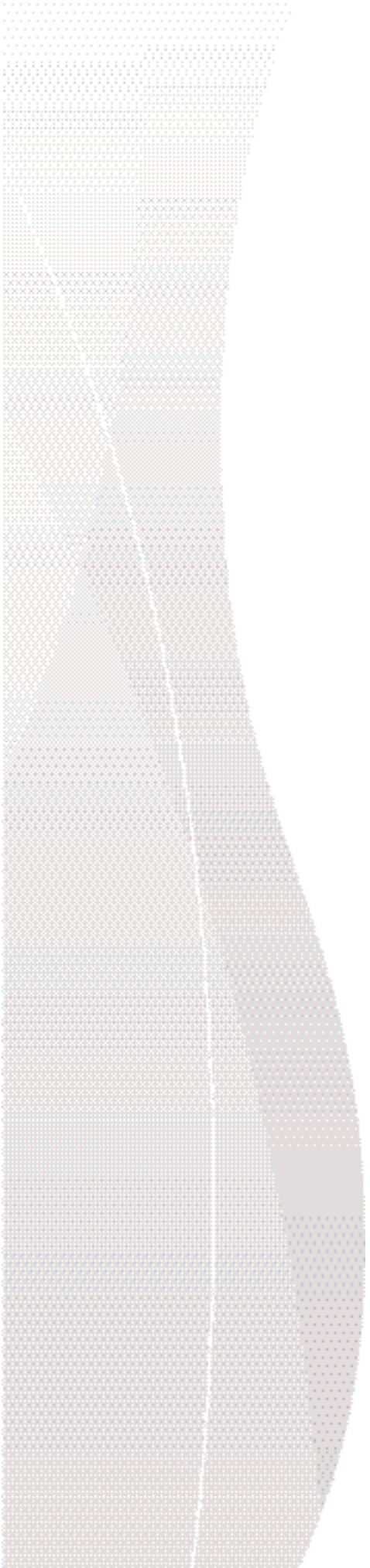

JAVA™ WEB START OVERVIEW

White Paper
May 2005

Table of Contents

Chapter 1
Introduction

Java™ Web Start provides the power to launch full-featured Java applications with a single click. Users can download and launch applications, such as a complete spreadsheet program or an Internet chat client, without going through complicated installation procedures. Once accessed through Java Web Start, the application integrates seamlessly with the user's desktop; from the user's point of view, it functions just as a native application. Additionally, Java Web Start serves users by managing Java™ Runtime Environment versions and automatically updating the application version.

Java Web Start helps developers deploy their applications as well. The design and development of applications for Java Web Start is substantially the same as it is for any other Java applications, and migrating a legacy application to Java Web Start is, in most cases, trivial. Deploying the application so users can access it from a web page involves just a few simple steps on the Web server.

This paper is written for developers and IT staff who are considering how to deploy their applications. It gives a broad overview of how Java Web Start works from both the user's and developer's perspectives, and highlights the key advantages of choosing Java Web Start.

Note – This paper is based on Java™ 2 Platform, Standard Edition (J2SE™), version 5.0.

Chapter 2

A Java Web Start Application from the Users' Perspective

Java Web Start is designed to make it easy for users to access and work with robust Java applications. Users enjoy easy installation and use, Java Runtime Environment version management, and integration with their Windows, Solaris™ Operating System (OS), and Linux desktops. Java Web Start also enables easy application updating.

The rest of this chapter describes how a user would work with a sample Notepad application. You can find all the source files for the Notepad application example in the `\demo\plugin\jfc\Notepad` directory within the Java™ Development Kit (JDK™).

Installing and Using the Notepad Application

Users can download a Java Web Start application by simply clicking a link on a web page. Figure 1 shows a web page from which you can access the sample Notepad application.

