Unofficial Java Web Start/JNLP FAQ

Gerald Bauer

July 22nd, 2002

Overview

This FAQ complements Sun's official Java Web Start FAQ page and aims to provide you with information that is not included on the official page for whatever reason. If you just started with Web Start, please check Sun's Official Java Web Start/JNLP FAQ [`http://java.sun.com/products/javawebstart/faq.html`] page first.

If you want to share your insights, please post it at Sun's Java Web Start and JNLP forum [`http://forum.java.sun.com/forum.jsp?forum=38`] or mail it to comments@vamphq.com. The Unofficial Web Start FAQ is a living, growing list of answers, not just a publish-and-forget-it static dead-tree booklet, help to improve it.

The latest version of the Unofficial Java Web Start/JNLP FAQ resides at Venus Application Publisher's (Vamp) website [`http://www.vamphq.com/jwsfaq.html`] . You also find the original DocBook, HTML and other formats there.

General

Q: What does the support of Web Start in Java 1.4 look like?

Web Start is bundled with Java 1.4.

On Windows, Web Start is installed silently during the installation of the Java 1.4. Look for a Web Start icon on your desktop. There will also be an entry for Web Start in the Start --> Programs menu.

On Solaris and Linux, the installation script for Web Start is contained within a zip file that you can find in the `jre` directory of the JDK (or in the top level of the JRE). Move the zip file to a location where you would like to install Web Start. Sun recommends that this location be outside the JDK or JRE directory structure. Unzip the file and run the `install.sh` script to install Web Start.

Be warned, if you install Java 1.4 Beta 2 it will erase the content of your Web Start cache. You have to download all apps again. Note, this behavior is fixed starting with Web Start 1.0.1_02.

Milestone	Expected Date
Beta 2 (aka Beta Refresh)	early September 2001
Beta 3	end of the year 2001
Release Candidate (RC) 1	beginning of 2002
Final Release	first quater of 2002 (Q1 2002)

Sun released Java 1.4 in mid-February 2002 as scheduled for Linux, Windows and Solaris.

Q: Can Web Start update itself?

Web Start has an update detection mechanism that can inform the user when a new release is available. At the moment the installation is manual. However, Web Start itself (without a JRE) is expected to be a download of 400k or less. Web Start already supports automatic upgrading to newer versions of JREs.

Q: Can I use Web Start for command line/batch apps?

Support for command line/batch apps is currently limited. You can launch apps from a command prompt/shell. Example:

```
javaws http://java.sun.com/apps/draw.jnlp
```

However, you cannot pass in arguments for your app on the command line as `javaws` only takes a single URL as an argument and ignores the rest.

If you want to pass in arguments to your app, you have to add them to the start-up file (aka JNLP descriptor) using `<argument>` or `<property>` elements. Nothing is wrong with "hard-coding" your arguments in the start-up file (aka JNLP descriptor) as long as your arguments don't change.

If your arguments are not known in advance or change from time to time, there are a couple of workarounds to pass your arguments along to your app.

A simple workaround is to create a servlet that takes in your arguments as parameters in the URL (e.g. `http://localhost/apps/notepad.jsp?dir=c:/carrie/chap1/shower.txt`) and adds them to the startup file (aka JNLP descriptor) that it sends along to Web Start. This workaround requires a web server.

(One trick is to make sure *not* to include the `href` attribute in the JNLP file that your servlet sends back to Web Start. This will tell Web Start to disable the update check on JNLP files, and Web Start will not treat each new JNLP file as an application update - only updated jar files will.)

A more esotereric workaround that doesn't require a web server is creating your own JNLP client that allows you to pass along your arguments to your apps. That sounds harder than it is. Building a stripped down JNLP client (e.g. no installer, no applets) using one of the two open-source client as a start takes probably just a couple of days and should be sufficient for in house usage.

A more sopisticated workaround that doesn't require a web server and that doesn't require you to replace Web Start is wrapping your own executable around `javaws` that takes your passed in arguments plus JNLP URL and looks up the original in the cache and patches it or adds a new one to the cache that it passes on to `javaws` to start up the app using your arguments.

The next step would be to create another executable that wraps your wrapper to make a command-line call look like as nothing had happened. Example:

```
javawsex http://jakarta.apache.org/ant.jnlp jar javadoc crossref
javawsex http://java.sun.com/apps/notepad.jnlp c:/carrie/chap1/shower.txt
```

becomes

```
ant jar javadoc crossref
notepad c:/carrie/chap1/shower.txt
```

Q: Can I use Web Start for Intranet apps?

Yes, Web Start is well-suited for Intranet apps. Note, that you can also use `file://` URLs to access the local file system. If you use `file://` URLs you need to use `file:///` because `file://` doesn't work. The third slash is required for the left out host name. Example:

```
javaws file:///c:/sandbox/venus/startup/hazel.jnlp
```

In case your path to the `.jnlp` file contains spaces, enclose the path in double quotes. Example:

```
javaws "file:///C:\Program Files\Java Web Start\
  .javaws\cache\http\Dlocalhost\P80\DMclient\AMdemo.jnlp"
```

Q: Is Web Start available for Macintosh?

Web Start ships with Mac OS X 10.1. You can see it with your own eyes at `http://developer.apple.com/java/javawebstart/`

Note, that Web Start is *not* available for download as a separate package. It is only available with Mac OS X 10.1 or later as a bundle.

Another option is alive and kicking OpenJNLP lead by Kevin Herrboldt at `http://openjnlp.sourceforge.net` which works on all platforms including Mac OS 9 and other JDK 1.1 platforms.

Q: Does Web Start support downloading jars using FTP?

Web Start cannot handle FTP connections because it depends on HTTP headers for determining things like last-modified date. FTP returns no such headers.

The JNLP spec explicitly limits JNLP to the HTTP protocol.

Q: How can I use Web Start and Jini together?

The following is a short excerpt from the book "A Programmer's Guide to Jini Technology" by Jan Newmarch. The full chapter is available online at `http://pandonia.canberra.edu.au/java/jini/tutorial/JNLP.xml`

Jini and JNLP Comparison

On the face of it, Jini and JNLP appear to occupy similar spaces in that they both allow code to be migrated to a client machine and to execute there. The following table summarises some of the differences and similarities

Jini	JNLP
downloads a service	downloads an application
a client must be running to request a service	a browser must be running which calls a helper to start the application
each service looks after itself, independently of any clients	each JNLP file specifies all required parts of an application; if any part changes, the JNLP file must be updated
the client may need to know the location of a lookup service, but not of any service	the user of a JNLP application must know the URL of the JNLP file
no generic client	generic JNLP helper

Combining Jini and JNLP

If a user does not have the client-side of a Jini application installed, then they cannot make use of Jini services. Here is where JNLP can help, by allowing a user to download an application that can run as the client. The user only needs to know a URL for this, and finding URLs is a common experience for most users nowadays. Corporate Web sites, Web search engines, portals and so on are all mechanisms used to find interesting Web sites and download information.

The converse question is: if JNLP is used to find an application, what is the value of Jini in this? The answer to this lises in the distributed management of Jini services. Suppose a JNLP application relies on a number of components put together, say as a collection of packages. The JNLP file has to specify the location of every one of these packages. If one of the packages changes, then the JNLP file has to be updated. It gets more complex if one of the packages changes to become dependant upon another new package: that new package has to be added to the JNLP file. In other words, management of a JNLP application has to be centralised, to the manager of the JNLP file.

Jini, on the other hand, lets every package/service be managed by its own manager. If a Jini service changes implementation, then it can do so without any external consultation, and just re-registers the new implementation with Jini lookup services. If a service changes to use another service, it does not need to inform anyone else about this change. A Jini client does not need to know how services are implemented ot even where they are located.

Jini is not quite management-free: a client may need to know where lookup services reside. On a local network a client can use multicast to locate lookup services, but outside of this local network clients will need to used unicast to find lookup services at known locations. This is still an improvement: Jini lookup services are relatively stable, persistent and stationary services, whereas the services themselves may be transient or unstable.

The combination of Jini and JNLP works like this:

1. A user finds the URL for an application
2. The user downloads the JNLP file which is executed by the JNLP helper
3. The helper downloads the application's jar files, and runs a Java runtime engine with these jar files
4. Properties are set in the JNLP file for Jini lookup services, so the application can locate its required Jini lookup services at runtime
5. When the application needs a service, it finds it from the Jini lookup services and uses it

6. If the service needs to make use of other services, then it can perform its own search for them, without the knowledge of the application

Q: Can I use Web Start to deploy server apps?

Web Start is designed for rich desktop apps and, therefore, has currently limited support for command line apps or mission critical, 24x7x365, always-up, high performance daemons serving thousands of customers simultaneously.

If you want to deploy web apps consider using Web Archives (`.war`).

An installation service for Java Daemons is also in the works as Java Specification Request (JSR) 96. Check out JSR 96 - Java Daemons at `http://www.jcp.org/jsr/detail/96.jsp`. The stated goal is to supply a small container framework for developing and deploying independently running services in order to fill the gap caused by different handling of existing native platfroms.

(Status: Expert Group Formed - Yes, Public Review Underway - No, Community Draft Published - No, Proposed Final Draft Published - No)

Q: Can I use Web Start to deploy apps to mobile devices (J2ME)?

Web Start is designed for rich desktop apps (aka resource hogs) requiring Megas of disk space and very likely doesn't fit on your Java phone unless you attach a Giga hard disk somehow.

If you want to deploy apps to mobile devices consider using MIDlets.

You can try the SavaJe XE operating system for StrongARM devices. SavaJe includes the J2SE 1.3, so it could work with Web Start. You can download a trial copy of SavaJe XE from `http://www.savaje.com`

- Marco Maier

A goal for OpenJNLP is to run on PDAs.

OpenJNLP has been split into a JNLP-launching library and a separate app/GUI. Swing and the GUI have never been required to use OpenJNLP launching, and it's even easier to use as a standalone library now. The OpenJNLP library is currently 48KB in size, should fit on a PDA easily even with the XML parser. Packaging is simplified as well, requiring just the library, SAX2 and NanoXML in the classpath. If you want the full app/GUI you add the app jar.

- Kevin Herrboldt (author of OpenJNLP)

Q: Is Web Start a general installer?

Web Start is different from classic, big blue gradient, pre-Internet, install-and-forget-it, single-shot installers. Web Start is designed for Java apps and, therefore, isn't of any help if you want to install Windows `.exe` apps that require a bunch of registry settings, for example.

Although Web Start doesn't offer offline installation (aka CD installers) out-of-the box, it's not hard to add them. Venus Application Publisher offers more than five different offline installation options that allow follow-up online upgrades. You can find out more at `http://www.vamphq.com`.

Sun has also a general application installation API in the works that seems to drag on forever. Check out Java Specification Request (JSR) 38 - Application Installation API at `http://www.jcp.org/jsr/detail/38.jsp`. The stated goal is to develop Java APIs that will enable cross-platform installation and de-installation of Java apps as well as platform specific apps.

(Status: Expert Group Formed - Yes, Public Review Underway - No, Community Draft Published - No, Proposed Final Draf Published - No)

Q: What's New In Web Start?

v1.2 Beta - June 2002

Web Start Promoted. Note that starting with Java 1.4.1 (aka Hopper) you can no longer download Web Start without a Java runtime (or a Java runtime without Web Start).

- "One-click" auto-install using an ActiveX browser plug-in that pulls down the Java runtime plus Web Start (if needed) and your app (Windope only)
- https support (that is, SSL - secure socket connections)
- roll your own splash screens
- examples, examples, examples (WebPad, random access file demo, extension installer demo, nativelib demo, CORBA RMI-IIOP demo)
- and much more

v1.0.1_02 - February 2002

- unsigned apps display a "Unsigned Window" warning in the window's title bar or status bar similar to the applet's "Unsigned Applet" warning.
- unsigned apps are no longer allowed to access any user-provided properties, but are restricted to properties beginning with `jnlp.` or `javaws.` only. Trying to access other user-provided properties leads to security access violation.
- upgrading from a previous Web Start versions preserves your cache and no longer deletes it

v1.0.1_01 -

- numerous bug fixes

v1.0.1 -

- now available in nine languages: French, German, Italian, Japanese, Korean, Simplified Chinese, Spanish, Swedish, and Traditional Chinese.
- developer's pack includes a servlet that supports bundling of Web Start apps in Web Archives (`.war`), and that supports the version and extension download protocols as well as jar diff generation.
- small installer for Windows available that ships without a Java Runtime reducing its download size significantly (< 1 Meg)
- numerous bug fixes

v1 - December 2000

- Everything is New

Q: Can I distribute Web Start apps without putting them on a web server (aka CD installers)?

Yes, you can. One option is to use the `file://` protocol instead of `http://`.

Vamp (aka Venus Application Publisher) offers a couple of choices such as installing your app directly into Web Start's cache or using single, self-contained jars that include a built-in, ultra light-weight web server that serves up your app's jars from the installer's jar itself.

- *Clio [`http://www.vamphq.com/clio.html`]* - single-jar installer (app served up through built-in web server)
- *Jess [`http://www.vamphq.com/jess.html`]* - single-jar installer (app installed directly into Web Start cache)
- *Celia [`http://www.vamphq.com/cache.html`]* - command line installer to install web archives (`.war`) or client archives (`.car`) directly into Web Start cache
- *Pam [`http://www.vamphq.com/pam.html`]* - package manager that installs web archives (`.war`) or client archives (`.car`) directly into Web Start cache
- *Nullsoft Installer plus Celia [`http://www.vamphq.com/winoffline.html`]* (Windows Only) - single Windows (`.exe`) that installs app directly into Web Start cache

Client archives (`.car`) are standard jars that include all required jars, icons as well as a all required JNLP descriptors including extensions that are needed by an app in a single jar similar to RPM packages.

Q: Can I run Web Start Apps on a headless (monitor-less) UNIX sys-

tem?

Headless (monitor-less) UNIX machines usually have no XWindow installed and, therefore, lack a browser. You can start your Web Start apps from the command line.

Web Start currently cannot run on headless UNIX systems even if you suppress Web Start's splash popup using the undocumented `javaws.cfg.showSplashScreen false` property.

What you can do, however, is to run your GUI-less apps with Web Start clones such as OpenJNLP or NetX that support command-line only launching without any download progress or signature GUI pop-ups.

Q: Has Web Start won any Awards?

- JavaWorld Editors' Choice Award 2002 [`http://www.javaworld.com/javaworld/jw-03-2002/jw-0326-awards-p3.html`]
  - Winner: Best Java Installation Tool
  - Finalist: Most Innovative Java Technology

Q: Is Java Dead On The Desktop?

Java is alive and kicking. To see yourself check out these links:

Sun's bi-monthly Swing Sightings Series [`http://java.sun.com/products/jfc/tsc/sightings/`] shows off many, many Java desktop apps from the Apollo Human Genome Browser to Tejina, a Japanese Handwriting Trainer using the Kunststoff Look And Feel.

JavaOne 2002 talk How to Build an Awesome Java Client with Swing [`http://servlet.java.sun.com/javaone/sf2002/conf/sessions/display-2180.en-85084.jsp`
(74 slides) by Scott Violet, Norbert Lindenberg and Dale McDuffie (Sun)

IBM's Eclipse Standard Widget Toolkit (SWT), a high-performance, low-footprint, cross-platform, open-source alternative to Swing online at `http://www.eclipse.org`

LimeWire, a Gnutella Peer File Sharing App, was Java's first killer app for non-developers tallying up more than ten million downloads from Joe Blocks in 2001 (average weekly download of two hundred thousand and three hundred thousand estimated daily users). Online at `http://www.limewire.com`

Q: What's the catch?

Biggie. Most desktop computer lack a factory pre-installed Java runtime plus Web Start add-on when bought in store. Broadband users need to download the ten mega Java runtime package themeselfs. Dial-up users need to get it on CD/DVD as a one hour download over a 28.8k modem is impratical.

Good News. Mac OS X ships with a factory pre-installed Java runtime plus Web Start. Java 1.4.1 (code-named Hopper) will include AutoInstall, a plug-in for Internet Explorer for Windows that installs Web Start and/or your app on the user's machine over the Internet through a single-click. Java's Plug-In installation automatically adds Web Start as a free bonus. Finally, a ten meg download isn't huge nowadays. It's roughly the size of the Adobe Acrobat Reader plug-in and much smaller than a MPEG movie.

Nuiscance

- Every Web Start app runs in its own Java runtime. Until Java 1.4.1 (Hopper) Java runtimes don't share the core libraries leading to mega-sized memory waste and sluggish startup.
- Certificate authorities (e.g. RSA, Thawte) only issue developer certificates for a year (that is, your certificate expires after a year).

Q: What alternatives exist to Web Start and Java for rich cross-platform, zero-admin desktop apps?

HTML/Javascript (aka Dynamic HTML - DHTML)

pro:

- HTML browser are ubiquitous (pre-installed on all desktop boxes)
- no installation, zero-admin

con:

- excessive roundtrips to server; limited offline support
- lack of interactivity (e.g. no drag and drop; no in-place, real-time validation; no tabs or trees)
- Javascript is designed for one liners, not for programming in-the-large
- web server overload; web server is the workhorse (thin-client)
- most plug-ins/add-ons aren't cross-platform

XUL/Python

see Mozilla (`http://www.mozilla.org`) or Luxor (`http://luxor-xul.sourceforge.net`) for details

XHTML/SVG/XForms

see W3C (`http://www.w3c.org`) for details

.Net Gtk#

see Mono (`http://www.go-mono.com`) for details

Other Open-Source Alternatives : XWT, Sash, etc.

Other Closed-Source Alternatives : Flash, Curl, etc.

Q: Who is using Web Start?

Check out the app directories listed in the link section categorizing hundreds of Web Start apps.

Q: How does Web Start differ from Applets?

Web Start != Applet 2.0/NT. Web Start is *not* a replacement for applets unless you abused applets for mega-sized pop-up windows floating outside of web pages. Applets run *inside the web page* on the browser's one and only Java runtime. In contrast Web Start apps run on its own Java runtime *outside the browser*.

Applets - dynamic content inside a web page in a browser

Web Start Apps - full-blown, stand-alone desktop apps that run without a browser

Q: Can I create my own Java runtime installers without breaching Sun's license?

Yes, you can. Read on for the fineprint.

Legalese demystified: Sun's Java runtime redistribution terms in plain-English:

Disclaimer: I'm not a lawyer, a ballerina, Scott McNeally or Elvis Presley.

You can freely (free-of-charge) redistribute Sun's Java runtime (including Web Start) with your masterpiece (known as value-added app).

Sun also allows you to repackage the Java runtime (so that you can auto-install it with your very own Web Start installer).

Taboos. Sun doesn't allow you to redistribute stand-alone Java runtimes (that is, without your app) on CDs in books or magazines, for example, without asking their licensing department first.

Also Sun doesn't allow you to redistribute pre-release (aka Beta) Java runtimes (e.g. JRE v.1.4.1 Beta) without permission.

Sun's license states that you aren't allowed to "modify" the Java runtime meaning you aren't allowed to change the code (e.g. replacing Sun's `java.net.Socket` class with your very own). Note that, "modify" in legalese doesn't forbid repacking. You can even drop some files from Sun's Java runtime distro if Sun tagged them as optional (vs. required).

Web Server Setup

Q: How can I add JNLP MIME-types to my ISP's Apache Web Server?

If you've read Web Start's docu you know that your web server has to return a special MIME type for JNLP files: `application/x-java-jnlp-file`.

It's not always easy to convince your web space provider to change their web server configuration. If your provider runs Apache (as many do) and hasn't switched off support for the `.htaccess` feature (which is usually true) there's an easier way: Just add these two lines to the `.htaccess` file at the top directory of your site (or create one, if it doesn't exit):

```
AddType application/x-java-jnlp-file    .jnlp
AddType application/x-java-archive-diff .jardiff
```

Now try to load a JNLP file from your site. If you browser still displays the content of the JNLP file instead of starting Web Start, shift-click the Reload button to make sure no caches are involved (this is at least how it works for Netscape).

