage.

Visit us online at: http://www.asprise.com
Sales enquiries: sales@asprise.com
Technical enquiries: support@asprise.com