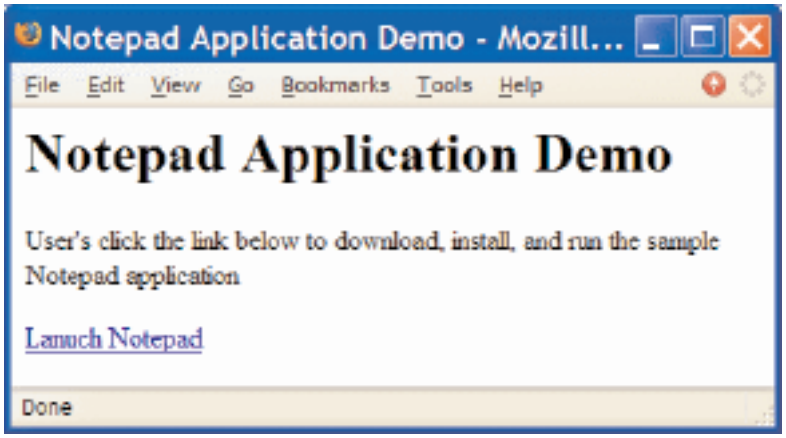

Figure 1 – Web page for Notepad application demo

Note – While this paper describes deploying the application from a web site because it is the simplest method, Java Web Start enables an enterprise to deploy an application through a CD or network share as well. When the user clicks the Launch Notepad link, the Notepad application opens in a separate window, as shown in Figure 2.

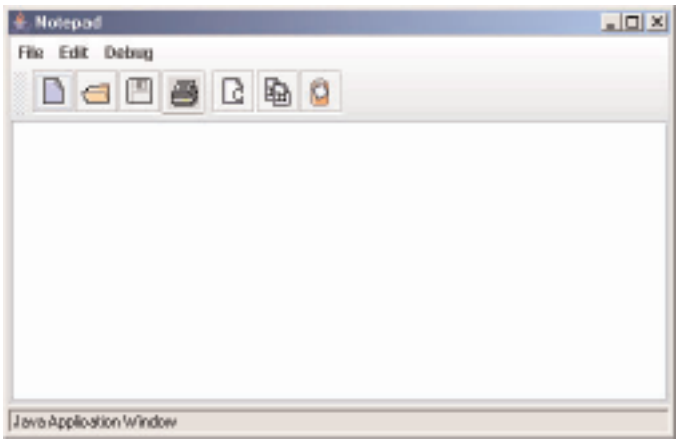

Figure 2 – Notepad application

Behind the scenes of this simple action two important things happen:

- If the user does not have the correct version of the Java Runtime Environment installed locally, Java Web Start downloads and installs it.
- Java Web Start adds the Notepad application to the local computer so the user can access it later without returning to the web page.

These actions are described in more detail in the following sections.

Java Runtime Environment Installation and Version Management

In order to run the Notepad application, the user must have the Java Runtime Environment installed on his computer. In addition, the version of the Java Runtime environment must match the version required by the Notepad application, as specified by the developer deploying the application.

Java Web Start downloads, installs, and manages one or more Java Runtime Environments as required by the applications the user runs. The following scenarios describe how Java Web Start manages this task.

- A user accessing the Notepad application currently has no Java Runtime Environment installed. The Notepad application requires a Java Runtime Environment of Version 1.3 or greater. When the user runs Notepad as described above, Java Web Start downloads and installs the latest version of the Java Runtime Environment, currently version 5.0.
- The user has Java Runtime Environment Version 1.3 installed. Java Web Start runs the Notepad application with this version.
- A user accessing the Notepad application currently has Java Runtime Environment Version 1.2 installed. When the user runs Notepad as described above, Java Web Start downloads and installs the latest version of the Java Runtime Environment. This process does not change the configuration of the user's computer and does not affect the previously existing Java Runtime Environment. The downloaded version is only used to run the Notepad application.

- A user runs multiple Java Web Start applications. Some require Java Runtime Environment Version 1.4, and others require Version 5.0. Java Web Start runs each application with the appropriate Java Runtime Environment; the user does not need to intervene or manually configure the computer to run the different versions.