As an alternative you can check if your web server supports scripts such as PHP or Perl and use it to set the MIME type header dynamically. In PHP this is a one line affair as the example below sent in by Marc Prud'hommeaux shows:

```
<? header ("Content-Type: application/x-java-jnlp-file"); ?>
<jnlp codebase="http://www.L2FProd.com/software/skinlf/jnlp"
      href="demo.php">
  <information>
    <title>Skin LF Java Web Start Demo</title>
    <vendor>L2FProd.com</vendor>
    <homepage href="http://www.L2FProd.com/software/skinlf/" />
    <description>Skin LF Java Web Start Demo</description>
    <offline-allowed />
  </information>
  <resources>
    <j2se version="1.4+ 1.3+"/>
    <jar href="demo.jar"/>
    <extension name="Skin Look And Feel" href="skinlf.php"/>
  </resources>
  <application-desc main-class="javawebstart">
  </application-desc>
</jnlp>
```

In the unlikely case, your provider supports Java Server Pages (JSP), but hasn't JNLP mime type support turned on; you can use the one line JSP page directive below:

```
<%@ page contentType="application/x-java-jnlp-file" %>

<?xml version="1.0" encoding="utf-8"?>
<jnlp codebase="http://localhost" href="hello.jsp">
  <information>
    <title>Hello Web Start</title>
    <vendor>Nobody</vendor>
    <desc>Hello Web Start Example</desc>
  </information>
  <resources>
    <j2se version="1.4+ 1.3+"/>
    <jar href="lib/hello.jar"/>
  </resources>
  <application-desc main-class="HelloWebStart"/>
</jnlp>
```

Installation and Troubleshooting

Q: How can I pre-install an app into the Web Start cache?

A brute force method is installing your app to your local machine and then taking the image and copying it onto different machines.

A better method is using Venus Application Publisher's Web Start Cache Utility code-named Celia. Celia allows you to install packages to your Web Start app cache without downloading them from the original web site. You can package apps you wish to install in jars and feed them to Celia. Celia will honor whatever codebase you specify in the JNLP descriptor and install all necessary files as if they were downloaded by Web Start itself from the original web site. You can find out more at `http://www.vamphq.com/cache.html`

Q: How can I change the Web Start application folder/cache?

Web Start doesn't allow you to change the application folder/cache.

However, Mik Tuver discovered the undocumented property below that lets you change Web Start's app cache. Example:

```
javaws.cfg.cache.dir=c:/Programme/JavaSoft/JWSCache
```

Q: Can I install Web Start silently?

Yes, use the `/silent` command-line flag (Windows only).

Q: Web Start returns a Bad MIME Type error. What's wrong?

This is a very common problem and most likely caused by Web Start's failed attempt to automatically detect your proxy settings. The "Bad MIME Type" error displayed by Web Start is somewhat confusing as it should really be a "Resource Not Found" error.

Your proxy will send back an error page either in HTML or plain text instead of the requested JNLP descriptor if it fails to get the JNLP descriptor requested by Web Start.

You can change your proxy settings for Web Start manually by starting the Application Manager and using the Preference panel. (Select File|Preferences to get there.)

You can use `telnet` to check what is holding back Web Start. Connect to the web server hosting the JNLP descriptor that causes the "Bad MIME Type" error. Once you are connected type in a HTTP GET request to check what the web server serves up. Example:

```
GET /venus.jnlp HTTP/1.0
```

You should get a response similar to the following:

```
HTTP/1.0 200 OK
Date: Wed Jul 18 19:57:33 CDT 2001
Server: Venus Light-Weight HTTP Server
Content-length: 1183
Content-type: application/x-java-jnlp-file

<jnlp [...]>
```

Now that you know that the web server returns the correct MIME type, all you need to do, is figure out your proxy settings.

Note, starting with version 1.2 Web Start accepts JNLP startup files even if they ship with the wrong MIME type badge (e.g `text/plain` instead of the proper `application/x-java-jnlp-file`). However, if you use jardiffs (that is, jar patches that hold only the changes to fix up outdated jars and turning them into shiny new jars sporting the latest and greatest bells and whistles) Web Start requires the proper MIME type to differentiate between plain-vanilla jars (`application/x-java-archive`) and jars holding jardiffs (`application/x-java-archive-diff`).

Q: Does Web Start support Mozilla?

Yes, it does. Mozilla works out of the box on Windows. Unfortunately, you have to register Web Start manually as a helper app to Mozilla on non-Windows boxes. Here are the steps:

- Select Preferences -> Navigator -> Helper App
- Add the following settings

Field	Value
extension	`jnlp`
mime type	`application/x-java-jnlp-file`
handled by application	`<path-to-javaws>/javaws.exe`

Note, that Sun's JavaScript Web Start detection script doesn't work for Mozilla. Adding Sun's JavaScript Web Start detection script to your HTML page is extremley short sighted and bad practice anyway and should be avoided. If you want to launch Sun's Web Start demo apps, use the no JavaScript page version instead of the malicous JavaScript page.

Q: Why can't Web Start and Microsoft Proxy Server get along?

Here are some postings showing Microsoft at its best:

If your Web Start client is behind a Microsoft Proxy Server and user authentification is switched on, your Web Start Java app never starts. It will work with anonymous authentification, however.

At some customer site the starting app stops when it tries to transmit serialized objects. On other sites we get an 407 "Proxy Authenfication required".

In the Web Start docs we can read "... Web Start will also prompt you for a user name and password required to access an authenticating proxy server. " Did anybody ever see this dialog with Web Start and Microsoft Proxy Server?

Microsoft Proxy Server uses NTLM authentication to authenticate its clients. Unfortunately, NTLM authentication is based on a Microsoft proprietary protocol and there is no official specification. Although NTLM authentication is similar to the standardized and widely-used BASIC authentication, it is not exactly the same and clients expecting a BASIC authentication handshake will fail unless they have a special WinSock Proxy Client installed (another Microsoft component, of course).

My guess is that Web Start is only able to handle BASIC authentication (which would definitely make sense). In any case, it cannot implement a handshake for NTLM because there is no official specification.

Web Start works fine with a proxy using BASIC for authentication.

For more information on NTLM see

- `http://www.innovation.ch/java/ntlm.html`
- `http://www.pla-netx.com/linebackn /evil/msproxy.html`

If anyone else than me is interested in trying to write some code which will accomplish this, please drop me an email at andda715@student.liu.se. Then we could set up a project at sourceforge.net and see what happens.

- Anders Dahlberg

Btw, I've found some c code which does this (a mod to apache), I'm sure we can copy this code if we ask nicely :).

Dmitry Rozmanov has created a free, open-source NTLM authorization proxy server in Python that supports Microsoft's proprietary NTML protocol. Features include:

- NTLM authentication via parent proxy server (Error 407 Proxy Authentication Required)
- NTLM authentication at web servers (Error 401 Access Denied/Unauthorized)
- HTTP 1.1 persistent connections
- HTTPS CONNECT for transparent tunneling through parent proxy server

- and more

See `http://www.geocities.com/rozmanov/ntlm/` for details.

The NTLM authorization issue is also logged at Sun's Bug Parade at `http://developer.java.sun.com/developer/bugParade/bugs/4423881.html` .

Sun's policy is *not* to support Microsoft's proprietary NTLM authentication. Instead Sun's suggests that you upgrade to Microsoft Internet Accelerator Server (IAS) as it supports HTTP digest authentication and use it instead of the preconfigured, Microsoft-only NTLM.

HTTP digest authentication like NTLM authentication no longer sends plain-text passwords over the wire as HTTP basic does. You need to upgrade from Microsoft Proxy Server to the rebranded Internet Accelerator Server as HTTP digest authorization is not supported in the older Microsoft Proxy Server.

Luigi Dragone wrote a 100% pure Java implementation of the Micropoly NTLM authentication protocol (released under the GNU GPL). Jump to `http://www.luigidragone.com/networking/ntlm.html` for details.

Q: Can I install Web Start on Windows without admin privileges?

Not out-of-the box using Sun's Web Start installer. Here are some roadblock you have to overcome:

- Web Start currently doesn't allow you to change the cache directory location after installation or to specify your own directory during installation. Web Start's cache directory (`.javaws/cache`) is automatically derived from Web Start's installation directory.
- Web Start needs write-access to the registry as it add several registry entries to register Web Start as a browser helper app for jnlp extenstions and mime types.

Here are some voices from the trenches:

We had no luck allowing the users to install, unless they were give admin priv, which we did temporarily. That worked fine.

We also thought we would pre-install it on the boxes but it seems that it will only work under the same user id that did the install. So we can't have an admin account load it up for everyone. Each user has to do it when they get their box, and has to have admin help when they do it. Kind of the worst situation since it requires the user and the admin to do it.

---

1. Because of the registry settings the user needs at least "main user" properties for the installation to be successfull (i.e. setting the MIME type, etc.). Unfortunately, the Web Start app manager icon is put into the installing user's desktop folder, so that only this user will have the icon on his desktop

and hence be able to start web start without problems. Our admins copy the icon to the All Users desktop so that everybody "has" it. Alternatively, you can give the user instructions how to acess web start or create a Web Start desktop icon. Having the installer put the shortcut to the All User's desktop folder may solve this problem.

2. In our configuration the program folder (`C:\Program Files`) is read-only for normal users, so that they cannot install software. Because the cache of Web Start is located below the install directory, which is the program folder by default, normal users cannot let Web Start write to the cache, and therefore cannot download applications.

We solved this problem by installing Web Start *not* to the program folder. This cannot be done automatically with the silent option, because there is not command line parameter through which you can specify the installation directory. So we do it manually for each Web Start installation (or use a standard hard disk image for standard configurations, where Web Start is included).

This problem could be solved, by moving the Web Start cache to the user's area or by giving us the possibility to specify an install directory from a command line using the silent install. Nullsoft has apparently this option (-D), but somehow, I am not successfull to carry it out. I asked already that the Nullsoft Installer script is being published so that everybody could modify it to his need. No answer yet to this topic.

- Thomas Gulden

What is possible under Windows, is allowing access to certain registry keys for all users: `http://www.microsoft.com/office /ork/xp/one/deph02.htm`

As a workaround, you can throw out Windows and replace it with Linux. Under Linux it is possible to do all necessary steps as common user. The cache will be put in the user's home directory, the "mime-type->file extension" association is set up in the user's `.mimetypes` file and the "mime-type->application" association in the user's `.mailcap` file.

Thomas Gulden reports on trouble with shortcuts on multi-user Windows machines using a single Web Start install:

If more than one user works on a machine, the shortcut creation for the second user (lets call her Betty) (not the lucky girl who installed Web Start) faces some bad karma.

1) When Betty logs in she does not see any icon for a downloaded Web Start app. This is correct, because Web Start creates the shortcut in the user's home directory leaving Betty to create the icon herself. When she uses the menu, she only finds a menu item allowing her to delete the shortcut (even though there isn't any). This is because Web Start maintains only a global property to track if it has created icons or not. This seems to be inconsistent design as Web Start should store the property in the user's home directory for each user instead of once for all.

2) After deleting the shortcut Betty can now invoke "create shortcut" again. If she dares to do so, a shortcut shows up on the desktop sporting Web Start's default icon instead of the app's very own. We found out that this is caused by the user privileges assigned to the icon in Web Start's cache. Only the user who first installed the app has the right to access it and this restriction seems only to apply for the app's icon,

the rights for all other files are set to everyone.

Q: Why doesn't Sun's Javascript Web Start detection script work for my household browser?

Sun's Javascript Web Start detection script works only with Netscape 4.x and Internet Explorer. If you use Netscape 6.x, Mozilla, Opera or other browsers, Sun's Web Start detection script fails because Netscape 6.x, Opera and Mozilla, for example, implement `navigator.mimeTypes` different than Netscape 4.x.

In conclusion, don't include Sun's Javascript Web Start detection script into your HTML page unless you work for Microsoft, instead point directly to the jnlp file. Using Sun's Javascript Web Start detection script not only blocks all browsers other than Internet Explorer but also hinders HTML parsers to retrieve the embedded jnlp links, in case you want to use the HTML page as Web Start's default app page, for example.

Q: How can I change the proxy configuration in javaws.cfg?

If you want to set the proxy setting to none, use

```
javaws.cfg.proxy.setting=NONE
```

Note, that under Windows there are two `javaws.cfg` files. `javaws.cfg` can be in `"\Program Files\Java Web Start\javaws.cfg"` which will get used for all users on that machine or it can be in the user's profile directory in `".javaws\javaws.cfg"`. Web Start's App Manager will put your settings in the user's profile, but you can actually put it in either.

Q: Why doesn't my app run under Web Start?

Note, that Web Start only ships jars to the user's desktop. You need to retrieve all your resources either from jars or over the network using HTTP.

To find out what holds back your app, you might follow these steps:

Check if you packed up everything your app needs into jars by trying to run it from the commandline without any classpath settings using the `-jar` switch. Example:

```
java -jar myapp.jar
```

Try to fire up your app with Web Start using a `file:///` codebase in your jnlp file. Using `file:///` spares you from uploading your jars and setting up your own web server. Example:

```
<jnlp href="myapp.jnlp" codebase ="file:///c:/sandbox/myapp/startup" >
```

Now you are almost there. If Web Start cries "Resource Not Found" when trying to download your jnlp file or jars check if your web server's jnlp MIME types are set up correctly. You can check up on your web server by using `telnet` and sending a GET request to your web server. See "Web Start returns a Bad MIME Type error. What's wrong?" for details.

If your web server's mime types are setup correctly and Web Start still refuses to start up your app , you need to figure out your proxy settings.

Q: Why can't I start Web Start's app manager under Windows? Why does the app manager's preferences dialog fail to appear?

This might be a proxy issue. Try to turn off your proxy by putting

```
javaws.cfg.proxy.setting=NONE
```

in your `javaws.cfg` file and see if it Web Start's app manager shows up.

Web Start checks your Windows registry to find out your browser's proxy settings.

You can extract `com.sun.javaws.proxy.WinInternetProxy` from `javaws.jar` and run Cygwin `strings` on it to find out what keys Web Start is looking for. If you're on a different platform, try a similar hack with whatever `com.sun.javaws.proxy` class shows up in your stack trace.

Here are the registry keys for Internet Explorer:

```
HKEY_CURRENT_USER/Software/Microsoft/Windows/CurrentVersion/Internet
Settings
```

- `AutoConfigURL`
- `ProxyEnable`
- `ProxyServer`
- `ProxyOverride`

If you upgraded to Internet Explorer 5, Internet Explorer 4 might have left bind bad values in `AutoConfigURL`. Delete `AutoConfigURL` and see if it makes a difference.

Q: Why doesn't Web Start reprompt for proxy logins?

Q: When I start my app and enter a wrong password in the proxy login dialog poped up by Web Start my app starts. However, whenever I use `HttpUrlConnection` and read the response an exception gets thrown saying "too many server directs".

Why doesn't Web Start pop up the dialog again if the password to the proxy was wrong? How can I tell Web Start to popup up the dialog again without rebooting the app?

A: I downloaded the Web Start source code. Web Start's proxy login dialog (aka `JAuthenticator`) gives you *one shot* to type in your right password. If you get it wrong, Web Start won't reprompt for another try.

Q: Can Web Start handle multiple XML parsers at once?

Web Start uses JAXP for parsing XML. I want to use Xerces 1.1.3. However, `xerces.jar` includes newer DOM classes that conflict with the older DOM classes shipped with Web Start. I also use SOAP which depends on newer DOM classes as well. When my app makes a SOAP call that uses a method available only in the newer DOM classes, Java's runtime throws a `NoSuchMethodError` because it is using the older DOM classes from `jaxp.jar` rather than the newer ones from `xerces.jar`.

Shouldn't Web Start use a separate ClassLoader to isolate my app's jars from Web Start's own to avoid class loading conflicts?

I face a similar problem. The following line throws `ClassNotFoundException`:

```
org.xml.sax.helpers.ParserFactory.makeParser("com.ibm.xml.parsers.SAXParser");
```

I can work around the multiple xml parser conflict by placing the XML jars (`xerces.jar` and `xml4j.jar`) in the runtime's `lib/ext` directory. This works for internal testing, but is not much use otherwise.

I've hit the same dead end using Web Start 1.0.1 under Mac OS X 10.1.

Web Start's jar (`javaws.jar`) contains what seems to be a dumbed-down version of the minimum XML parsing needed to interpret jnlp files.

Unfortunately, it's inadequate for my needs that require Xerces 1.4.3. Web Start's built-in XML package lacks (among many others) `Document.importNode()`, alas. `NoSuchMethod` errors abound. Sigh.

I'm going to experiment with custom class loaders to see if I can find a workaround.

If you want to use a specific XML parser bundled with your app, instead of whatever XML parser happens to get picked up by JAXP, you can set the property `javax.xml.parsers.DocumentBuilderFactory` in your JNLP startup file. Example:

```
<resources>
  <j2se version="1.3"/>
  <jar href="xerces-1.4.3.jar"/>
  <property name="javax.xml.parsers.DocumentBuilderFactory"
            value="org.apache.xerces.jaxp.DocumentBuilderFactoryImpl"/>
</resources>
```

Q: How do I know that jardiff is working?

To make jardiff work you need to add version attributes to your `<jar>` tags in your jnlp startup file. Example:

```
<jar href="/jars/AccentVic.jar"  version="1.1"/>
```

On the server you need to use Sun's JnlpDownloadServlet to let it handle Web Start requests for your jars as a plain Web Server won't suffice. You can get JnlpDownloadServlet for free by downloading

Sun's JNLP developer's pack [http://java.sun.com/products/javawebstart/download-jnlp.html].

You also need to tell Sun's JnlpDownloadServlet what version a jar belongs to. You can use either a name mangling convention that adds the version directly to your jar's filename (e.g. AccentVic__V1.1.jar) or add versions entries to version.xml in the directory holding your jars.

To check if everything works turn on JnlpDownloadServlet's debug log. Add two init parameters to JnlpDownloadServelt's config file. Example:

```
logLevel=DEBUG
logPath=<log file name>
```

After a Web Start request you should find something like below in your servlet log file:

```
JnlpDownloadServlet(3): Resource returned: /jars/AccentVic1.jar
JnlpDownloadServlet(3): Generating JarDiff for Accent70Vic.jar 1.0->1.1
JnlpDownloadServlet(4): Generating Jardiff between
    /usr/WebSphere/AppServer/hosts/Accent/WebApp/jars/Accent70Vic.jar and
    /usr/WebSphere/AppServer/hosts/Accent/WebApp/jars/AccentVic1.jar Store in
    /tmp/jnlp29099.jardiff
JnlpDownloadServlet(4): JarDiff generation succeeded
JnlpDownloadServlet(3): JarDiff returned for request
```

This tells you that JnlpDownloadServlet just dished out a jardiff and everything works as advertised.

Q: How can I install Web Start on Unix as root once for all users?

Brett Humphreys solved the puzzle. Web Start updates the current users' .mime.types and .mailcap files. However, if you install Web Start as root (that is, once for all users), it won't update netscape's global files, so you have to do it yourself.

Here are Brett Humphreys's distilled instructions to register Web Start as Netscape's helper app for JNLP mime types as root once for all users.

Follow Web Start's install instructions, including

1. Downloading the zip file for Solaris
2. Unzipping the zip file to a logical global install directory (e.g. /usr/local)
3. run the install.sh script to install

One of the last messages the install script spits out is:

```
Updating ~/.mailcap...
Updating ~/.mime.types...
```

This updates the MIME types and mailcap files for the current user only; use the instructions below to update them for all users.

These instructions assume `/opt/NSCPcom` as Netscape's install directory. To find out where Netscape hides on your machine, type:

```
which netscape
```

This should reveal a directory path telling you where Netscape's executable hangs out, for example:

```
/opt/NSCPcom/netscape
```

As root, open the file `/opt/NSCPcom/etc/mailcap` (`/usr/local/lib/netscape/mailcap` on Linux) and append the line below to register Web Start as Netscape's helper app for JNLP's mime type:

```
application/x-java-jnlp-file; /export/home/bretth/javaws/javaws
```

`/export/home/bretth/javaws/javaws` tells Netscape where the Web Start executable hangs out. If Web Start frequents a different directory on your machine, change it accordingly. Save your changes and close the file.

As root, open the file `/opt/NSCPcom/etc/mime.types` (`/usr/local/lib/netscape/mime.types` on Linux) and append the line below:

```
type=application/x-java-jnlp-file desc="Java Web Start"
exts="jnlp"
```

Close any running Netscape browser windows and restart it to kick start Web Start apps.

Q: Why does Web Start use two javaws.cfg files?

Web Start uses two `javaws.cfg` files.

The first one resides in Web Start's installation directory and holds the initial settings (mainly the location of Java runtimes available at installation time.)

The second one resides in the users "home" directory (whatever the system property `user.home` returns varies from Linux to Windows). This config file holds the user's configuration changes such as proxy settings, new or changed Java runtimes, logging, console, and other preferences.

Q: Why Can't Web Start Create Desktop Shortcuts?

Web Start uses the app's title stored in the jnlp startup file (e.g. `<title>Venus Application Publisher<title>`) to create the desktop shortcut. Make sure your app's title doesn't contain illegal characters (such as `\ / : * ? " < > |`), otherwise you end up with the Web Start popup: "unable to create shortcut, try again later". Also check your title's length as too long titles cause problems as well (under Windope you can use up to 255 characters).

Q: How can I autoinstall Web Start?

Franklin Schmidt has done it using Internet Explorer and the Java Plug-In. See yourself at `http://www.myquerytool.com/download/`.

Franklin Schmidt uses JavaScript to check if Web Start is installed. If it isn't, he links to a "automated Web Start install" page looking like this:

```
<html>
  <body>
  <h1>Installing Java</h1>
  <OBJECT
     CLASSID="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
     width="0" height="0"
     codebase="http://java.sun.com/products/plugin/autodl/jinstall-1_4_0-win.cab#Version=1,
    <PARAM NAME="code" VALUE="Redirect.class">
    <PARAM NAME="type" VALUE="application/x-java-applet">
    <PARAM NAME="where_to_next" VALUE="/download/?a=b">
    You did not install Java.
    <p>Please use the <b>Back</b> button to return to the previous page.
  </OBJECT>
  </body>
</html>
```

This magic HTML snippet installs the Java Plug-In plus Web Start. Once installed it runs the "Redirect" applet below taking the user back to the download page.

```
public class Redirect extends Applet
{
  public void start()
  {
    try
    {
      getAppletContext().showDocument(
        new URL(getCodeBase(),getParameter("where_to_next")) );
    }
    catch(MalformedURLException e)
    {
      throw new RuntimeException(e);
    }
  }
}
```

Sun has added autoinstall to Web Start 1.2 for Windows upcoming in fall 2002 (betas expected this summer) with Java 1.4.1 (aka Hopper).

Q: What autodownloadable Java Runtimes does Sun offer?

Sun offers:

- Java Runtime 1.3.0_02 for Windows 95/98/NT/2000
- Java Runtime 1.3.0_02 for Solaris/SPARC
- Java Runtime 1.3.0_02 for Linux/i486

Source: `http://java.sun.com/products/javawebstart/developers.html#auto`

Sun has not yet posted Java bundles for 1.4 or even 1.3.1 on their site. To lobby Sun cast your vote at Sun's bug parade at `http://developer.java.sun.com/developer/bugParade/bugs/4638667.html`

Q: How can I change the splash screen?

Web Start uses two splash screens. A mini splash screen (`miniSplash.jpg`) that pops up before Web Start kicks off the Java runtime and gets torn down once the Java runtime takes over. A progress splash screen that tells you what Web Start is up to (downloading, checking for updates, checking signatures, etc.) and what app is about to popup on on your desktop.

You can turn off the mini splash screen using the `javaws.cfg.showSplashScreen` setting in the Web Start configuration (`javaws.cfg`). See "How can I turn off Web Start's splash screen?" for details.

Using brute force, you can sneak in your very own splash screen by overwritting Sun's `miniSplash.jpg` image.

Starting with Web Start 1.2 and greater you can now choose your own mini splash screen image in your app's XML startup file using the `<icon>` tag. Example:

```
<icon kind="splash" href="venus.gif" />
```

Note, however, that Web Start will frame your image on the first start up. For all following startups Web Start will popup the freshly minted splash screen showing off your very own image.

Also note that the Web Start Jukebox (that is, Application Manager) uses its own splash screen (`splash.jpg`).

Q: How can I turn off Web Start's splash screen?

The Splash Screen Firewall Dead-Lock. If your firewall forbids opening connections on random ports, Web Start likely gets stuck in an endless loop trying to get through to the splash screen server.

To keep you entertained while the Java runtime starts offs Web Start kicks off a tiny, turbo-charged, non-Java, native app (`splash.exe`) that pops up a splash screen in an instant. Once the Java runtime is up and running Web Start tells the splash screen to shutdown. The magic works because the tiny splash screen app is a full-blown TCP server listening on a random port waiting for the shutdown signal.

To avoid this dead-lock scenario, turn off the splash screen by adding `javaws.cfg.showSplashScreen=false` to your Web Start configuration (`javaws.cfg`).

If Web Start still hangs, dive into the Windows registry and change the `.jnlp` file type setting by adding the extra `-Xnosplash` switch for launching (for example `"C:\Program Files\Java Web Start\javaws.exe" "-Xnosplash" "%1"`).

Q: How can I create a shortcut on the first launch?

By default Web Start pops the desktop-shortcut-icon-creation question on the second launch.

If you want Web Start to pop the question on the first launch, use the Web Start Wurlitzer (that is, the Application Manager) to adjust the setting. Note, you can also choose "always create shortcuts" or "never install shortcuts".

You can also adjust the desktop-shortcut-icon-creation setting directly in the Web Start configuration (`javaws.cfg`) instead of klicking around in the wurlitzer. Use the `javaws.whenInstall` property and set the value to your liking: 0 - Always; 1 - Ask on first startup; 2 - Ask on second startup; 3 - Never. Example:

```
javaws.whenInstall=1
```

Q: How can I auto-download an international Java runtime?

Web Start passes your computer's locale (that is, language and country code, for example `en_US` or `de_AT`) to Sun's auto-download servlet. For `en_US` locales Sun's servlet dishes out the US English version, for all other locales Sun's servlet dishes out the international version.

Note, that you can only download an international version if you set the locale on your machine to something other than US English (`en_US`). No magic product version string in the JNLP startup file exists to get you an international version for a US English locale or a US version for a non-US locale from the Sun site.

Note, also that Web Start will not even contact Sun's auto-download servlet if your app requires an international version when your computer already has an US version installed.

As a workaround you can roll your own Java runtime installer that dishes out a international version no matter what locale Web Start sends over. Use a Java runtime product version to request an international version. Example:

```
<j2se version="1.4+" href="http://cloud7.org/autodl/j2se-int" />
```

Q: What's the best way to move my Web Start apps to a different server?

If you move your Web Start app to a different server (that is, different codebase in Web Start parlance), your users will likely end up ruminating why they now have two identical icons for your wonderful app on their desktop and how they differ.

To help your user differentiate your apps hosted on the old server (e.g. `cloud7.com`) from your apps hosted on the new server (e.g `cloud8.com`), use a different title and description in your JNLP startup file (for example, use "Whack-A-Bill @ Cloud Seven " and "Whack-A-Bill @ Cloud Eight" instead of just "Whack-A-Bill" for both.)

Tell your users to fire off the Web Start Jukebox (that is, application manager in Web Start parlance) and delete your old tune from the playlist (that is, downloaded apps or favorite apps).

Note, if you delete your app from the playlist, Web Start won't clean up its cache and throw out the junk (that is, the now orphanded jars, icons, and so on), instead Web Start leaves the cache surgery up to you.

Open up the hood (that is, the Web Start cache directory e.g. `c:/java/jws/v101/.javaws/cache`) and delete the branch in the directory tree that holds your old app parts (e.g.`http/Dwww.cloud7.com/P80/DMapps` translating to `http://www.cloud7.com:80/apps`).

HTTP Redirects. Note, you can also use HTTP redirects such as 301 Moved Permanently or 307 Moved Temporarily.

Resource Loading

Q: How can I load resources from a jar?

Check out my resource loading tutorial. You can find it at `http://www.vamphq.com/tutorial.html`.

Q: How can I load resources specified in a property file?

Use the `class://` URL protocol as you can't use absolute or relative file paths. For a detailed explanation check out Rachel, an open-source Resource Loading Toolkit for JNLP/Web Start. It ships with user docs and examples. Note, that your apps will also work without Web Start. You can find Rachel at `http://rachel.sourceforge.net`

Q: Can I change my apps look and feel?

Yes, you can. However, there are some limitations and workarounds you should be aware of. Check out the following bug reports:

- `http://developer.java.sun.com/developer/bugParade/bugs/4432700.html`
- `http://developer.java.sun.com/developer/bugParade/bugs/4155617.html`

Here is the code fragment for switching to the Kunstoff look and feel:

```
try
{
  com.incors.plaf.kunststoff.KunststoffLookAndFeel kunststoffLF
    = new  com.incors.plaf.kunststoff.KunststoffLookAndFeel();
  kunststoffLF.setCurrentTheme( new com.incors.plaf.kunststoff.KunststoffTheme() );
  UIManager.setLookAndFeel( kunststoffLF );
}
catch ( javax.swing.UnsupportedLookAndFeelException x )
{
   // handle exception
}

// make Web Start happy
// see http://developer.java.sun.com/developer/bugParade/bugs/4155617.html
```

```
UIManager.getLookAndFeelDefaults().put( "ClassLoader", getClass().getClassLoader()  );
```

I am not sure, if this is a workaround to something that should work by itself.

Q: How can I load Skin Look and Feel themes from a jar?

Use the `class://` URL protocol. Example:

```
URL themepack = new URL( "class://ThemeAnchor/skinlf/themes/modern.zip" );
Skin skin = SkinLookAndFeel.loadThemePack( themepack );
```

For a detailed explanation check out Rachel, an open-source Resource Loading Toolkit for JNLP/Web Start. It ships with user docs and examples. You can find Rachel at `http://rachel.sourceforge.net`. In case you don't know SkinLF, you can find out more at `http://www.l2fprod.com`.

Q: How can I reference a Java Help helpset?

Note, that you cannot use absolute or relative file paths. Instead pack your helpset in a jar and use the resource anchor trick to reference it. See the Resource Loading Tutorial [`http://www.vamphq.com/tutorial.html`] for details. Here is a code snippet that loads a helpset:

```
public class HelpSetResourceAnchor
{
   // do nothing; add it to the jar with all your helpsets
}

public class HelloHelp extends JFrame
{
  JHelp   _browser;
  HelpSet _helpset;

  public HelloHelp( String topic_id )
  {
   try
   {
     // use resource anchor trick to reference helpset
     ClassLoader cl = HelpSetResourceAnchor.class.getClassLoader();
     URL url = cl.getResource( "hello.hs" );
     _helpset = new HelpSet( cl, url );

     _browser = new JHelp( _helpset );
     _browser.setNavigatorDisplayed(true);

     getContentPane().setLayout(new GridLayout());
     getContentPane().add( _browser);

     setSize(500, 600);
     setLocation(0, 0);
     setTitle( "Hello Help");

     showHelp( topic_id );

     setVisible(true);
  }
  catch (Exception ex)
  {
     System.out.println("*** error: " + ex.toString() );
  }
}

public void showHelp( String topic_id )
{
  try
  {

   if( topic_id.equals( "" ) )
      topic_id = _helpset.getHomeID();

   _browser.setCurrentID( topic_id );
  }
  catch (Exception ex)
  {
     System.out.println("*** error: " + ex.toString() );
  }
}
```

Q: How can I list all resources in a jar?

Java doesn't currently support a method to list all resources in a jar. You can only retrieve one resource at a time.

What you can do, however, is create your own index page and retrieve it from your jar so you know what treasures it holds. Here is an example how an index page looks like. Feel free to invent your own format:

```
template.1=template/brief.htt
startup.1=startup/icon.xul
startup.2=startup/key.xul
startup.3=startup/menu.xul
startup.4=startup/toolbar.xul
images.1=images/about.gif
images.2=images/blank.gif
images.3=images/book.gif
images.4=images/exit.gif
images.5=images/folder.gif
images.6=images/world.gif
```

Q: Can I use my own custom ClassLoader?

Yes, you can. Note, that Web Start uses its own class loader. If you want to delegate class loading from your class loader to Web Start's class loader use:

```
ClassLoader wcl = Tool.class.getClassLoader();
URLClassLoader cl = new URLClassLoader(urls, wcl);
```

Where `Tool.class` is a class loaded by Web Start's own class loader (e.g. your entry point class holding `main()` packed up in a jar listed in your JNLP startup file).

Don't use `ClassLoader.getSystemClassLoader()` as a delegate, e.g.

```
ClassLoader scl = ClassLoader.getSystemClassLoader();
```

```
URLClassLoader cl = new URLCLassLoader(urls, scl);
```

unless you want to exclude all jars listed in your JNLP startup files from the search path. Note, that if you don't pass in a parent class loader (aka delegate) to your own custom class loader Java will automatically pick the system class loader as your custom class loader's delegate.

As an alternative you can set your own URLClassLoader as the context class loader for the event dispatch thread (aka Swing GUI thread) at the beginning of your app. Example:

```
public static void main(String[] args)
{
  ...
  ClassLoader wcl = Tool.class.getClassLoader();
  URLClassLoader cl = new URLClassLoader(urls, wcl);
  try
  {
    EventQueue eq = Toolkit.getDefaultToolkit().getSystemEventQueue();
    eq.invokeAndWait(new Runnable() {
      public void run() {
        Thread.currentThread().setContextClassLoader(cl);
      }
   });
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
  ...
}
```

If you want that Web Start downloads, caches and updates your jars but doesn't load them, you can add your jars to a jar that's listed in your JNLP startup file. Example:

```
Contents of plugins.jar:
  bookmark.jar
  history.jar
  PlugInAnchor.class  // one class need for identification
```

You can use your own class loader (based on `URLClassLoader`) to load the classes packed up in the jar's jar according to your own policies.

JNLP Descriptor (aka Start-Up File)

Q: Why can't Web Start find my main class?

Make sure your main class is the *first* jar listed in the `resources` section. As an alternative you can set the jar's main attribute. (See JNLP Tag Reference [`http://www.vamphq.com/jnlpref.html#jar`] for details.) Example:

```
<jar href="lib/mailicep.jar" main="true" />
```

Q: How can I specify a relative codebase in the JNLP descriptor?

Use Sun's jnlp-servlet that is part of Sun's Web Start developer's pack. It allows you to set `jnlp.codebase` to `$$codebase` and `jnlp.href` to `$$name`. Sun's jnlp-servlet will replace `$$codebase` and `$$name` with the correct values at run-time. Sun's jnlp-servlet also supports different versions of the same file and jar diffs (aka incremental updates).

A more light-weight approach is to rewrite your JNLP as a JSP. This allows you to put the JNLP file on any JSP-enabled host you want. Note that both `codebase` and `href` in the example below are pulled from the request and are *not* hard-coded.

```
<%@ page
 contentType="application/x-java-jnlp-file"
 info="My JNLP"
%>

<%
 StringBuffer codebaseBuffer = new StringBuffer();
 codebaseBuffer.append(!request.isSecure() ? "http://" : "https://");
 codebaseBuffer.append(request.getServerName());
 if (request.getServerPort() != (!request.isSecure() ? 80 : 443))
 {
   codebaseBuffer.append(':');
   codebaseBuffer.append(request.getServerPort());
 }
 codebaseBuffer.append('/');
%>

<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+"
      codebase="<%= codebaseBuffer.toString() %>"
      href="<%= request.getRequestURI() %>">
[...]
```

Note, that you can do the same in Perl, PHP or whatever CGI scripting language you prefer.

Q: How can I pass in arguments to my app using a HTML hyperlink?

One trick is to make sure *not* to include the `href` attribute in the JNLP file that your cgi-script sends back to Web Start. This will tell Web Start to disable the update check on JNLP files, and Web Start will not treat each new JNLP file as an application update - only updated jar files will.

This trick gives you complete freedom in what your URL looks like. Example:

```
<a href=
 "http://localhost/apps/notepad.jsp?dir=c:/carrie/chap1/shower.txt">
 Notepad<a>
```

This trick, however, has some limitations. Because Web Start will not cache the startup file (aka JNLP descriptor), you cannot run your app offline or through Web Start's app manager.

If you want to include a `href` attribute in your startup file (aka JNLP descriptor) , read on.

For a live real-world example check out Java Zoom's (MP3 music player with Winamp skins) online JNLP configuration service at `http://www.javazoom.net/jlgui/jnlp_configurator.jsp`

There is a limit to what you can do with dynamically generated JNLP files. The combination of `codebase` and `href` must exactly specify the JNLP resource that you are retrieving. If you're using a servlet with parameters in a query string to generate the file, it should look like this:

```
<jnlp spec="1.0+"
      codebase="http://app.datadevelopment.com/app"
      href="MyServlet?param1=value1&param2=value2">
```

As long as the MIME type returned by the servlet is `application/x-java-jnlp-file`, Web Start will start just fine. Unfortunately, Web Start will then bomb badly because it tries to use the `href` attribute as part of the file name for saving the file, and Windows at least doesn't like question marks in a file name. A workaround is to replace the query string with path values:

```
<jnlp spec="1.0+"
      codebase="http://app.datadevelopment.com/app"
      href="MyServlet/param1=value1/param2=value2">
```

I'm not 100% sure that this will work with the '=' sign as my application needed only one parameter so I dispensed with the `paramN=valueN` and replaced the directory with `valueN` only. The point is, though, that each and every time the application is launched Web Start will go to the web server to get the latest copy of the JNLP and the only way I have found to dynamically generate the content is to append the necessary parameters as path names after the script (JSP in my case, servlet in your case).

If you have multiple parameters and the '=' sign is a problem, strip out the parameter name and require that the parameters be specified in the same order each time. Use one directory delimiter per parameter.

You can use `request.getPathInfo()` to retrieve the path after your servlet.

Micheal Mandel has posted a servlet that can replace the four macros below in your JNLP startup file.

Macro	Description
`$$codebase`	same as Sun's JnlpDownloadServlet macro
`$$href`	same as Sun's JnlpDownloadServlet macro
`$$title`	allows you to dynamically change the app's title - the servlet checks if a parameter in the request URL matches a key for a title defined in the servlet's `web.xml` config file
`$$properties`	allows you to dynamically add arguments through properties; the servlet adds the request URLs query string parameters as properties to your JNLP startup file (if `parse.querystring` in the servlet's `web.xml` config file is set to true)

Macro Usage Example:

```
<jnlp spec="1.0+" codebase="$$codebase" href="$$href">

  <information>
    <title>$$title</title>
    <offline-allowed/>
  </information>
  <security>
    <all-permissions/>
  </security>

  <resources>
    <j2se version="1.4+ 1.3+"/>
    <jar href="myapp.jar"/>
    $$properties
  </resources>

  <application-desc main-class="MyApp"/>

</jnlp>
```

You can find Micheal Mandel servlet's source in the appendix.

Q: What values can I use for os and arch?

You can find a list of `os` and `arch` values at `http://www.vamphq.com/os.html`

Q: Can I use native libraries for my apps?

Yes, you can.

To roll a jar with native libs, put all your native libs in the topmost directory ("/"). This results in a flat jar, that is, a jar with no subdirectories except for `MANIFEST` that carries all that administrative stuff like entry method and cryptographic hashes.

Adding your native libs to the jnlp is not sufficant. You also need to load the libraries in your java code as demanded by the jnlp spec:

> It is up to the launched application to actually cause the loading of the library (i.e., by calling System.loadLibrary). Each entry must contain a platform-dependent shared library with the correct naming convention, e.g., *.dll on Windows, or lib*.so on Solaris.