Integration with the Desktop

After running the application for the first time from the web page as described above, the application is cached. This both improves application performance and enables the user to access it without returning to the web page. The Notepad application is added to the Java Application Cache Viewer, as shown in Figure 3.

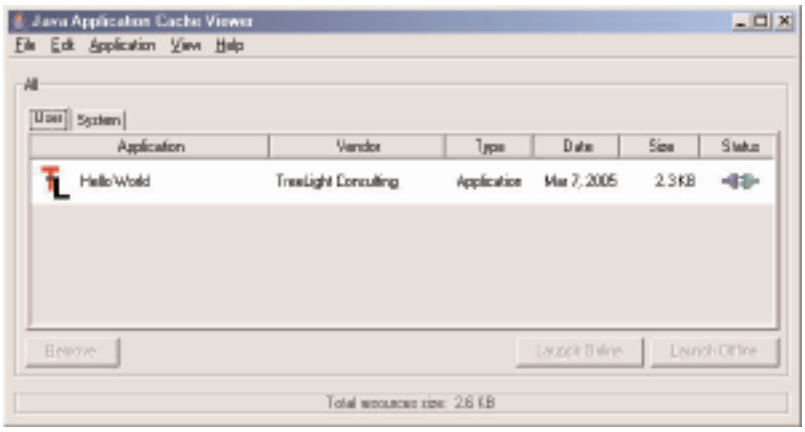

Figure 3 – Java Application Cache Viewer (Version 5.0)

From this dialog box, the user can launch the Notepad application with a single click. All Java Web Start applications can be launched online, provided the computer is connected to the network. Applications for which it is appropriate, such as Notepad, can be launched offline as well.

Note – Developers control whether or not users can run an application offline. An example of a situation where offline use would not be appropriate would be for an application that must access a remote database. Users can simplify running a Java Web Start application further by adding a shortcut to the desktop, by selecting Install Shortcuts from the Application menu in the Java Application Cache Viewer. The resulting shortcut on a Windows desktop is shown in Figure 4.

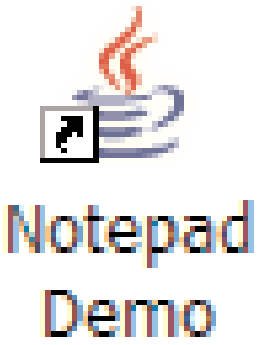

Figure 4 – Shortcut to Notepad application on the desktop

The user can now access Notepad as she would a typical application installed on the local computer, with a single click.

Updating the Application

For an application deployed to multiple users, updating and keeping consistent all installations can be quite burdensome for developers and IT staff. Java Web Start eliminates the man-hours required to update or patch applications by automating the process from each desktop.

When the user starts the application through the Java Application Cache Viewer or a shortcut, Java Web Start automatically checks for application updates. If new versions of any Java™ Archive (JAR™) files that comprise the application are available from the site from which they were originally accessed, Java Web Start downloads them before running the application.

A developer can ensure that users will get a fix simply by replacing necessary files on the web server. There's no need for the vendor to contact customers or ship media with complicated patch instructions, and there's no need for users to take action.

Chapter 3

A Java Web Start Application from the Developer's Perspective

Java Web Start is designed to make deployment as easy for developers as it is for users. You design and package applications to be deployed through Java Web Start in substantially the same way as you do other Java programs. The process for then deploying an application on a web server is quite simple.

Designing and Packaging a Java Web Start Application

For the most part, your Java Web Start application is no different than any standalone Java application. Following are some considerations you should be aware of.

Packaging the Application in JAR Files

You deploy a Java Web Start application as one or more JAR files. You cannot deploy classes outside of a JAR file. In addition, you must deploy all application resources, such as property files and images, in JAR files.