Here is Marc's Java 3D example from Marc's Web Start Kamasutra [`http://forum.java.sun.com/thread.jsp?forum=38&thread=166873`] post.

```
public static void main(String[] args) throws Exception
{
  // Note, that Web Start needs explicit native lib loading
  String os = System.getProperty( "os.name" );
  System.out.println("loading " + os + " native libraries ..");
  if( os.startsWith( "Windows" ) )
  {
    // Note, that order matters here
    // load those libs that are required by other libs first

    System.out.print( "j3daudio.dll .. " );
    // drop ".dll" suffix here
    System.loadLibrary( "j3daudio" );
    System.out.println( "OK" );

    System.out.print( "J3D.dll .. " );
    System.loadLibrary( "J3D" );
    System.out.println( "OK" );
  }
  else if( os.equals( "Linux" ) )
  {
    System.out.print( "libj3daudio.so .. " );
    // drop "lib" prefix and ".so" suffix
    System.loadLibrary( "j3daudio" );
    System.out.println( "OK" );

    System.out.print( "libJ3D.so .. " );
    System.loadLibrary( "J3D" );
    System.out.println( "OK" );
  }
  else
   throw new Exception( "OS '" + os + "' not yet supported." );

  [...]
}
```

Note, two things:

1. Prefixes and suffixes of the native lib's name have to be removed for the `System.loadLibrary` call.

2. Order matters. First pull in the lib with the least dependencies.

Circular Dependency - Known Bug

If you use two libs that depend on each other, your app won't take off no matter what operating system you try (e.g. Linux, Windows, etc.) and you will end up with an unsatisfied link error.

Example: If the lib `romeo.dll` uses a function in lib `julia.dll` and `julia.dll` uses a function in `romeo.dll` than they depend on each other. No matter which library you load first, e.g.:

```
System.loadLibrary( "romeo" );   System.loadLibrary( "julia" );
System.loadLibrary( "julia" );   System.loadLibrary( "romeo" );
```

it will fail. If you start off with `System.loadLibrary( "romeo" )`, Web Start's class loader will find the library `romeo.dll` and try to resolve all references. However, Web Start's class loader won't find the library `julia.dll` as `System.loadLibray( "julia" )` hasn't been called yet and `julia.dll` resides in the Web Start cache under a mangled name and in a directory that is not included in the system path (that is, `LD_LIBRARY` for Linux or `PATH` for Windows). You can only break the cycle if you pass in both libraries at once (e.g. `System.loadLibrary( new String[] { "romeo", "julia" } )`). Unfortunately, this method doesn't exist yet and you're stuck with the circular dependency bug filed at: `http://developer.java.sun.com/developer/bugParade/bugs/4491398.html`

Marc van Woerkom suggests a brute-force workaround.

Step 1: Try out the workaround by hand

- Find the Java runtime that runs your Web Start app
- Copy the dlls into the Java runtime's binary directory (e.g. `jre/v1.4/bin`) and the native-

code-calling jars into the Java runtime's extension directory (e.g. `jre/v1.4/lib/ext`)

- Try out your app using Web Start or stand-alone; check if the "fixed-up" Java runtime now loads your native dlls without a hitch.

Step 2: Automate: Create an extension installer that

- identifies the os/plattform
- locates the Java runtime
- beams over your native dlls zipped up in a jar as well as the native-code-calling jars
- unzips your dlls from the jar into the `jre/bin` directory, and shuffles the native-code-calling jars to the `jre/lib/ext` directory

Q: Can I use my own URLs for downloading JREs?

Yes, Todd Dunst has tried it with IBM's Java runtime 1.3.0 and it works.

Dale Searle's posted the complete sample code for a custom Java Runtime Installer plus a servlet that handles the JNLP HTTP extension protocol. You can find Dale Searle's code reprinted in the appendix plus distilled instructions and comments.

According to the jnlp spec, section 6.4, Web Start uses the extension protocol to request JREs from the server. The jnlp spec tells you what HTTP request will result from a j2se version specification. Like

```
<j2se version="1.3"
      href="http://www.jrevendor.com/servlet/jreinstaller"/>
```

leads to a

```
GET http://www.jrevendor.com/servlet/jreinstaller?
             arch=x86&os=Windows+95&locale=en_US&
             version-id =1.3&known-platforms=1.2
```

request to the web server.

Here is Todd Dunst's distilled post:

On the web server you need a servlet (e.g. `JREInstaller`), PHP CGI script or whatever running that decodes the extra URL parameters (arch, locale, version-id, known-platforms) from Web Start's request and returns a JNLP file that describes how to download and start a Java extension installer (e.g. `<installer-desc>`) that installs the requested Java runtime.

Note, that you can't just return IBM's installer. Instead you have to write your own Java installer as Web

Start's Extension Installer service can't handle native installers (`.exe`) but only works with Java installers.

In order to package the IBM JRE, I installed it to my local machine using IBM's Windows installer and then packed the complete JRE file structure in a single jar (this jar was approx. 17 MB).

Web Start downloads this jar along with another jar containing my custom JNLP installer app to the user's desktop. Web Start starts the JNLP installer app and the JNLP installer app unzips the jar containing IBM's JRE to a configurable directory and uses the JNLP Exension Installer service to to popup a progress dialog to keep the user informed. Finally, the JNLP installer app configures Web Start so that it can to use the newly installed Java runtime.

Once the installer has installed and configured the Java Runtime, control returns to Web Start which starts the app using your newly installed Java runtime. It's all automatic and completely transparenet to the user.

Do I need to sign any of the Jars?

The JRE installer app is no different than any other Web Start app. Therefore, you need to sign all jars to get permission to expand all JRE files on the user's hard drive.

Note, that you need sign the jar holding the compressed JRE as well. If you want to avoid signing all entries in the compressed JRE, you can add the compressed JRE to yet another Jar and sign it. This signs only the compressed JRE jar instead of all its entries.

How do you unpack the JRE Jar file?

Use Java's standard zip/jar classes (`java.io.zip.*`).

Where do I install the JRE to? How do I tell Web Start to use this newly installed JRE?

Use `setJREInfo(platformVersion, jrePath)` method of the `ExtensionInstallerService`.

The first argument (platformVersion) is the actual version of the JRE that you decided to install based on the extra URL parameters sent by Web Start to your JREInstaller servlet. As the requested JRE version may contain wildcards (e.g. `<j2se version="1.3+"/>`), only you know which JRE version your servlet decided to send back.

The second argument (jrePath) is the path to the JRE executable of your newly installed JRE. This argument consist of the base installation path, plus whatever additional path is necessary to get to the JRE executable. (For Windows, the path may be something like `c:\program files\ibm\ibm-jre-1_3_0\bin\javaw.exe`. You would build it as follows:

- Base path (from installer properties file) e.g. `C:\Program Files\ibm\ibm-jre-1_3_0`
- Plus Java executable path e.g. `\bin\javaw.exe`

Q: Where can I find JNLP's DTD?

Sun has published JNLP's DTD in the appendix of JNLP's spec (JSR-58).

If you don't want to type it in, you can download a typed-in version at `http://www.vamphq.com/download/jnlp-dtd-schema.txt`

I also created an XML Schema for JNLP available for download at `http://www.vamphq.com/download/jnlp-xml-schema.txt`

To validate your JNLP file, you can use Vamp's validator, code-named Vanessa, that supports DTD, XML Schema, Relax and more. See `http://www.vamphq.com/vanessa.html` for details.

Q: Can I use Windows UNC names (aka shared network drives) in file:// URLs?

Yes, you can. Dale King figured it out.

If your app's files were in `\\foobar\share\directory` you would set the codebase URL to `codebase="file:////foobar/share/directory"`.

You can use backslahes if you want, but forward slashes will work and allow you to copy it to Java code without requiring you to escape them. Note that you need four slashes between `file:` and the machine name.

Q: How can I add arguments to an installer?

An installer doesn't take in arguments, but you can pass on properties achieving the same effect; add properties to your installer's resources section.

Q: Why doesn't 1.3.1_03 work as a Java runtime platform version?

Platform vs. Product versions. Web Start differentiates between platform and product versions for Java runtimes. If you choose a vendor-specific product version such as `1.3.1_03` or `1.3.0_03` instead of a vendor-neutral platform version like `1.3`, you always need to add a vendor-specific href (e.g. `http://java.sun.com/products/autodl/j2se` for Sun) or else Web Start assumes `1.3.1_03` is a vendor-neutral platform version and won't find any matching Java runtimes.

- version without href --> vendor-neutral *platform* version
- version plus href --> vendor-specific *product* version

Example:

```
<j2se version="1.3.0_03" /> <!-- wrong; non-existant platform version -->
<j2se version="1.3.0_03" href="http://java.sun.com/products/autodl/j2se" />
<!-- bingo -->
```

Q: What properties can I pass on to the Java runtime?

Aside from the documented Java runtime properties, Web Start knows some undocumented "magic" properties that it will pass on to the Java runtime *before startup* instead of adding them to the system property table after the Java runtime is up and running. These undocumented "magic" properties include:

- `sun.java2d.noddraw`
- `javax.swing.defaultlf`
- `javaws.cfg.jauthenticator`

Example:

```
<property name="sun.java2d.noddraw" value="true"/>
```

Note, hackers cannot abuse these undocumented "magic" properties to break out of the sandbox. Web Start only lets through properties that can cause no harm.

Libraries (JSSE, JCE, JAAS, etc.)

Q: How can I use Web Start and JCE together?

JCE is part of the JDK 1.4 core and no longer a separate package and should work out-of-the-box.

Prior to JDK 1.4 you have to remove some roadblocks before JCE flies.

- Since JCE is signed with a different key, you need to sign Sun's JCE jars with the same key you're signing your Web Start app.
- JCE's `local_policy.jar` and `US_export_policy.jar` have to be in the same directory as `jce1_2_1.jar`. Both jars also have be named exactly as listed. When Web Start adds jars to its cache it renames the jars by prepending an RM prefix (e.g. `RMlocal_policy.jar`). This breaks JCE. To make it work, you need to manually copy the jars in Web Start's cache using the original/ unmangled names. (Everything continues to work even if you remove your app from the cache since Web Start does not delete files it did not create.) This patch, however, breaks Web Start's single click installation philosophy. You can overcome it by creating a jnlp installer.

Don't despair. It can be done. Here are John Archer's and Adam Ramadan's success stories.

I have successfully used the Java Cryptography extension (JCE) in conjunction with Web Start.

I seem to remember there are two options for using extensions with JWS - either you install your extension permanently into the ext dir of the relevant jre, or you handle it instead by using an extension jnlp file. I did the latter and it works fine - you don't need to bother about directories or classpaths - Web Start takes care of everything for you.

Don't forget that you need to use a different extension jnlp for each set of jars that are signed with different certificates.

- John Archer

To use JCE you have to enable <all-permissions/> in the jnlp file and sign your application. However, since the JCE is signed with a different key, you need to sign Sun's jars as well, with the same key you're using with your Web Start app. As soon as I did that, no problems.

- Adam Ramadan

Here are John Archer's jnlp files and comments.

I'm now using the Web Start installer feature to copy `local_policy.jar` and `US_export_policy.jar` to the cache directory if they are not already there, prior to the main app starting. My main jnlp file looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
  <jnlp spec="1.0+"
        codebase="http://www.metacompute.com/apps"
        href="spssuite.jnlp">
     <information>
        <title>SPS Suite</title>
        <vendor>MetaCompute.com</vendor>
        <homepage href="http://www.metacompute.com/spssuite.htm"/>
        <description>SPS Suite Version 1.0</description>
        <description kind="short">SPS checking, viewing, and editing
tools</description>
        <icon href="images/head2_water.jpg"/>
        <offline-allowed/>
     </information>
     <security>
        <all-permissions/>
     </security>
     <resources>
        <j2se version="1.3+" maximum-heap-size="160m"/>
        <jar href="spssuite_1_0_15.jar"/>
        <extension name="Java Crypto" href="crypto.jnlp"/>
        <extension name="Install_SPS" href="install.jnlp"/>
     </resources>
     <application-desc/>
  </jnlp>
```

The JCE stuff must go in a separate jnlp as it is signed by Sun. This file looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
  <jnlp spec="1.0+"
        codebase="http://www.metacompute.com/apps"
        href="crypto.jnlp">
    <information>
       <title>Java Crypto</title>
       <vendor>Sun MicroSystems, Inc.</vendor>
```

```
      <offline-allowed/>
    </information>
    <security>
      <all-permissions/>
    </security>
    <resources>
      <jar href="jce1_2_1.jar"/>
      <jar href="sunjce_provider.jar"/>
    </resources>
    <component-desc/>
  </jnlp>
```

So far so good. As mentioned above, the problem is that with JCE 1.2.1 there are two policy jar files (`local_policy.jar` and `US_export_policy.jar`) which must reside in the same directory as the JCE jars. If you include them as resources in the jnlp file, Web Start puts them in the right directory, but renames them as it does with all the cache files (e.g. `RMUS_export_policy.jar`), so they aren't found and you get a bunch of exceptions.

This is where the installer jnlp comes in. This basically copies these policy jars to the relevant cache directory before the application is run, so that they are there when the JCE looks for them. The installer jnlp looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
  <jnlp spec="1.0+"
        codebase="http://www.metacompute.com/apps"
        href="install.jnlp">
    <information>
      <title>SPS Installer</title>
      <vendor>MetaCompute.com</vendor>
      <icon href="images/head2_water.jpg"/>
      <offline-allowed/>
    </information>
    <security>
      <all-permissions/>
    </security>
    <resources>
      <j2se version="1.3" />
      <jar href="install.jar"/>
    </resources>
    <installer-desc main-class="SpsInstaller/Install"/>
  </jnlp>
```

Agnes Juhasz has posted the source for an installer to add the JCE policy files to your Java runtime. Fast-forward to the appendix for Agnes Juhasz's installer source code.

Note, that there are many more JCE libraries apart from Sun's JCE. Try an alternative open-source JCE library such as BouncyCastle, Cryptix, BeeCrypt or Forge and hope for less hassle.

Name	Provider	URL	Comments
SunJCE	Sun	`http://java.sun.com/products/jce/index-12.html`	Sun's reference implementation
Cryptix	The Cryptix Foundation	`http://www.cryptix.org/`	UC Berkley License
Bouncy Castle	Legion of Bouncycastle	`http://www.bouncycastle.org/`	MIT License

Name	Provider	URL	Comments
BeeJCE, BeeCrypt	Virtual Unlimited	`http://www.virtualunlimited.com/products/beecrypt/`	GNU LGPL
CDCStandard, CDCEC	TU Darmstadt	`http://www.informatik.tu-darmstadt.de/TI/Forschung/cdcProvi`	GNU GPL, LGPL
Forge	The Forge Group	`http://www.forge.com.au/products/crypto/index.html`	Forge Public License
IAIK	TU Graz	`http://jcewww.iaik.at/`	closed-source payware, free for non-commercial use
JCSI	Distributed Systems Technology Centre (DSTC)	`http://security.dstc.edu.au/projects/java/jcsi.html`	closed-source payware, free for non-commercial use
Crypto-J	RSA Security	`http://www.rsasecurity.com/products/bsafe/cryptoj.html`	closed-source payware
JProv	Eracom	`http://www.eracom.com.au/products/ejprov.html`	closed-source payware
KeyTool Crypto	Baltimore Technologies	`http://www.baltimore.com/keytools/crypto/`	closed-source payware
Phase Base Crypto	Phaos Technology	`http://www.phaos.com/e_security/prod_crypt.html`	closed-source payware

For an up-to-date JCE provider list check

- `http://www.nue.et-inf.unisiegen.de/SignStreams/csp/overview_provider.html`
- `http://java.sun.com/products/jce/jce12_providers.html`

Note, that starting with JDK 1.4 JCE libraries must be signed or else you end up with the exception below:

```
java.security.NoSuchProviderException: JCE cannot authenticate
     the provider <your provider here>
```

Currently only Sun's, IBM's and Bouncy Castle's JCE libraries are signed. Cryptix has already received Sun's signing key and should be available in a signed version shortly.

Q: How can I use Web Start and Comm API together?

If you use Sun's `comm.jar`, you are out of luck as Paul Lucassen's posting attests below. However, you can use serialio.com's Serial Port [`http://www.serialio.com`] package as an out-of-the-box replacement for Sun's library as it works with Web Start as advertised. As an alternative you can also use IBM's Comm API [`http://www-106.ibm.com/developerworks/java/jdk/linux130/othpkgs.html`] which works wonderfully according to Jay Colson.

I have tried to get the Sun's Communications API to work with Web Start, but have failed in all attempts. I will describe what I have tried so far.

Sun's Comm API (2.0) requires three files to be available on the client machine: a jar (`comm.jar`), a dll (`win32com.dll` - I have so far only tried this under NT4.0) and a properties file (`javax.comm.properties` - containing the line `Driver=com.sun.comm.Win32river`) For a "standard" install, not using Web Start the jar goes into `jre\lib`, the dl into `jre\bin` and the prop file into `jre\lib` as well. This works fine. I can then access the chip card reader on COM2 without problems.

Now comes Web Start. I deploy `comm.jar` in the usual way. I pack `win32com.dll` into `win32com.jar` (in the `root' of this file as the spec says about loading native libs. Then at first I decided that the prop file should in principle also not be pre-installed on the client machine, so at first I naively packed it in the `win32com.jar` (either in the root or as `lib/javax.comm.properties`). When I do this I get a "NullPointerException: name can't be null" the first time the code tries to `CommPortIdentifier.getPortIdentifier("COM2")`.

I studied this carefully and after some debugging decided that probably the properties file was the problem. I came to this conclusion since my debugging code was able to load the classes from `comm.jar` without problems (by using the classloader obtained by `this.getClass().getClassLoader`), and to load the native library `win32com.dll` as well (by calling `System.loadLibrary("win32com")`).

The next step was to put the properties file on the client, in `jre\lib`. This dramatically changes the message I got: "NullPointerException: name can't be null while loading com.sun.comm.Win32Driver". Thus the properties file is found, but loading the driver failes for some 'name' is still null. Needless to say I attempted the same with a non-existent drivername, I then got "ClassNotFoundException: while loading <name of driverclass>"

Another attempt was made where the dll was in `jre\bin` and not deployed thru Web Start (only `comm.jar` was in the `comm.jnlp` extension file), giving the exact same result as above where the dll is deployed in a jar by Web Start.

So: I have the properties file where the API reads it, I have `comm.jar` where the API can find the `com.sun.comm.Win32Driver.class` (see the `ClassNotFoundException` if I change the name of the driver) and I have the dll in a place where a simple call to `System.loadLibary()` can load it (changing the name yields an Exception, I tried) -- but still no success.

This describes the attempts. Some more context: The app is deployed through a couple of signed JAR's (certificate signed by an official CA) that are listed in one jnlp. I have tried deploying the Comm API in the same .jnlp file and as an extension .jnlp file: no difference. (Probably no difference as all jar's are signed by the same ceritificate, `comm.jar` from Sun is not signed so I sign it with our own cert.)

- Paul

Here is Scott Hughes experience with RXTX an open-source library available for Linux as well as Win32 at `http://www.rxtx.org`.

I'm trying to get RXTX running on Win32. In the static initializer for `CommPortIdentifier` the code snippet below fails:

```
try
{
  CommDriver commdriver = (CommDriver)Class.forName(s1).newInstance();
  commdriver.initialize();
}
catch(Throwable throwable)
{
  System.err.println( "Caught " + throwable + " while loading driver " + s1);
}
```

The error I get (which is virtually identical on both windows and linux) is:

```
Caught java.lang.NullPointerException:
  name can't be null while loading driver com.sun.comm.Win32Driver
```

I tried writing my own serial driver that is just a wrapper for the Win32Driver and stored it in my signed jar. Then I get a `ClassNotFoundException` in `CommPortIdentifier`'s static initializer.

I tried using reflection to instantiate and initialize both my wrapper driver and the win32 driver in the main method of my app, and it worked fine. However, when I go out to the `CommPortIdentifier`'s static initializer, it fails.

My guess was that it could be that `comm.jar` wasn't signed, so it was a case of trusted code calling untrusted code which wanted to perform a protected operation. So, I signed `comm.jar` and included it in the download of my app and I get the same error.

I discovered a workaround. In the main app (in a class that resides in a signed, downloaded jar) try setting the `SecurityManager` to `null` before accessing the serial ports for the first time. At the top of your static void main method, just add:

```
System.setSecurityManager(null);
```

This fixed the problem and my app works with RXTX on Linux and the Java Communications API on Windows.

Dale King found a workaround to use Sun's Java Comm API with Web Start. Here is Dale King's distilled posting:

The workaround uses the Comm API without the properties file by setting up the driver yourself. The contents of the properties file is usually a one-liner like:

```
driver=com.sun.comm.Win32Driver
```

that lists the class name of the operating system specific driver. Sun's Comm API sets up the driver automatically when the `CommPortIdentifer` class is loaded. But you can do it yourself. Example:

```
String driverName = "com.sun.comm.Win32Driver";
CommDriver commDriver = (CommDriver)Class.forName( driverName ).newInstance();
commDriver.initialize();
```

Instead of hard-wiring the driver's class name you can add it as a property in your JNLP startup file.

Note that even after you set up the driver yourself, Sun's Comm API keeps checking if the property file exists by asking the security manager if it is deletable. On most machines without Sun's Comm API installed, the file name is null and the security manager throws an exception. To make it work turn off the security manager:

```
System.setSecurityManager( null );
```

`moa` has posted a patched version of Sun's `comm.jar` that works fine with Web Start. The patched version ignores the properties file and instead loads the driver using the system property `javax.comm.properties.Driver` that you can set in your JNLP startup file. The patch is available for download at `http://www.host.dunlops.com/java/commws.jar`

Q: How can I use Web Start and JAAS together?

JAAS is part of the JDK 1.4 core and no longer a separate package and should work out-of-the-box as Sun has fixed the class loading issue present in older JAAS versions prior to JDK 1.4.

Prior to JDK 1.4 JAAS has to be added to your app and you are up for a struggle to make it work. Using JAAS directly, to login to an ejb app server or whatever, is out of the question with Web Start on JDK 1.3 according to Sun:

> The problem you got is due to JAAS 1.0 (`jaas.jar`) uses `systemClassLoader` to load classes that are defined in the JNLP app jar files, which should be loaded by the `contextClassLoader` instead. For more information on class loading with Web Start, look at:
>
> - `http://java.sun.com/products/javawebstart/faq.html#54`
> - `http://java.sun.com/products/javawebstart/docs/developersguide.html#dev`
>
> The new JAAS that comes with JDK 1.4 fixed this problem, which uses `contextClassLoader` in their class.

A workaround is to use a SOAP servlet as a proxy for the app server and to define a SOAP xmlrpc api between your client and app server. Using SOAP has the advantage of getting through firewalls as SOAP is transported over HTTP. Use Web Start to install and start your SOAP Java client app.

Another workaround that only works for Intranets is to patch Web Start. Put the `jaas.jar` in the `lib/ext` directory of the JRE used by Web Start and add your own policy file. Obviously, this workaround is an act of desperation and not recommend for long term use.

Andy Armstrong maintains some JAAS plug-in modules (including Windows NT/2000 authentication) released under the GNU LGPL online at `http://free.tagish.net/jaas/`.

Q: How can I use Web Start and JSSE together?

JSSE is part of the JDK 1.4 core and no longer a separate package and should work out-of-the-box.

Prior to JDK 1.4 you have to add JSSE to your app and tweak it to make it work. It should work as Gavin Everson's posting attests:

> I work for a company that has a working app via Web Start that downloads `jsse.jar`, `jh.jar`, `jcert.jar`, `jnet.jar` all from the one jnlp file, you don't need to try to copy them to any spot on the client, Web Start handles them as needed, and the calls to `import` from within your code will all function as they should.

The following is a distilled version of a thread from Sun's Web Start forum. Don't blame me, if it doesn't work as I haven't tried it myself.

My jnlp file looks like this:

```
<jnlp [...]>
   [...]
   <security>
     <all-permissions/>
   </security>
   <resources>
     <jar href="myapp.jar"/>
     <jar href="jsse.jar"/>
     <jar href="jnet.jar"/>
     <jar href="jcert.jar"/>
   </resources>
   <application-desc [...]>
   [...]
</jnlp>
```

(All the jar's are signed with a self-issued certificate: when asked by Web Start, the user must trust it for the app to run). The app main method looks like:

```
public static void main(String[] args)
{
  java.security.Security.addProvider(
    new com.sun.net.ssl.internal.ssl.Provider() );

  System.setProperty("java.protocol.handler.pkgs",
                     "com.sun.net.ssl.internal.www.protocol");

  System.setProperty("javax.net.ssl.trustStore",
                     "mytruststore");

  try
  {
    URL myUrl = new URL("https://myserver.com");
  }
  catch (MalformedURLException mue)
  {
    System.err.println(mue);
  }
}
```

I've read somewhere that the `setProperty`-approach used above doesn't work for protocol handlers (instead one should use `URLStreamHandlerFactory`) Moreover, I suspect that the SSL engine would not find the truststore file I specify (which is bundled in the jar), since it does not seem to look for it in the classpath. In this particular case, however, the certificate used in the apps for https communications is the same as the one trusted by the user at the launching of the app. So:

- How can one deploy JSSE and have it handle "https" correctly?
- How can one have JSSE trust a self-issued certificate already trusted by the user in Web Start?

Try setting up the SSL library like this:

```
  com.sun.net.ssl.internal.ssl.Provider p =
    new com.sun.net.ssl.internal.ssl.Provider();
  Security.addProvider(p);
  URL.setURLStreamHandlerFactory(new HttpsHandlerFactory());
```

That should solve the problem.

Could someone tell me where the `HttpsHandlerFactory` class is to be found. I am unable to see it in the JSSE library.

I have the following in place:

```
 URL.setURLStreamHandlerFactory( new URLStreamHandlerFactory() {
   public URLStreamHandler createURLStreamHandler(final String protocol)
   {
     if(protocol != null && protocol.compareTo("https") == 0)
     {
       return new com.sun.net.ssl.internal.www.protocol.https.Handler();
     }
     return null;
   }
 });
```

Q: How can I load my own keystore from the app's jar? The following code snippet from my SSL client app works when invoked from the command line with `mykeystorefile` in the same directory:

```
...
ks = KeyStore.getInstance( "JKS" );
ClassLoader cl = this.getClass().getClassLoader();
InputStream keyStore = cl.getResourceAsStream( "mykeystorefile" );
ks.load( (InputStream)keyStore, "password" );
```

But fails with "SSL implementation not available" when I remove the keystore from the directory and try to load it from my app's jar.

Try using a slash in front of your keystore filename, then the Java runtime should load the keystore from the classpath (which contains your jar). Example:

```
InputStream keyStore = cl.getResourceAsStream("/mykeystorefile");
```

If you omit the slash, the Java runtime tries to load the keystore from the directory the calling class is located in.

Note, that `-Djavax.net.ssl.truststore=xxx` works only for keystores specifid by a path (that is, stored in stand-alone files). It fails for keystores packed up in jars.

Here is Dean Cording's code snippet that get SSL and HTTPS rolling with Web Start:

```
java.security.Security.addProvider( new com.sun.net.ssl.internal.ssl.Provider() );

java.net.URL.setURLStreamHandlerFactory( new java.net.URLStreamHandlerFactory() {
   public java.net.URLStreamHandler createURLStreamHandler(final String protocol)
   {
      if ("https".equals(protocol))
      {
        return new com.sun.net.ssl.internal.www.protocol.https.Handler();
      }
      return null;
   }
});

// This is a kludge to get JSSE to use Web Start's cacerts keystore

if( System.getProperty( "javawebstart.version" ) != null )
{
  System.setProperty( "javax.net.ssl.trustStore", System.getProperty( "jnlpx.home" )
    + System.getProperty("file.separator") + "cacerts");

  if( System.getProperty( "javax.net.ssl.trustStorePassword" ) == null )
  {
     System.setProperty( "javax.net.ssl.trustStorePassword", "changeit" );
  }
}
```

Note, that you must either have your key signed by a built-in, factory-shipped certificate authority (e.g. Thwate) or install your public key into Web Start's cacerts keystore (which is separate from the Java runtime's cacerts keystore).

Q: How can I use Web Start and Java 3D together?

As an example app that uses Java 3D together with Web Start check out Masa Takatsuka's JBeanStudio app [`http://www.geovistastudio.psu.edu/JBeanStudio/`] .

Check your JRE as Java 3D doesn't work with all JRE versions. When using JRE 1.3.0_02 (the JRE that is default for JBuilder 5) things won't work. You need at least JRE 1.3.0_03 on Windows. (JRE 1.3.0_03 comes with the full install of Web Start 1.0.1_01.)

For an excellent explanation of all the details check out Marc's Web Start Kamasutra [`http://forum.java.sun.com/thread.jsp?forum=38&thread=166873`] .

Q: How can I connect to a database?

Note, that you cannot use Sun's JDBC ODBC Bridge driver because ODBC is only available for Windows and requires that you set up a data source name (DNS) on every machine.

Instead use a zero-admin 100 % Java JDBC driver (aka type 4 JDBC driver) that spares you from setting up an ODBC DNS on every machine. Try `http://sourceforge.net/projects/jtds/` or `http://www.thinweb.com/tw_products_twfreetds.html` for MS SQL Server or try Sybase JConnect which is free as well.

Here is Ivan Ooi's post using Oracle's thin driver:

```
DriverManager.registerDriver( new oracle.jdbc.OracleDriver());

gu_app.dbCon = DriverManager.getConnection(
  "jdbc:oracle:thin:@10.0.0.1:1521:myDB", "DBA", "SQL" );
```

This is what I use in my code and I am able to connect to my office Oracle DB. My app's jars are not on my Oracle DB server's machine. DB server, web server and Web Start client reside all on different machines and it works.

- Ivan Ooi

PS: Note, that you need to sign Oracle's `classes12.zip` file as well because all jars in your jnlp must have the same certificate.

Note, that most databases contain sensible data and therefore do not allow arbitrary connections from nobodys.

In this case you can use servlets to extract whatever data you need from the database and send it (zipped) to your Web Start app. As your user never connects to your database directly, you can better protect your database from intruders.

Q: How can I use Web Start and RMI together?

Q: I am currently developing a Java game using RMI to make it networkable (squeezing both the server and client into the same app). Here is my puzzle:

In order to run an RMI server app, you must start the RMI naming service from the command line, for example:

```
unset CLASSPATH
start rmiregistry

java -
Djava.rmi.server.codebase=file:/c:\home\ann\public_html\classes/
-Djava.rmi.server.hostname=zaphod.east.sun.com
-Djava.security.policy=java.policy
```

```
engine.ComputeEngine
```

Is there any way around this (either in Web Start or via some other magic?)

A: Note, that you don't need to start the rmi registy because it is optional. If you need the registry you may start it and export your objects at runtime. Example:

```
LocateRegistry.createRegistry( PORT );
System.out.println( "Registry created" );
UnicastRemoteObject.exportObject(object);
Naming.rebind("rmi://" + HOST_NAME + ":" + PORT
  + "/MyService", (MyServer) object);
```

Check out some RMI tutorials or how todos for further explanations.

Q: How do you hand over the policy file to your app?

```
java -Djava.security.policy=<path-to-policy-file> Client
```

How do you translate the pre-Web Start command line into jnlp?

A: You don't need a policy file. You can request all permissions for your Web Start app in the jnlp startup file using `<all-permissions>`.

I use a signed jar and when the user starts my app, Web Start pops up a security dialog begging the user to grant my app all permissions. So you don't need a policy file.

Q: Does it mean that I ignore the security manager? Do I need this line in my Web Start RMI client app?

```
System.setSecurityManager( new RMISecurityManager() );
```

[Editor's Note: I do not understand what's going on here. I will clear this entry up once I get a chance to try it myself.]

Security, Signing, Sandbox

Q: Can I use a secure socket (SSL) connection back to the host when my app runs in the sandbox?

No, you need to request `all-permissions` for you app.

Q: Does Web Start support SSL?

Yes Sir. Starting with the 1.2 series Web Start supports HTTPS in JNLP startup files (`codebase`, `href`, and so forth). Note, however, that Web Start must run on Java 1.4 or greater. Java runtimes prior to 1.4 lack the built-in HTTPS machinery that Web Start 1.2 or greater relies on.

Web Start prior to 1.2 does *not* support HTTPS for downloading your app's jars from the web server. However, Web Start supports codesigning so you can be sure the jars are downloaded uncompromised.

Web Start also supports JSSE (Java Secure Socket Extension) for apps. Thus, apps can use JSSE to connect back to the web server using HTTPS.

Q: Why doesn't Web Start launch my signed application?

Signed applications won't work if Web Start itself (not your app) runs under 1.2.2. Web Start needs to run at least under 1.3. You can check what JRE Web Start picked up by opening your `javaws.cfg` config file in the Web Start installation directory (e.g. `c:/java/jws/v101`). Don't look in your home directory as this is a different config file although it has the same name. There is no GUI panel in Web Start to edit this file so you have to change it manually if Web Start happend to pick up a 1.2 JRE.

Rationale: You may be wondering why, right. My certificate is a RSA certificate from Thawte. The RSA algorithm is only supported under JRE 1.3. JRE 1.2 has no way to recognize the RSA encoded certificate.

Q: How can I turn off the sandbox?

If your app is signed, you can call `System.setSecurityManager(null)` to turn off the sandbox. If you get rid off the security manager, your app should speed up as it no longer goes through security layers. Don't expect miracles, though.

Q: How can I sign jars using Ant?

Use the `signjar` task:

```
<signjar jar="<yourjar.jar>"
         storepass="<store_password>"
         alias="<youralias>"
         keystore="<keystore>"
         keypass="<key_password>" />
```

This assumes your key was already added to the keystore using the `keytool` tool.

If you package your jars in a Web Archive (`.war`), you can use Vamp's Ant Task Suite to sign your jars. You can find Vamp's Ant Task Suite at `http://www.vamphq.com/ant.html`

Q: Can I make Web Start use my own policy file?

Web Start doesn't allow you to use your own policy file.

A workaround that only works for Intranets is to patch Web Start. Change Web Start's default policy file to suit your needs or ditch it and create your very own.

Q: How can I use jars signed by someone else?

You can either sign them with your own certificate or if you want to leave them untouched you can put them in a separate jnlp file (aka extension) and use the vendor's signature.

Q: How can I sign jars?

You can read Sun Web Start Developer's Guide: Signing Jar Files with a Test Certificate [`http://java.sun.com/products/javawebstart/docs/developersguide.html#dev`] to get started.

You can find more details in Appendix C: How to Sign Java Code: Section 6 - Signing Code with Sun's Java 2 in the free Securing Java book online at `http://www.securingjava.com/appdx-c/appdx-c-6.html`

Q: Which trusted root certificates ship with Web Start?

Web Start stores its trusted root certificates in the `cacerts` keystore. Use `keytool -list -keystore cacerts` to list all root certificates. Here's the condensed output from my machine:

```
Keystore type: jks
Keystore provider: SUN

Your keystore contains 11 entries:

thawtepersonalfreemailca,
thawtepersonalbasicca
verisignclass3ca
thawtepersonalpremiumca
thawteserverca
verisignclass4ca
cybertrust
verisignserverca
verisignclass1ca
thawtepremiumserverca
verisignclass2ca
```

Use `keytool -list -v -keystore cacerts` for a verbose listing.

Q: How can I protect my code in jars?

You can use an obfuscator such as the open-source RetroGuard tool that strips all debug info leftovers such as (line number table, local variable table, source file, etc.) from your jars and renames your non-public (aka private) classes, methods and variables to meaningless names (mostly single letter names like `a`, `b`,`c`,`aa`,`ab`, etc.) As a side-effect your obfuscated jars usually shrinks by about a third in size (that is, 30%).

RetroLogic's RetroGuard resides online at `http://retrologic.com/retroguard-main.html`

Note, that renaming is the best protection available as it's like a one-way hash function that cannot be reversed.

In contrast, if you encrypt your jars and use your own custom classloader to decrypt your binary classes on the fly for the Java runtime's eyes only, your code is only sealed. Although you can use "unbreakable" cyphers you always need to pass along your classes unencrypted (that is, in plain binary format) to the Java runtime and, therefore, curious minds looking for your magic formula can always attach a debugger or use a fake Java runtime to grab the class bytes after they are decrypted without ever bothering to attack your "unbreakable" cyphers head-on.

For details about what's in the binary class format check out:

- The Java Virtual Machine Specification by Lindholm, Tim, and Frank Yellin, Second Edition published by Addison-Wesley in 1999 online at `http://java.sun.com/docs/books/vmspec/`

Q: Where can I find certificate authorities (CAs)?

Check out `http://www.pki-page.org/` for an extensive listing.

Q: How can I install my own authenticator?

Starting with version 1.0.1_02 or greater you can stop Web Start from installing its own authenticator. Add the undocumented `javaws.jauthenticator` "magic" property to your JNLP startup file. Example:

```
<property name="javaws.jauthenticator" value="none" />
```

Q: Where can I get a free, zero-dollar Web Start key certificate?

You can sign up for a free, zero-Euro key certificate underwritten by Thwate Free Mail at `http://www.thawte.com/getinfo/products/personal/join.html`. Web Start includes the Thwate Free Mail certificate in its built-in certificate key ring.

Richard Dallaway wrote up all the steps (including all keytool commands):

- generate a RSA key
- export the key to a certification request text file ready for cut-and-paste hand-over to Thwate

- import the underwritten real thing in your key ring
- and so on.

Check out Richard Dallaway's "Java Web Start and Code Signing" hands-on, how-to paper (also available as a 4-page pdf booklet) at `http://www.dallaway.com/acad/webstart/` for details.

Q: Do signed jars with a certificate that expired still work?

Yes.

Here is a quote from the VeriSign Knowledge Base about Netscape Object Signing:

Q: After a signing certificate has expired, does the object continue to be trusted?

A: Signed objects are stored and used long-term, well after the certificates used for signing have expired. Although certificates expire, valid signatures do not. Signature validation is based on the date of the signature rather than the time verification occurs. If a certificate chain was valid at signing, Communicator will continue to recognize the signature even after certificates in that chain expire.

Voices from the Web Start forum trenches:

I have used Web Start for a while with a self-signed certificate and Web Start launches my apps without regard to the expiry date.

For example, when I start my app Web Start pops up the certificate dialog showing that the certificate expired two weeks ago, but when I click the Start button the app starts without a murmur. And from then on the app starts without any certificate popups.

When you signed your jars at a certain date, the certificate must be valid at that time not in the future. If you sign your jars, after the certificate expired, then it's wrong. The certificate is a prove that your jars have been signed at a precise time and not that the signature is valid for a certain time.

Most developer certificates expire after a year for no other reason than to fill the issuer's pocket.

Q: Unable to sign jar. How come?

When I try to sign a jar using `jarsigner`, I end up with the error below:

```
jarsigner: unable to sign jar: java.util.zip.ZipException:
  invalid entry compressed size (expected 54386 but got 54949 bytes).
```

Unjar the jar causing the hickup into a temp directory, and delete the manifest files. Then rejar it and sign it.

Q: How can I remove a signature from a jar?

Use brute force: Unjar the jar into a temp directory, and delete the manifest files. Then rejar it.

Web Start Extensions

Q: How can I display a license agreement?

Web Start doesn't support the display of a license agreement out-of-the box. You have to do it yourself.

One way is to display the license in the browser before the download. If the user doesn't accept your license, she cannot download your app. If the user accepts your license, you set a cookie and the user can start downloading your app.

Q: Why can't Web Start do ...?

You can request or vote for new Web Start features at Sun's Bug Parade at `http://developer.java.sun.com/developer/bugParade/`.

Some features under consideration include:

- better user experience by improving downloading, installing, upgrading, and administration
- improved multi-user and multi-platform support
- improved Java runtime configuration
- HTTPS support
- fine grained security support
- easier installation of extensions into `lib/ext`
- combine some Web Start and Java Plug-In components (for example, merging the Java Plug-in control panel and the Web Start preference panel into a single control panel; combining the Java Plug-In cache with the Web Start cache and so on).
- rearchitecture of Web Start's cache format
    - Web Start currently mangles cache entries (e.g. `venus.jar` becomes `RMvenus.jar`). To make the cache more intuitive one idea is to stop mangling cache entries. It is not clear why, for example, `http/Dwww.jenomics.de/P80/DMvamp/DMlib` can't be simply `http/www.jenomics.de/80/vamp/lib`. Another idea would separate Web Start's admin information for cache entries into a separate tree or subdirectory a la CVS. This would allow intuitive offline installations by letting you copy your files directly into Web Start's cache.

Don't expected any changes in the JNLP spec for the Java 1.4 series. All JNLP spec changes (2.0 ?) will show up in the upcoming Java 1.5 series (aka Tiger).

Note, in theory Web Start is not bound to a specific Java distro as it can upgrade itself and as it can download new Java runtimes and juggle various versions at the same time. However, in practice Sun seems to ship new Web Start versions only in synch with new Java versions (v1.4.1, v1.5).

Hopper (aka Java 1.4.1 - scheduled for Fall 2002)

- full Itanium (Intel 64-bit) support

Mantis (aka Java 1.4.2 - scheduled for Spring 2003)

Tiger (aka Java 1.5 - scheduled for Winter 2003)

- JSR 121: Application Isolation API Spec (`http://www.jcp.org/jsr/detail/121.jsp`) - faster startup and reduced memory footprint thanks to application partioning (aka app domains, multiple processes share a single VM) and resource sharing (aka binary class/bytecode sharing between processes)

If you think an important feature is missing in Web Start, you might not be the first one. Check out the Web Start 2.0 petition [`http://www.vamphq.com/king.html`] to see if your feature is listed. If your feature isn't listed, you are invited to send an e-mail to comments@vamphq.com to add it.

Upcoming Web Start 1.2 Features in Java 1.4.1 (aka Hopper)

For Developers

- customizable splash screens (with or without icon, icon only)
- updated app manager
- more sample code
- expanded encoding support for JNLP startup files

For Admins

- Enterprise installation enhancements (multiple clients share single configuration, set or change config settings with AutoInstall, control any config setting)
- AutoInstall Plug-In for Internet Explorer for Windows (ease initial installation of Web Start; install Web Start and/or app; configure Web Start)
- HTTPS support

AutoInstall Preview: Uses HTML `object` tag, e.g.:

```
<OBJECT
  CODEBASE="http://java.sun.com/javaws-1_2.cab"
  CLASSID="clsid:5852F5ED-8BF4-11D4-A245-0080C6F74284"
  HEIGHT="96" WIDTH="192">
  <PARAM NAME="configuration"
         VALUE="http://deploy.com/global.cfg" />
  <PARAM NAME="application"
         VALUE="http://deploy.com/app.jnlp" />
  <!-- Alternate HTML for non-Internet Explorer browsers -->
  <A HREF="http://java.sun.com/jws-platform.sh?">
     Click here to download Web Start
  </A>
</OBJECT>
```

For Users

- improved multi-user support
- improved multi-platform support
- single instance service (quickly restart app with new data; Listener API; Example: stock charting app)
- bookmark support
- improved Java runtime configuration
- enhanced cache management

(Source: JavaOne 2002 Talk: Web Start: Advanced Topics by Andrew Herrick and Steve Bohne)

J2EE Troubleshooting

Q: Can I use Sun's J2EE Reference Implementation EJBs?

It took some work, but I finally got an EJB client to work with Web Start. There are two problems:

1. First, you must get your EJB client to work as a jar. This is tricky because there are two files (`c:\j2sdkee1.3\config\ejb.properties` and `c:\j2sdkee1.3\config\security.properties`) that the Sun J2EE classes must find below your local directory. In other words, if your EJB client jar is here `c:\myproject\blah.jar`, then you must have these files too (`c:\myproject\config\ejb.properties`, `c:\myproject\config\security.properties`).

2. The second problem is that you gotta sign all your jars, as explained in Sun's Web Start Developer's Guide [`http://java.sun.com/products/javawebstart/docs/developersguide.html`], and the client needs "all-permissions" as explained same place.

Q: How can I connect to Borland's AppServer?

The fact that you can't sign Borland jar files seems to be a deliberate design on the part of Borland; they effectively kill the jar and jarsigner tools by corrupting the checksum of one of the files. You have to use a command line tool like PKZIP 2.5 as follows:

```
rem Create a temporary directory.
md temp
cd temp
rem Unzip the jar file.
pkzip25 -nozip -dir -silent -extract ..\asrt.jar
rem Rejar the file, this time with the correct checksum.
jar cf ..\asrt.jar *
Remove the temporary directory.
cd ..
rd /s/q temp
```

You will need to do this for `vbjorb.jar` as well, after which you should be able to sign the jar files. Signing them is necessary because `asrt.jar` in particular violates all sorts of security restrictions during initialization.

If your EJB server is on your internal network, that's all you need to do; you should be able to bind to your objects without any further problems. If you're trying to run the EJB's across the Internet, there are probably half a dozen more steps, most of which involve configuring the Gatekeeper.

Miscellaneous

Q: How can I debug apps under Web Start?

Here is a simple solution to debug your app under Web Start:

```
C:\Program Files\JavaSoft\JRE\1.3.0_02\bin\javaw.exe
-Xdebug -Dnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=8118
"-Djnlpx.home=C:\Program Files\Java Web Start"
"-Djnlpx.heapsize=NULL,NULL"
"-Djnlpx.jvm=C:\Program Files\JavaSoft\JRE\1.3.0_02\bin\javaw.exe"
"-Djava.security.policy=file:C:\Program Files\Java Web Start/javaws.policy"
"-DtrustProxy=true"
-classpath "C:\Program Files\Java Web Start\javaws.jar;
        C:\Program Files\Java Web Start\javaws-l10n.jar"
com.sun.javaws.Main http://yourwebserver/your.jnlp
```

The basic idea in this frightening command line is to start Web Start yourself without the help of the native wrapper that hides all these juicy details. To make this work at your very own desktop you have to adjust the path settings accordingly and attach a debugger. The key is that

```
-Xrunjdwp:transport=dt_socket,server=y,address=8118
```

will make your app wait for a socket connection to port 8118 and you can now fire up a debugger that supports remote debugging and attach it to the VM via port 8118.

After much wailing and gnashing of teeth, I have developed a method for debugging Web Start apps. The code at the bottom of this message is a C++ Win32 console application that wraps around `javaw.exe` to pass in extra debugging parameters.

1. Compile this file and and copy the `jwsdebug.exe` output file to the JDK bin directory (e.g. `C:\jdk1.3.1\bin`).
2. Under "File | Preferences" in Web Start, select the "Java" tab and set the "Command" to point to the exe (e.g. `C:\jdk1.3.1\bin\jwsdebug.exe`).
3. Create a configuration file with additional parameters to be passed to `javaw.exe`. This file has to be in the same directory as the executable and have the same name with .config instead of .exe as the extension (e.g. `C:\jdk1.3.1\bin\jwsdebug.config`). A sample file is shown below.
4. Launch the JNLP file. Note: You cannot run more than one JNLP app and you cannot run the Web Start console at the same time. The Web Start console will also be run with the extra parameters which, in the example below, includes the debugger port number. You can't have more than one app listening on a single port.

If you want a pre-compiled binary and the sample configuration file, e-mail me directly (kdean@datadevelopment.com) and I'll send them along.

Example configuration file:

```
-classic
-Xdebug
-Xnoagent
-Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,server=y,address=5000,suspend=n
```

```
-- BEGIN SOURCE - CUT HERE --

  #include <dir.h>
  #include <string.h>
  #include <fstream.h>
  #include <windows.h>

  // Compiled with C++Builder 5.0.

  int main(int argc, const char * argv[])
  {
    // Get the home directory.
    char homeDirectory[MAXPATH];
    strcpy(homeDirectory, argv[0]);
    *strrchr(homeDirectory, '\\') = '\0';

    // Get the application path with the .exe extension.
    char applicationPath[MAXPATH];
    strcpy(applicationPath, argv[0]);
    *strrchr(applicationPath, '.') = '\0';

    // Get the configuration file path.
    char configFilePath[MAXPATH];
    strcpy(configFilePath, applicationPath);
    strcat(configFilePath, ".config");
    // Rebuild the command line starting with javaw.exe in the same directory.
    char commandLine[4096];
    strcpy(commandLine, homeDirectory);
    strcat(commandLine, "\\javaw.exe");

    // Append parameters from configuration file.
    ifstream configF(configFilePath);
    if (configF.good())
    {
      char param[512];
      while (configF.getline(param, sizeof(param)).good())
      {
        strcat(commandLine, " \"");
        strcat(commandLine, param);
        strcat(commandLine, "\"");
      }
    }

    // Append original parameters.
    for (int i = 1; i < argc; i++)
    {
      strcat(commandLine, " \"");
      strcat(commandLine, argv[i]);
      strcat(commandLine, "\"");
    }

    STARTUPINFO startupInfo;
    GetStartupInfo(&startupInfo);

    PROCESS_INFORMATION processInfo;
    CreateProcess(NULL, commandLine, NULL, NULL, FALSE, 0, NULL, NULL,
                  &startupInfo, &processInfo);
    return 0;
  }
```

Marc van Woerkom has found out a bunch of undocumented debugging options. Turn on Web Start's console (File | Preferences | enhanced | enable console) and add the properties below to your JNLP startup file:

```
<resources>
  <property name="javaws.debug.0" value="+TraceSecurity"/>
  <property name="javaws.debug.1" value="+TraceCache"/>
  <property name="javaws.debug.2" value="+TraceDiskCache"/>
  <property name="javaws.debug.3" value="+TraceDownload"/>
  <property name="javaws.debug.4" value="+TraceXMLParsing"/>
</resources>
```

Q: How can I find out if my app is running under Web Start?

There are several methods. Check for one of these system properties set by Web Start:

Property	Example
javawebstart.version	javaws-1.0.1-ea
jnlpx.heapsize	NULL,NULL
jnlpx.home	C:\JAVA\JWS\V101B
jnlpx.jvm	C:\java\jbuilder\v4\jdk1.3\jre\bin\javaw.exe

Property	Example
jnlpx.remove	true
jnlpx.splashport	1029

Another method is to check if JNLP services are present. Yet another method is adding your very own property to the JNLP descriptor and check if it is present.

Q: How can I prevent my app from being started twice?

One method is to open an obscure server socket port (e.g. `9876`) when your app starts. If the port is already taken, your app is already running and you can shut it down or send a request to the running app or do something completly different. It's all up to you.

Q: How can I get the system's temp directory? getenv("TEMP") doesn't work.

Use the system property `java.io.tempdir`.

Q: Where can I store config files on the user's disk?

The common practice nowadays is to store it under <user.home> (e.g `c:/windows`). Create a subdirectory under <user.home> for your app (e.g `c:/windows/.venus`) and store your config files there (e.g `c:/windows/.venus/profile.xml`).

If you are not interested in running your app offline and you want to get locked into Web Start, you can use Web Start Muffins to store a client ID on the user's disk and keep the rest on your server.

Q: How to share cookies/sessions between Web Start apps and servlets?

Sun's Wireless Dev Site features two articles on session tracking between servlets and wireless apps (aka MIDP midlets). You can cut and paste the code into your Web Start desktop apps almost without any changes.

- Jonathan Knudsen's *Session Handling in MIDP*, January 2002 - `http://wireless.java.sun.com/midp/articles/sessions/`
- Qusay Mahmoud's *MIDP Inter-Communication with CGI and Servlets* , February 2001 - `http://wireless.java.sun.com/midp/articles/servlets/`

`paernoud` posted some code snippets to show how to use a `jsessionid` cookie to share a session between a servlet and a Web Start app. Fast-forward to the appendix for the source code.

When a servlet uses sessions (e.g `HttpServletRequest.getSession`), it adds a special HTTP cookie containing a session ID. This cookie allows it to associate additional HTTP requests from that client (your Web Start app) as part of that session. The HTTP cookie header from a Tomcat servlet might look like this:

```
cookie: JSESSIONID=to1002mC0123456789at
```

Other servers would have slightly different cookies. I think that WebSphere would use `cssessionid` rather than `JSESSIONID` and its value would be somewhat different too. The trick to re-establish that session with future `URLConnection`s is to add the HTTP cookie header back in each time.

How can I pass the session cookie to my Web Start app?

One way is to add the session ID dynamically to the JNLP file and hand it over to your Web Start app so that your Web Start app can add it to the HTTP header when it connects back to the servlet using `HttpURLConnection`.

How do I pick up the session cookie in my servlet?

Within the servlet it all depends on whether this is the first request of the session or a subsequent session. If it's a subsequent session, then it's easy; use `HttpServletRequest.getCookies()` or `getHeaders("cookie")`. If it's the first request, then there is some difficulty since the session has just been created and there's no request cookie yet; in this case, use `getId()` to fetch the ID string.

Unfortunately, unless you were able to do a `getHeaders("cookie")`, you need to come up with the *name* of the session ID cookie since it's not necessarily `JSESSIONID` as it is on Tomcat. The somewhat unsatisifactory way that I've found to get around this is to force the web server to always use `JSESSIONID` using some manual configuration. This has worked for me on WebSphere and iPlanet.

How do I add the session cookie to the HTTP header in my Web Start app?

Once you've done this, you have to synthesize the cookie on the client with something like:

```
http.setRequestProperty("cookie", "JSESSIONID=" + sessionId);
```

assuming an `HttpURLConnection`.

[Editor's Note: I don't pretend to understand what's going on here. I'm going to clear this entry up once I have a better understanding and tried it myself. For now, I leave it as it is in the hope that it is useful to you.]

Q: How can I control Web Start's update policy?

1. You could use a version based download instead of a basic download.
2. Separate your app in two parts: the first part is a stable one with an eager download option, the second part is the one that changes from time to time with a lazy download option; and in the first part (of course, your main class is in the first part), you could use `DownloadService` to check the

cache and do what ever you want.

Q: How can I start Web Start's app manager (aka Web Start Player) automatically at boot time/restart time?

Web Start's app manager is just a simple program `javaws[.exe]`. Put it wherever you put programs for startup on your OS (startup folder under Windows; script in one of the `rc.d` dirs under Unix)

Q: How can I start Web Start's app manager from a web page?

Sun uses the undocumented tag `<player/>`. To fire up Web Start's app manager from a web page add a standard HTML link. Example:

```
<a href="http://java.sun.com/products/javawebstart/apps/player.jnlp">
  Fire Up Web Start Player</a>
```

Or create your very own jnlp file and refer to it. Cut and paste the undocumented jnlp file below:

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

Q: How can I call other programs or shell scripts in Java?

There is no difference from a plain-vanilla Java app other that you need to sign your Web Start app and that you never know in advance what files live on your user's desktop and where they are. Example:

```
String args = {"/bin/sh", "sendMessage.sh" };
Runtime.getRuntime().exec(args);
```

There is an excellent article on `Runtime.exec()` detailing all the pitfalls under Windows at `http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html`

Q: How can I tell Web Start to use either a hotspot or classic virtual machine?

Start up Web Start and use the File | Set Preferences | Java tab to choose a virtual machine (VM) and specify parameters.

You can check if this worked by activating the console via File | Set Preferences | Advanced | Show Console

Note, that you *cannot* pass system properties to your virtual machine in the JNLP file other than `initial-heap-size` and `max-heap-size`.

If you want to pass on additional arguments (such as `-Xincgc`) to the Java runtime, submit a Request For Enhancement (RFE) at Sun's bug parade at `http://java.sun.com/cgi-bin/bugreport.cgi/` and pray that Sun includes it in the next

JNLP spec update. A proposal for stack size args is already in the queue.

Q: How can I retrieve the proxy settings from Web Start?

Use the system properties `proxyHost`, `proxyPort` and `proxySet`. Example:

```
System.getProperty("proxyHost")
System.getProperty("proxyPort")
```

Web Start will set the values for you before it starts up your app.

See Ron Kurr's Java World tip for a proxy property primer online at `http://www.javaworld.com/javaworld/javatips/jw-javatip42.html`

Q: How can I talk to servlets from my Web Start app?

You can use `java.net.URL` to talk to servlets.

Note, that if your app is not signed you can only phone home, that is, talk to your web server that delivered your app to the user's desktop as your app runs in an applet like sandbox.

Q: How can I detect if my app is offline?

One method is to try to resolve a host name. If your app throws `UnknownHostException` because it cannot resolve the host name, you are offline, otherwise you are online. Example:

```
try
{
  InetAddress addr = InetAddress.getByName( host );
  Sytem.out.println( "online" );
}
catch( UnknownHostException ex )
{
  System.out.println( "offline" );
}
```

You may also use the undocumented property `sun.net.inetaddr.ttl` to turn off DNS caching. Example:

```
System.setProperty("sun.net.inetaddr.ttl", "0");
```

By default negative DNS answers are cached and it may happen that dial-in users that were offline will not be able to connect to the host if the first resolve attempt failed (timeout etc.) until the app restarts and the cache is cleared.

There is no way to really tell if you are offline or online. You can try `javax.jnlp.BasicService.isOffline()`, but note that `isOffline()` always returns false unless you switch Web Start's App Manager into offline mode *and* start your app from there resulting in `javaws -offline ...` causing `isOffline()` to return true.

Q: How can I start other Web Start apps from my Web Start app?

You can start other Web Start apps from your Web Start app by passing the app's JNLP startup file URL to your browser using `BasicService.showDocument`. Your browser will then download the startup file and pass it on to Web Start to finish up your request. Example:

```
bs.showDocument( "http://www.jenomics.de/vamp/venus.jnlp" );
```

An alternative is starting Web Start yourself using `Runtime.exec`. You can find out where Web Start hides away on your user's disk using the `javax.home` property.

Q: How can I start a browser without Web Start?

Web Start makes starting a browser using `BasicService.showDocument` easy. To create your own `showDocument` version without Web Start in 100 % Java check out

- *BrowserLauncher* by Eric Albert at `http://browserlauncher.sourceforge.net`
- Java World Tip 66: *Control Browsers from Your Java Application* by Steven Spencer at `http://www.javaworld.com/javaworld/javatips/jw-javatip66.html`

Q: How does a muffin differ from a cookie?

A web browser uses cookies to store small amounts of data on the user's computer. Muffins are similar to cookies in that they store data on the user's computer in the Web Start cache that is uniquely identified by URLs (e.g. `www.vamphq.com`) and available to untrusted (aka sandboxed, unsigned) Web Start apps.

A web browser, however, is only required to accept 20 cookies per site and 300 total per user, and the browser can limit each cookie's size to 4096 bytes. In contrast Web Start apps can use at least up to 128 k for muffins without any permission. If an app needs more space for its muffins, it needs to ask the user for permission.

In contrast to cookies you can also tag muffins as dirty, cached, or temporary to help Web Start decide what muffins to delete if space gets tight. Web Start's deletes temporary muffins first; then cached muffins; and finally dirty muffins. (If Web Start deletes dirty muffins it must ask the user as data might get lost.)

Tag Value	Description
dirty	Server doesn't have an up-to-date copy of the muffin
cached	Server has an up-to-date copy of the muffin
temporary	Muffin is not stored on the server; Muffin can always be recreated

Muffins Without Web Start - Roll Your Own Muffin Storage Service

If you want to use muffins and run your app without Web Start as well, you can create your own muffin storage service. This is not as hard as it sounds. For a start you can check out Mauro Marillini's open-source muffin storage service published in the his Java Deployment with JNLP and Web Start book. Chapter 11: Runtime Client Services is available online for free at Sun's Java Developer Connection (JDC) site at `http://developer.java.sun.com/developer/Books/jnlp/`

Cookies Inside Muffins - Use Cookies In Your Web Start App

Note, that you can use cookies as well in your Web Start app. Web Start apps in contrast to applets, however, run outside a browser and you, therefore, cannot use the browser's cookies but you have to manage your own cookie cache. For unsigned apps you can put your cookie cache inside a muffin and consult it whenever you send a HTTP request back to your home web server, for example.

Q: Can I disable jardiff for selected jars?

If you use `version.xml` files, you can disable jardiff generation by keeping only the jar's current version in the `version.xml` file. To generate jardiffs Sun's JnlpDownloadServlet requires the jar's current and previous version to be listed in `version.xml`.

Q: Why does the Java runtime linger on after my app is gone?

All Swing apps must call `Sytem.exit()` to shutdown the Java runtime. Note, that Web Start turns every app it starts into a Swing app. Therefore, always call `System.extit()` to shutdown the Java runtime even if your app teared down the Java runtime automatically (without calling `System.exit()`) when run stand-alone.

Q: How can I associate file extensions with Web Start apps under Windows?

It's quite a challenge because Web Start doesn't support any command line arguments other then the app's URL (e.g. `javaws http://www.jenomics.de/vamp/venus.jnlp`).

The hard part is to figure out how to add more command line arguments. One method is to lobby Sun, another is to create your own Web Start clone.

Once you succeeded all that's left to do is to add some registry entries to associate your app with your file extension of choice using Windows-only native code.

Q: How does a Web Start app differ from a plain-old app?

A Web Start app differs from a plain-old app in numerous ways including:

- Web Start installs a `SecurityManager`
- Web Start runs your app in a secondary AppContext
- Web Start loads your app's classes using `JNLPClassLoader`

- Web Start creates top-level window(s) (that is, `JFrame`) before your app pops up (thus, `Frame.getFrames()[0]` may not point to your window)

Q: How can I tell Web Start to start my apps offline?

Note, that Web Start cannot automatically detect if your computer is offline or online. Use the Web Start Jukebox (that is, Application Manager) to toogle between offline and online. (Click on the connector-plug icon in the bottom-left corner.)

To start your Web Start app offline from the shell (commandline) use the `-offline` switch. Example:

```
javaws -offline "http://www.jenomics.de/vamp/hazel.jnlp"
```

Note, that `javax.jnlp.BasicService.isOffline()` returns true only if you use the `-offline` switch or if you switched the Web Start Jukebox to offline. Web Start doesn't use any wizardry to second-guess your setting.

If you start your app offline, Web Start skips the update check. Also note that Web Start by default forbids offline startup and requires your blessing (use the `offline-allowed` tag in your app's JNLP file to grant the permission).

Q: Two Event Queues. How come?

If you run your app with Web Start, you will end up with two event queues (`AWT-EventQueue-0` and `AWT-EventQueue-1`).

Web Start also has its own thread group called `javawsApplicationThreadGroup`.

You will also get two application contexts (one for Web Start and one for your app).

If you use RMI and Swing and touch up the GUI in your RMI callbacks than you will face the following threading bug:

RMI creates threads for each connection; but in Web Start they're created within the context of Web Start, not your app. This has the result that if you use `SwingUtilities.invokeLater()` from an RMI thread, it'll be dispatched to the Web Start context, which has its own event queue thread. Your application also has an event queue thread, so your code is running in the wrong thread.

[Editor's Note: I will clear up the following posts once I better understand what's going on.]

Rhys Parsons reports:

I've done some tests and found that when I use `invokeLater()` from a thread created by an RMI server (which is actually a callback interface), the call is synchronized with `AWT-EventQueue-0`; however, when I created my own thread and synchronize it using `invokeLater()`, it's synchronized with `AWT-EventQueue-1`. GUI events (key events and mouse events) occur in `AWT-EventQueue-1`, so updating a GUI from some threads (such as RMI threads) seems to cause a threading problem for the app.

I fail to get key events from a custom `JTextField`. I do, however, still get mouse events.

My GUI is created from events received from an RMI server. The calls are synchronized with the AWT event queue using `invokeLater()`. Items, such as text fields, are also updated from the RMI Server, and the `invokeLater()` method called appropriately.

However, I've noticed that under Web Start there are two AWT event queues: `AWT-EventQueue-0` and `AWT-EventQueue-1`; and whereas the `invokeLater()` calls are synchronized with `AWT-EventQueue-0` when called from RMI threads, GUI events (the mouse events I do get) are occuring in `AWT-EventQueue-1`.

I've tried two test apps. One which has no RMI and used `invokeLater()` in a separate thread. Both key and mouse events are caught correctly in `AWT-EventQueue-1`, and the updates from the `invokeLater()` occur in `AWT-EventQueue-1`.

I then changed the same application to register a callback RMI interface with a server. The server updates a `JTextField` with a new value every second. Sure enough, the updates using `invokeLater()` from the RMI thread occur in `AWT-EventQueue-0`, but key and mouse events happen in `AWT-EventQueue-1`. Eventually the test app crashes.

It seems to me that there is something about the RMI thread which causes `invokeLater()` to synchronize with `AWT-EventQueue-0`, whereas it should be synchronizing with `AWT-EventQueue-1`.

David Geary confirms:

We use RMI callbacks from a server, and everything appears in `AWT-EventQueue-1` except when we call `invokeLater()` from an RMI callback thread, this then executes on `AWT-EventQueue-0`.

`dietz` posts a workaround:

The method called back by RMI runs on a thread in a ThreadGroup created by RMI code. This ThreadGroup is a child of the main system ThreadGroup (which has associated its own AppContext) as opposed to the `JavawsApplicationThreadGroup` (which has associated the application's AppContext). GUI code that runs in this thread will be in the wrong AppContext and, therefore, use the wrong EventQueue. The same problem shows up in applets (which also uses different AppContexts).

As a workaround replace the call to `invokeLater()` with one that will run on the correct AppContext as follows:

```
private final static ThreadGroup _applicationThreadGroup =
  Thread.currentThread().getThreadGroup();

public void myInvokeLater(final Runnable code)
{
   ( new Thread( _applicationThreadGroup, new Runnable()  {
        public void run() { SwingUtilities.invokeLater(code); }
      } )
   ).start();
}
```

With this code the ThreadGroup the class is loaded on is saved and used, so this may not work if the jar containing the class is loaded from other RMI code running on a thread in the wrong ThreadGroup (either implicitly by making the first ref to a resource in a lazy jar, or explicitly using the `JNLPDownloadService`). I haven't tried that but it seems it would also be easy to work around.

Rhys Parsons follows up:

The point is to create a thread which you know will be in your application's context (e. g. create it within the application's main thread), and call `SwingUtilites.invokeLater()` from this thread.

The workaround creates a thread for every call. I think this is wasteful of system resources, so I wrote the class `AppContextSynch` (see the appendix for the source).

To use it, create it within the first few lines of your app. Example:

```
new AppContextSynch();
```

And instead of doing `SwingUtilities.invokeLater()` do `AppContextSynch.invokeLater()`.

Q: How can I find out why Web Start fails to parse my JNLP startup file?

If Web Start chokes on your JNLP startup file (e.g. "Launch File Error: The following required field is missing from the launch file: `<jnlp>`"), and your app won't show, turn on the magic `TraceXMLParsing` switch, to find out why me. Follow these steps:

- go into the Java tab in the app manager's preference panel and change `javaw.exe` to `java.exe` (dropping the trailing `w` tells the Java runtime to pop up a console window that shows every message sent to `stdout`)
- go to the directory where `javaws` itself hangs out (on Windope usually at `C:\Program Files\Java Web Start`)
- create a file called `.javawsrc` containing the line: `TraceXMLParsing=true`
- run `javaws` from the shell and pass in your JNLP file's URL as an argument (e.g. `javaws http://www.jenomics.de/vamp/hazel.jnlp`)
- watch the console window as the story unfolds (scrolls by) and Web Start chokes when swallowing (parsing) your JNLP startup file.

Links

Q: Where can I find more information about Web Start programming?

Here are some developer links that include code examples:

- Sun's Java Web Start Developer's Guide [http://java.sun.com/products/javawebstart/docs/developersguide.html]
- Sun's JNLP Specification [http://java.sun.com/products/javawebstart/download-spec.html] (JSR-56)
- Sun's JNLP and Java Web Start Tech Tips [http://developer.java.sun.com/developer/JDCTechTips/2001/tt0530.html] by Stuart Halloway (May 30, 2001)
- JavaWorld article: Java Web Start to the rescue [http://www.javaworld.com/javaworld/jw-07-2001/jw-0706-webstart.html] : Find out how Java Web Start aids client-side deployment by Raghavan N. Srinivas (July 6, 2001)
- IBM DeveloperWorks article: Java Web Start [http://www.ibm.com/developerworks/java/library/j-webstart/index.html?dwzone=jav : Developing and distributing Java applications for the client side by Steven Kim (September, 2001)

JavaOne 2001 Slides: *Web Start Software: Advanced Topics* by Rene Schmidt and Andy Herrick online at http://java.sun.com/javaone/javaone2001/pdfs/1318.pdf

Topics include:

- Application Partitioning
- Incremental Update
- Java 2 Runtime Installation
- and more

JavaWon 2002 Talk Slides About Web Start

- Andrew Herrick and Steve Bohne (Sun), *Web Start: Advanced Topics [http://servlet.java.sun.com/javaone/sf2002/conf/sessions/display-1591.en-85084.* , (76 slides), Web Server Deployment, Application Partioning, Incremental Update, Security and Sandbox, Server Communication; Upcoming Features in Hopper (Java 1.4.1)
- Rick Spickelmier (CTO Extensity), *Java on the Desktop in the Enterprise [http://servlet.java.sun.com/javaone/sf2002/conf/sessions/display-2251.en-85084.* , (32 slides), Case Study: Rolling Out A Million Java Desktop Apps In 400 Enterprises In 40 Countries; Moved From Java 1 Applets to Java 2 Web Start Apps
- Eric Shapiro (CEO Zero G) and Greg Maletic (CTO Zero G), *'nSync: Keeping Java Clients in Sync with the Back-End through Web Services [http://servlet.java.sun.com/javaone/sf2002/conf/sessions/display-1044.en.jsp]* , (35 slides), high level overview making the case for rich, zero-admin, always-up-to-date Java 2 desktop apps mixed up with marketing hype for Zero G's own closed-source payware claiming to outshine Web Start; dream on Eric

Here are some books that have major Web Start content:

Java Deployment (JNLP, WebStart, J2EE, J2SE) by *Mauro Marinilli*, September 2001, 512 pages, paper format, ISBN: 0-672-32182-3, Sams Publishing

Chapter 2: An Abstract Model for Deployment and Chapter 11: Runtime Client Services of Mauro Marinilli's Java Deployment with JNLP and Web Start book are online for free at Sun's Java Developer Connection (JDC) site at `http://developer.java.sun.com/developer/Books/jnlp/`

Early Adopter: J2SE 1.4 by *James Hart*, September 2001, 200pp, Wrox Press; Wrox has published an updated and expanded version under the new title: *Java J2SE 1.4 Core Platform Update*, March 2002, 250pp, ISBN: 1861007272, as Java 1.4 has long shipped and no longer is avant-garde.

In Chapter 3: Clients James Hart explains JNLP and Web Start and develops a Web Start example app for viewing photographs in a sandboxed and in a signed version.

Gregory M. Travis's *JDK 1.4 Tutorial [`http://www.manning.com/travis/index.html`]* (March 2002, 408 pages, Manning), includes a 30-page chapter on Web Start:

- Understanding the Web Start execution model (Client, server, and application; The sandbox; Consider the possibilities)
- Building and deploying a Web Start application (JAR files; The JNLP file; Configuring the web server)
- Using the sandbox: services; resources
- Bypassing the sandbox
- Example: a simple drawing program (`PicoDraw.java`, `DrawCanvas.java`, `TransferableImage.java`)

Q: Are there any open-source Web Start/JNLP projects?

Yes, there are. Here's the line up. Alive and kicking projects:

- *OpenJNLP* (100 % Java *JNLP client* that runs on any JDK 1.1 platform) - `http://openjnlp.sourceforge.net`
- *NetX* - (100 % Java *JNLP client*; command line only; part of the Object Component Desktop (OCD)) - `http://ocd.sourceforge.net/netx/netxPage.html`
- *Rachel* - (resource loading toolkit) - `http://rachel.sourceforge.net` or `http://freshmeat.net/projects/rachel`
- *Wharf* - (an early stage, open-source Web Start clone part of the JDistro Java desktop) - `http://www.jdistro.com`

NetX Java Start Button Java Start Button displays an always-up-to-date menu of Web Start apps that you start by selecting the desired menu item. As of April 2002, NetX Java Start Button lists about fifty Web Start apps in eight categories. More info at `http://ocd.sourceforge.net/netx/start/start.html`

Dead or neglected projects:

- *Juniper* - (JNLP servlet) - `http://sourceforge.net/projects/juniper`
- *JavaURL* (JNLP client) - `http://www.amherst.edu/~tliron/javaurl`

Heiss Stephan has created a little tool allowing jar creation, code signing and more. You can kick start his little tool using Web Start at `http://62.65.146.182/java/applications/JarCreator.jnlp` (The source is packed up along will all `.class` files in the jar. For further study extract it from the jar stored in Web Start's cache.)

Web Start Alternative Comparison Chart

	Web Start	OpenJNLP	NetX
Source License	Sun Community License (similar to Microsoft's shared source license; you are allowed to look at the code, but not allowed to touch it)	Mozilla Public License (MPL)	GNU General Public License (GPL)
Security	Yes	No security	Yes; sandbox/trusted but no code signing check
Requires Installer	Yes	No (several Jars)	No (single executable Jar)
Platforms	Windows, Linux, Solaris, Mac OS X, HP/UX; uses native code for desktop integration, proxy detection, browser integration etc.	100 % Java 2; no native code; Java 1.1 version for Mac Classic available as separate branch	100 % Java 2; no native code
Command-line support	No command line options; accepts a single URL only	Yes, accepts list of URLs; no other options	Yes, rich support
JNLP compliance			
Overall compliance	Excellent	Fair. Missing: installers, applets, extensions, security checks, JNLP services, and more	Fair. Missing: installers, signing, native code, proper JVM selection, and more
Required JNLP services	All	None (stubs only)	Most
Sets system properties, applet params, application args	Yes	system props and args	Yes
Native Library Support	Yes	Yes	Yes

	Web Start	OpenJNLP	NetX
Extension Descriptors	Yes	No	Yes
Supports Applets	Yes	No	Yes
Supports Jar Diffs and Versioning	Yes	No	No
Supports Proxies	Yes	No	No
API Features			
Embeddable in other programs	No	Yes	Yes
API to Launch in Same JVM	No	Yes	Yes
API to Launch new JVM	Yes (by opening URL)	No	Yes
Access to JNLP file information	Yes, codebase only; missing everything else	Yes, missing: security, applet, etc.	Yes, all
Insert extra properties, params, args at runtime	No	No	Yes, programmatically or command-line
Can use any SAX parser as system default / does not affect launcher	Yes, uses own XML parser	Yes, uses NanoXML parser	Yes, uses TinyXML parser
GUI/system integration			
GUI Tools	Yes, app list (delete, etc) plus console window	Yes, app list plus console window	No, but paired with Object Component Desktop (OCD)
GUI Separate from Launcher	No	Yes	Yes
Automatically associated with .jnlp file extension	Yes	No (manual)	No (manual)
Console / stdio handling	Console window and output log	Console window (combines multi-app output)	All IO to console

Q: Are there any third party tools available to package, sign and publish Web Start/JNLP apps?

Venus Application Publisher offers a rich tool collection to help you package, sign and publish Web Start/JNLP apps. You can find out more at `http://www.vamphq.com`

Q: Where can I find JNLP/Web Start apps?

Check out the following indices that list JNLP/Web Start apps:

- `http://www.up2go.net` - Top 10 List, User and Editor's Ratings, User Comments, Spotlight, New App Submission Form, And More
- `http://www.connectandwork.com/external`
- `http://www.puzzlecode.com/puzzlecode/jnlp/index/` - Categorized App Directory, New App Submission Form
- `http://www.webstartapps.com` - Categorized App Directory, New App Submission Form
- `http://openjnlp.nanode.org/app-list.html` - OpenJNLP's Categorized App Directory
- Bi-Monthly Swing Sightings Series Includes Links To Various Web Start Apps [`http://java.sun.com/products/jfc/tsc/sightings/`]

Q: Where can I download Web Start's source?

You can download Web Start's source from Sun at `http://www.sun.com/software/communitysource/javawebstart/download.html`

Q: Where can I get JnlpDownloadServlet's source?

You can download the source for the JnlpDownloadServlet at `http://developer.java.sun.com/developer/restricted/javawebstart/`. The JnlpDownloadServlet source ships with the Web Start 1.0.1 source bundle. (Java Developer Connection (JDC) login required; membership free-of-charge)

Q: Where do I find the javax.jnlp classes?

Download Sun's JNLP developer's pack at `http://java.sun.com/products/javawebstart/download-jnlp.html` . The `javax.jnlp` classes are packed up in `jnlp.jar`.

Q: Where can I find more info about Web Start programming in German?

- Java Magazin - Workshop: Ihre erste Applikation mit Unterstützung von Suns JNLP-Referenzimplementierung Web Start [`http://www.javamagazin.de/itr/online_artikel/show.php3?id=38&nodeid=11`] von Sven Haiges und Steffen Müller, April 2002

Q: Where can I find more info about Java's class loading architecure?

Books

Stuart Halloway's Component Development for the Java Platform [`http://cseng.aw.com/book/0,3828,0201753065,00.html`] (December 2001, Addison-Wesley) tells you everything you ever wanted to know about class loading and more. The 50+ page coverage includes:

- Chapter 2: The Class Loader Architecture Goals of the Class Loader Architecture; Explicit and Implicit Class Loading; The Class Loader Rules; Hot Deployment; Unloading Classes; Bootclasspath, Extensions Path, and Classpath; Debugging Class Loading; Inversion and the Context Class Loader
- Chapter 5: Customizing Class Loading Java 2 Security (The Role Of Class Loaders); Custom Class Loaders; Protocol Handlers; Getting Past Security to the Loader You Need; Reading Custom Metadata

Ted Neward's Server-Based Java Programming [`http://www.manning.com/Neward3/`] (July 2000, 592 pages, Manning) includes a record-breaking 60+ pages, in-depth coverage:

- Chapter 2: ClassLoaders
  - Dynamic linking (Run-time dynamic loading; Reflection)
  - ClassLoaders: rules and expectations (Java .class file format; Using ClassLoader; java.lang.ClassLoader; Java name spaces)
  - Java's built-in ClassLoaders (java.security.SecureClassLoader; java.net.URLClassLoader; sun.applet.AppletClassLoader; java.rmi.server.RMIClassLoader; Bootstrap ClassLoader; sun.misc.Launcher$ExtClassLoader)
- Chapter 3: Custom ClassLoaders
  - Extending ClassLoader (FileSystemClassLoader; HashtableClassLoader; CompilerClassLoader; StrategyClassLoader and ClassLoaderStrategy; CompositeClassLoader; Other ClassLoader tricks; Other ClassLoaders)
  - On-the-fly code upgrades
  - GJAS: first steps (Goals; Service; Server; ServerManager)

Online Articles

- Ted Neward's Understanding `Class.forName()` tech paper online at `http://www.javageeks.com/Papers/ClassForName/ClassForName.pdf`

Appendix

Java Runtime Installer Example

To use Dale Searle's Java Runtime Installer Example follow these steps:

- Go to your `jdk<version>/jre` directory and jar it (e.g. `jar cf jre.jar *.*`)
- Jar the `jre.jar` into a `JRE_<version>.jar` file
- Sign the `JRE_<version>.jar` file
- Create a `JRE_<version>` directory in `tomcat/ROOT`
- Copy the signed `JRE_<version>.jar` to the `JRE_<version>` directory
- Create a `<version>.properties` file containing the entries below:

```
# Lists the platforms supported by this Java Runtime
locale=en_US
arch=x86
os=Windows
version-id=1.3.1_01
```

- Next, create your JNLP installer file named `<version>.xml`. Example:

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+" codebase="http://marvin/JRE_1.3.1_01">
  <information>
    <title>J2RE 1.3.1_01 Installer</title>
    <vendor>PerMedics, Inc.</vendor>
  </information>
  <security>
    <all-permissions/>
  </security>
  <resources>
    <j2se version="1.3"/>
    <jar href="Installer.jar"/>
    <jar href="JRE_1.3.1_01.jar"/>
    <property name="jre.container" value="JRE_1.3.1_01.jar"/>
    <property name="jre.execute" value="bin\\javaw.exe"/>
    <property name="jre.version" value="1.3.1_01"/>
  </resources>
  <installer-desc main-class="JREInstaller"/>
</jnlp>
```

The `jre.container` property is the name of the jar holding the Java Runtime (e.g. `jre.jar`) that you packed up in the beginning. `jre.execute` is the path to the Web Start executable (e.g `javaw.exe`) and `jre.version` is the version passed on to `service.setJREInfo()`.

Note, in the JNLP installer file use a Java Runtime version such as 1.3+ that is already installed on your user's machine, and not the version for the Java Runtime you try to install.

Make sure that you put both these files into the `JRE_<version>` directory in `tomcat/ROOT`.