If the application needs unrestricted access to the local system, all JAR files and entries must be signed. See Chapter 4 of this paper for more information about security.

Accessing Resources in a JAR File

You access resources in a JAR file from within Java code using the `getResource` method.

The following code example shows how to retrieve images from a JAR file.

```
// Get current classloader
ClassLoader cl = this.getClass().getClassLoader();

// Create icons
Icon saveIcon = new ImageIcon(cl.getResource("images/save.gif"));
Icon cutIcon = new ImageIcon(cl.getResource("images/cut.gif"));
```

The example assumes that the following entries exist in the JAR file.

- images/save.gif
- images/cut.gif

Developing Applications for a Secure Sandbox

If the application is to run in a secure sandbox, you must adhere to the following:

- The application cannot access the local disk.
- All JAR files for the application must be downloaded from the same server.
- The application can only make network connections to the server from which the JAR files were downloaded.
- A security manager cannot be installed.

- The application cannot use native libraries.
- The application has limited access to system properties. The application has read/write access to all system properties specified in the JNLP file, and read-only access to the same set of properties as an Applet.

See Chapter 4 of this paper for more information about security.

Using the (Java™ Network Launching Protocol (JNLP) API

The Java platform includes the JNLP API to enable you to provide additional information to applications deployed through Java Web Start. The JNLP API makes it easy for you to add functionality to an application to do such things as:

- Determine whether or not the application is online.
- Interact with the computer's clipboard.
- Work with files on the computer.

The JNLP API is included in the `javax.jnlp` package, which is delivered in the `jnlp.jar` as part of the Java Development Kit version 5.0, in the `sample\jnlp\servlet` directory.

Deploying a Java Web Start Application

Deploying an application involves the following steps:

1. Setting up the web server
2. Creating the JNLP file
3. Placing the application on the web server
4. Creating the web page

Setting up the Web Server

To deploy an application with Java Web Start over the web, you must ensure that the web server you are using can handle JNLP files.

You configure the web server so that files with the `.jnlp` extension are set to the `application/x-java-jnlp-file` MIME type.

How you set the JNLP MIME type depends on the web server you are using. For example, for the Apache web server, you simply add the line `application/x-java-jnlp-file JNLP` to the `mime.types` file.

Creating the JNLP File

The key to running an application with Java Web Start is the Java Network Launching Protocol, or JNLP, file. The JNLP file is an XML file, which contains elements and attributes that tell Java Web Start how to run the application.

Following is the JNLP file for the Notepad application:

```
<?xml version="1.0" encoding="utf-8"?>

<jnlp spec="1.0"
      codebase="URL of application on your Web server"
      href="Notepad.jnlp">
   <information>
      <title>Notepad Demo</title>
      <vendor>Sun Microsystems, Inc.</vendor>
      <offline-allowed/>
   </information>
   <resources>
        <jar href="Notepad.jar"/>
        <j2se version="1.3+"
            href="http://java.sun.com/products/autodl/j2se"/>
   </resources>
   <application-desc main-class="Notepad"/>
</jnlp>
```

Placing the Application on the Web Server

The next step in deploying your application with Java Web Start is as simple as placing all the JAR files and the JNLP file on the web server. You must ensure the JAR files are in the locations specified by the `href` attribute of the `jar` element in the JNLP file.

Creating the Web Page

You are now ready to write a web page that gives users access to your application.

In order to enable your users to launch the application from a web page, you must include a link to the application's JNLP file from that web page. To add this link, you use the standard HTML link syntax, with the `href` attribute specifying the location of the JNLP file:

```
<a href="Notepad.jnlp">Launch Notepad Application</a>
```

Assuming Java Web Start is installed on the client computer, when the user clicks this link, Java Web Start:

1. Checks the user's computer for the required version of the Java Runtime Environment and if it is not present, downloads it.
2. Adds the Notepad application to the local computer for access through the Java Application Cache Viewer.
3. Runs the application.

For users who don't have Java Web Start installed, you must write scripts in your web page to:

1. Detect which browser the user has.
2. Detect whether Java Web Start is installed.
3. If Java Web Start is not installed, either auto-install it, or direct the user to a download page.

Chapter 4

Security

As a developer you determine the security model used for your application.

By default, Java Web Start applications run in a restricted environment, known as a *sandbox*.

You can also provide for functionality that goes beyond what is allowed in the sandbox by signing the application's JAR files.

The Java Web Start Sandbox

Unsigned JAR files launched by Java Web Start run in the sandbox, meaning they cannot access local files or the network. The purpose of the sandbox is to:

- Protect users against malicious code that could affect local files.
- Protect enterprises against code that could to access and destroy data on networks.

While the application's functionality is restricted, Java Web Start safely provides flexibility to developers to solve common tasks. For example, an application in the sandbox can:

- Use the network to access the host from which the application was downloaded.
- With the user's permission, access a local printer.
- Persist session data.
- Use the JNLP APIs to interact with the local file system.

Providing for Functionality Beyond the Sandbox

Java Web Start supports signed JAR files so that your application can work outside of the sandbox described above.

Java Web Start verifies that the contents of the JAR file have not changed since it was signed. If verification of a digital signature fails, Java Web Start does not run the application.

When the user first runs an application as a signed JAR file, Java Web Start opens a dialog box displaying the application's origin based on the signer's certificate. The user can then make an informed decision regarding running the application.

Security and JNLP Files

For a signed JAR file to have access to the local file system and network, you must specify security settings in the JNLP file. The `security` element contains security settings for the application.

The following example provides the application with complete access to the client system if all its JAR files are signed:

```
<security>
  <all-permissions/>
</security>
```

Chapter 5

Advantages

The preceding chapters have described how Java Web Start makes common deployment tasks easier for both end users and developers.

Following is a summary of the advantages you can get by using Java Web Start to distribute applications.

- **Easy installation**. Users can install a new application by simply clicking a link on a web page.
- **Platform independence**. With Java Web Start, you can place a single Java application on a web server for deployment to a wide variety of platforms, including Windows 98/NT/2000/ME/XP, Linux, and Solaris .
- **Java Runtime Environment management**. Java Web Start supports multiple, simultaneous versions of the Java 2 Standard Edition platform. Specific applications can request specific Java versions without conflicting with the different needs of other applications. Java Web Start automatically downloads and installs the correct version of the Java platform, as necessary, based on the application's needs and the user's environment.
- **Desktop integration**. Users can access any Java Web Start application, including those that rely on the network, just as they can any native application, right from their familiar desktops.
- **Application updates**. You can update an application for all users simply by providing an updated JAR file on the Web server. On each user's computer, Java Web Start checks the Web server for updates when the application runs.
- **Familiar Java development requirements**. You develop applications you intend to deploy with Java Web Start just as you would any Java application, with a few familiar packaging requirements. Updating a legacy application to be deployed through Java Web Start is, in most cases, a simple process.
- **Security**. Java Web Start takes advantage of the inherent security of the Java platform. Application users can be confident that a Java Web Start application is restricted to a sandbox and cannot corrupt their systems. If you have provided for additional functionality and signed the application's JAR files, users decide if they trust the application's source and, if so, allow it to run. Nothing can happen behind the scenes without the user's awareness and approval.
- **Performance**. Applications launched with Java Web Start are cached locally, for improved performance.

Chapter 6

For More Information

You can find detailed technical documentation on Java Web Start at:
http://java.sun.com/j2se/1.5.0/docs/guide/javaws/index.html

Additional information can be found on the Java Developer's site at:
http://java.sun.com/products/javawebstart/developers.html

The complete JNLP specification is at:
http://jcp.org/en/jsr/detail?id=56

The complete JNLP API Javadoc is at:
http://java.sun.com/j2se/1.5.0/docs/guide/javaws/jnlp/index.html