Compile the installer (e.g. `JREInstaller.java`), jar it (e.g. `installer.jar`), sign it and put it in the `JRE_<version>` directory in `tomcat/ROOT` as well. Make sure your version tag is the same as your `<version>` string.

That's it. Go nuts.

Java Runtime Installer Servlet Code

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class JREServlet extends HttpServlet
{
  //Initialize global variables
  public void init(ServletConfig config) throws ServletException
  {
    super.init(config);
  }


  //Process the HTTP Get request
  public void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException
  {
    PrintWriter out = new PrintWriter (response.getOutputStream());
    String sep = File.separator;
    String home = System.getProperty("tomcat.home") + sep;

    //Gets put in the tomcat/bin directory
    PrintWriter logWriter = new PrintWriter(
       new FileWriter(home + "logs" + sep +"JREServlet.log"));

    /**
     * Capture parameters, note known-platforms is only used by the servlet
     * to determine if the JNLP client has a version that will run this installer
     */
    String locale         = request.getParameter("locale");
    String arch           = request.getParameter("arch");
    String os             = request.getParameter("os");
    String versionID      = request.getParameter("version-id");
    String knownPlatforms = request.getParameter("known-platforms");

    StringBuffer jnlp = new StringBuffer();
    try
    {
      //I dont care which version of windows it is.
      if (os != null && os.indexOf("Windows") != -1)
        os = "Windows";

      //Log parameters for debug
      logWriter.println("locale=" + locale);
      logWriter.println("arch=" + arch);
      logWriter.println("os=" + os);
      logWriter.println("version-id=" + versionID);
      logWriter.println("known-platforms=" + knownPlatforms);

      /**
       * If the requested JRE is supported then it will be in a directory
       * that follows the convention JRE_[version-id] which
       * contains JRE_[version-id].jar, [version-id].properties and [version-id].xml.
       * The JRE_[version-id].jar contains a jre.jar holding the jre itsself.
       * For example if I request version 1.3.1_01 then I must have a
       * JRE_1.3.1_01 directory which contains JRE_1.3.1_01.jar (which contains jre.jar),
       * 1.3.1_01.properties, and 1.3.1_01.xml files.
       */

      //This directory should exist in the Root directory
      String path = home + "webapps" + sep + "ROOT" + sep;

      logWriter.println("Checking version info.");
      String directoryName = "JRE_" + versionID;
      logWriter.println("JRE directory path = " + path +directoryName);
      logWriter.println("");
      File directory = new File(path + directoryName);

      if( directory.exists() )
        path += directoryName + sep;

      String propPath = path + versionID + ".properties";
      logWriter.println("Properties file = " + propPath);
      File properties = new File(path + versionID + ".properties");

      String xmlPath = path + versionID + ".xml";
      logWriter.println("XML file = " + xmlPath);
      File jnlpFile = new File(xmlPath);


      if (properties.exists() && jnlpFile.exists())
      {
        logWriter.println("Files exist.");

        Properties prop = new Properties();
        FileInputStream fis = new FileInputStream(properties);
        prop.load(fis);
        fis.close();
        logWriter.println("Properties loaded.");

        //Verify that this JRE will be compatible with the requested platform.
        if (!versionID.equals(prop.getProperty("version-id"))
           || !os.equals(prop.getProperty("os")))
        {
          logWriter.println("version-id or os failed validation");
            throw new Exception("11 -version-id or os failed validation");
        }
        logWriter.println("Passed properties validation");

        //Read in the jnlp file
        FileReader fr = new FileReader(jnlpFile);
        BufferedReader reader = new BufferedReader(fr);

        while (reader.ready())
        {
          jnlp.append(reader.readLine().trim());
        }

        reader.close();
        fr.close();
        logWriter.println("Loaded xml.");
        logWriter.println(jnlp);

        response.setContentType("application/x-java-jnlp-file");
        response.setHeader("x-java-jnlp-version-id", versionID);
        out.println(jnlp);
     }
     else
     {
        throw new Exception("10 -Directory or Files do not exist.");
     }
  }
  catch(Exception e)
  {
   if( logWriter != null )
     e.printStackTrace(logWriter);
   else
```

```
      e.printStackTrace();
    response.setContentType("application/x-java-jnlp-error");
    response.setHeader("x-java-jnlp-version-id", versionID);
    out.println(e.getMessage());
   }
   finally
   {
     logWriter.flush();
     logWriter.close();
     out.flush();
     out.close();
   }
  }
}
```

Java Runtime Installer Code

```
import javax.jnlp.*;
import java.io.*;
import java.util.jar.*;
import java.util.zip.*;
import java.util.*;

public class JREInstaller
{
  private static Class clazz;
  private static String installPath;
  private static ExtensionInstallerService service;

  public JREInstaller()
  {
    clazz = getClass();
  }

  public static void main(String[] args)
  {
    PrintWriter writer = null;
    //This is the name of the jar file that holds jre.jar.
    String jarName = System.getProperty("jre.container");
    //Actually a path to javaw.exe e.g. bin\javaw.exe
    String execute = System.getProperty("jre.execute");
    //The version being installed
    String version = System.getProperty("jre.version");


    JREInstaller jreInstaller = new JREInstaller();
    try
    {
      //Set up the log file
      writer = new PrintWriter(new FileWriter("/C:/JREInstaller.log"));
      writer.println("initiated, services available:");

      String[] arr = ServiceManager.getServiceNames();
      for( int i = 0; i < arr.length; i++ )
        writer.println( "\t" + arr );
      service = (ExtensionInstallerService) ServiceManager.lookup(
                   "javax.jnlp.ExtensionInstallerService");

      installPath = service.getInstallPath();
      writer.println("JRE = " + jarName);
      writer.println("execute = " + execute);
      writer.println("Install path = " + installPath);

      if( service == null )
        throw new Exception("Service is null.");

      service.setHeading("Looking for Jar file, please wait.");
      Thread.sleep(2000);

      //Create a temp file to get around URL file problem
      InputStream fis = clazz.getResourceAsStream("jre.jar");
      File temp = File.createTempFile("jre",".tmp");
      temp.deleteOnExit();
      FileOutputStream fos = new FileOutputStream(temp);
      byte[] buff = new byte[512];
      int read = 0;

      while (true)
      {
        read = fis.read(buff);
        if( read == -1 )
         break;
        fos.write(buff, 0, read);
      }
      fos.flush();
      fos.close();
      fis.close();

      //Now that we have an accessable file lets use it.
      JarFile jarFile = new JarFile(temp);

      service.setHeading("Jar found, inflating.");
      Thread.sleep(500);

      //Inflate the jar into the install directory.
      jreInstaller.inflateJar(jarFile);
      writer.println("JRE inflated.");

      Thread.sleep(500);
      service.setHeading("JRE installed, setting path info.");
      service.updateProgress(98);
      Thread.sleep(500);
      service.setHeading("JRE info set.");
      service.updateProgress(100);
      Thread.sleep(500);
      writer.println("Execute path = " + installPath + File.separator + execute);
      service.setJREInfo(version, installPath + execute);
      writer.println("Complete.");
      writer.flush();
      writer.close();
      service.installSucceeded(false);  // no need to reboot
   }
   catch(Exception e)
   {
     try
     {
       e.printStackTrace(writer);
     }
     catch(Exception ex){}

     writer.flush();
     writer.close();
     if (service != null)
       service.installFailed();
   }
   finally
   {
     try
     {
       writer.flush();
       writer.close();
```

```
      }
      catch(Exception ex){}
    }
  }

  private void inflateJar(JarFile jarFile) throws Exception
  {
    Enumeration e = jarFile.entries();
    int count = 0;
    ZipEntry entry = null;
    StringBuffer path = null;
    String name = null;

    while (e.hasMoreElements())
    {
      ++count;
      if (count % 2 == 0)
        service.updateProgress(count/2);

      entry = (JarEntry)e.nextElement();
      name = entry.getName();
      path = new StringBuffer(installPath + File.separator + name);
      service.setHeading("Unzipping " + name);
      Thread.sleep(300);

      //Not woried about meta info.
      if(path.toString().indexOf("META-INF") != -1)
          continue;

      //Create Directories.
      if( entry.isDirectory() )
      {
         path.setLength(path.length() -1); //Get rid of the ending separater.
         File dir = new File(path.toString());
         if( !dir.exists() )
         {
           if( !dir.mkdirs() )
              throw new IOException("Failed to create directory.\n" + path.toString());
         }
      }
      //Create files.
      else
      {
         byte[] buff = new byte[512];
         int read = 0;
         InputStream fis = jarFile.getInputStream(entry);
         FileOutputStream fos = new FileOutputStream(path.toString());
         while( true )
         {
           read = fis.read(buff);
           if( read == -1 )
             break;
           fos.write(buff, 0, read);
         }
         fis.close();
         fos.flush();
         fos.close();
      }
    }
   }
  }
```

Java Runtime Installer Comments

Note, that a lot of the code just tells you what's going on, feel free to cut out what you don't want (e.g. `sleep(300)` inside `inflateJar()` to speed up the Java Runtime explosion).

On Windows NT4 Web Start installs Java Runtimes per user. That is, if three users, use the same app on the same machine Web Start downloads the app only once; the Java Runtime, however, three times. (40 MBs per user -> 120 MB). Why install the apps per machine, but extensions (such as Java Runtimes) per user (in the Web Start's `.ext` directory)?

I noticed that Web Start gives each user a `.javaws` directory The `.javaws` directory contains a `javaws.cfg` file that keeps track of the jre/url key value pairs. Example:

```
#
#Thu Jan 03 11:09:36 PST 2002
javaws.cfg.jre.2.platform=1.3.1_01
javaws.cfg.jre.2.path=C\:\\Program Files\\Java Web Start\\.javaws\\
   cache\\.ext\\E1010084942064\\bin\\\\javaw.exe
javaws.cfg.jre.2.location=http\://neutron\:80/permedics-web/servlet/JREServlet
javaws.cfg.jre.2.product=1.3.1_01
```

If you copy the path info into the other users `javaws.cfg` in `WINNT/Profiles/<user>/.javaws`, Web Start no longer reinstalls your own Java Runtime. Of course, this breaks the zero-admin principle, as admins have to fix the config files to save disk space. If Web Start used the `.javaws` directory in the `All Users` profile, it would install the Java Runtime only once.

Why does Web Start save the Java Runtime URL/Value pairs in the user's profile `jawaws.cfg` if you use the extension installer service for Java Runtime installs?

In "larger departments" roaming profiles are nothing unusual. Now imagine a user logging in on one machine, runs an app using Web Start that installs a Java Runtime on that machine, now logging off and on another machine.

Starting the same app on another machine leads to an error as the URL/Value pair inside the user's `javaws.cfg`, that moved along with the user, still states that the required Java Runtime is installed, which no longer holds true.

I found a work around that makes Web Start use your custom installed Java Runtime for all users. If you update the `javaws.cfg` file in the Web Start directory with your Java Runtime info, Web Start uses the same Java Runtime for all users. Web Start still gives each user its own `javaws.cfg` file and `.javaws` directory but copies the correct path info to it. I inserted the code below right after the installer's `setJREInfo()` call. Granted it's a hack but it works.

```
int start = installPath.indexOf("Java Web Start");
if (start != -1)
{
  String cfgPath = installPath.substring(0,start + 15) + "javaws.cfg";
  writer.println("Looking for javaws.cfg at " + cfgPath);
  File cfgFile = new File(cfgPath);
  if( cfgFile.exists() )
  {
    Properties prop = new Properties();
    fis = new FileInputStream(cfgFile);
    prop.load(fis);
    writer.println("");
    writer.println("\t** backup of javaws.cfg, paste in if something blows");
    prop.list(writer);
    fis.close();
    writer.println("");
    String test = "";
    int i = 0;
    while (test != null)
    {
      test = prop.getProperty("javaws.cfg.jre." + i + ".product");
      ++i;
    }
    --i;
    prop.setProperty("javaws.cfg.jre." + i + ".product", version);
    prop.setProperty("javaws.cfg.jre." + i + ".platform", version);
    prop.setProperty("javaws.cfg.jre." + i + ".path", actualExecute);
    prop.setProperty("javaws.cfg.jre." + i + ".location",
    service.getExtensionLocation().toString());
    writer.println("writing new javaws.cfg");
    fos = new FileOutputStream(cfgFile);
    prop.store(fos,"Rewritten by JREInstaller");
    fos.close();
  }
}
```

Micheal Nadel's Servlet

Micheal Nadel full-length, uncensored servlet source. For questions, comments, suggestions or improvements send a mail to Micheal Nadel (mNadel@chicagojava.com).

```
import javax.servlet.*;
import javax.servlet.http.*;

import java.io.*;
import java.util.*;

public class DynamicJNLPServlet extends HttpServlet
{
  private HashMap   jnlpMap = new HashMap();
  private ArrayList matchList;
  private boolean   parseQueryString;

  public void init( ServletConfig config ) throws ServletException
  {
    super.init( config );

    this.initMatchList();

    this.parseQueryString =
          config.getInitParameter( "parse.querystring" ).equalsIgnoreCase( "true" ) ? true

    config.getServletContext().setAttribute( "DynamicJNLPServlet", this );
  }

  private void initMatchList()
  {
    this.matchList = new ArrayList();
    String toMatch = getServletConfig().getInitParameter( "match.list" );

    StringTokenizer tok = new StringTokenizer( toMatch, "|" );
    while( tok.hasMoreTokens() )
      this.matchList.add( tok.nextToken() );
  }

  private String getPropertyXML( String name, String value )
  {
    return "<property name=\"" + name + "\" value=\"" + value + "\"/>\n";
  }

  public void clearCache()
  {
    this.jnlpMap.clear();
  }

  private StringBuffer getJnlpFromDisk( HttpServletRequest req, String match ) throws IOExc
  {
    String uriSansContext = req.getRequestURI().substring(
      req.getRequestURI().indexOf( req.getContextPath() ) + req.getContextPath().length() )

    String filepath = this.getServletContext().getRealPath( uriSansContext );

    //System.out.println( " URI: " + req.getRequestURI() );
    //System.out.println( "Path: " + filepath );

    if( this.jnlpMap.containsKey( filepath ) )
      return (StringBuffer) this.jnlpMap.get( filepath );

    int ch;
    StringBuffer sb = new StringBuffer();
    BufferedReader br = new BufferedReader( new FileReader( filepath ) );

    while( ( ch = br.read() ) > -1 )
      sb.append( (char) ch );


    br.close();

    this.replaceCodebase( req, sb );
    this.replaceDisplayName( match, sb );
    this.replaceHref( req, sb );

    this.jnlpMap.put( match, sb );

    System.out.println( "[" + new Date() + "] Found JNLP Template" );
    System.out.println( "     URL: " + HttpUtils.getRequestURL( req ) );
    System.out.println( "   Match: " + match );
    System.out.println( "    File: " + filepath );

    return sb;
  }

  private void replaceCodebase( HttpServletRequest req, StringBuffer sb )
  {
    int start = sb.toString().indexOf( "$$codebase" );
    int length = "$$codebase".length();

    String url = HttpUtils.getRequestURL( req ).toString();

    String codebase = url.substring( 0, url.lastIndexOf( "/" ) );

    sb.replace( start, start+length, codebase );
  }
```

```
private void replaceDisplayName( String matchName, StringBuffer sb )
{
  int start = sb.toString().indexOf( "$$title" );
  int length = "$$title".length();

  String name = this.getServletConfig().getInitParameter( matchName + ".title" );

  sb.replace( start, start+length, name );
}

private void replaceProperties( HttpServletRequest req, StringBuffer sb )
{
  if( sb.toString().indexOf( "$$properties" ) < 0 )
    return;


  StringBuffer propXmlBuffer = new StringBuffer();
  Enumeration enum = req.getParameterNames();
  while( enum.hasMoreElements() )
  {
    String name = (String) enum.nextElement();
    propXmlBuffer.append( this.getPropertyXML( name, req.getParameter( name ) ) );
  }

  int start = sb.toString().indexOf( "$$properties" );
  int length = "$$properties".length();

  sb.replace( start, start+length, propXmlBuffer.toString() );
}

private void replaceHref( HttpServletRequest req, StringBuffer sb )
{
  int start = sb.toString().indexOf( "$$href" );
  int length = "$$href".length();

  String url = HttpUtils.getRequestURL( req ).toString();

  String href = url.substring( url.lastIndexOf( "/" ) + 1 );

  sb.replace( start, start+length, href );
}

private String getJnlp( HttpServletRequest req ) throws IOException
{
  String matchName = "", url = HttpUtils.getRequestURL( req ).toString();
  boolean foundMatch = false;

  for( int i = 0; i < this.matchList.size(); i++ )
  {
    matchName = (String) this.matchList.get( i );
    if( url.indexOf( matchName ) > -1 )
    {
      foundMatch = true;
      break;
    }
  }

  if( !foundMatch )
    return null;

  StringBuffer jnlpBuffer = this.getJnlpFromDisk( req, matchName );

  if( this.parseQueryString )
  {
    StringBuffer tmpBuffer = new StringBuffer( jnlpBuffer.toString() );
    this.replaceProperties( req, tmpBuffer );
```

```
      return tmpBuffer.toString();
    }
    else
      return jnlpBuffer.toString();
  }

  protected void service( HttpServletRequest req, HttpServletResponse res )
     throws ServletException, IOException
  {
    try
    {
      String jnlp = this.getJnlp( req );
      if( jnlp == null )
        res.sendError( HttpServletResponse.SC_NOT_FOUND );
      else
      {
         res.setContentType( "application/x-java-jnlp" );
         res.getWriter().print( jnlp );
         res.flushBuffer();
      }
    }
    catch( IOException e )
    {
      System.err.println( "DynamicJNLPServlet Error: " + e.getMessage() );
      res.sendError( HttpServletResponse.SC_NOT_FOUND );
    }
  }
}
```

And the servlet's `web.xml` config file:

```
<web-app>
    <servlet>
        <servlet-name>
            DynamicJNLPServlet
        </servlet-name>

        <servlet-class>
            DynamicJNLPServlet
        </servlet-class>

        <!-- Map URIs to JNLPs and Display Name  -->
        <init-param>
            <param-name>match.list</param-name>
            <param-value>Dev-Test|Dev|Stable|Prod</param-value>
        </init-param>

        <init-param>
            <param-name>parse.querystring</param-name>
            <param-value>true</param-value>
        </init-param>

        <init-param>
            <param-name>Dev.title</param-name>
            <param-value>Nightly</param-value>
        </init-param>

        <init-param>
            <param-name>Dev-Test.title</param-name>
            <param-value>Test</param-value>
        </init-param>

        <init-param>
            <param-name>Stable.title</param-name>
            <param-value>Stable</param-value>
        </init-param>
        <init-param>
            <param-name>Prod.title</param-name>
            <param-value>Prod</param-value>
        </init-param>

        <load-on-startup>1</load-on-startup>

    </servlet>

    <servlet-mapping>
        <servlet-name>
            DynamicJNLPServlet
        </servlet-name>

        <url-pattern>
            *.jnlp
        </url-pattern>
    </servlet-mapping>
</web-app>
```

Agnes Juhasz's JCE Policy Installer

Agnes Juhasz donated the class below that installs the 'Unlimited Strength Cryptography Extension Policy Files" into the current `JAVA_HOME/jre/lib/security` folder. Before using it you have to bundle the proper files into a signed jar (e.g. `UnlimitedCryptoPolicy.jar`) and update your installer's JNLP file with the line: `<jar href="UnlimitedCryptoPolicy.jar"/>` in the `<resources>` section.

```
import java.util.*;
import java.io.*;

public class UnlimitedCryptoPolicyInstaller
{
  /** Put here the names of files to install. */
  String[] FILE_NAMES = { "local_policy.jar", "US_export_policy.jar"};

  /** Put here the sizes for checking. */
  long[] FILE_SIZES = { 2643, 2631 };

  /** Put here the dates of files for checking. */
  GregorianCalendar[] FILE_DATES = { new GregorianCalendar(2002,1,12), new GregorianCalenda

  public UnlimitedCryptoPolicyInstaller()
  {
  }

  /**
   * The install procedure needs 'ALL PERMISSION'.
   * @return a vector with the names of installed files with full pathname or an empty vect
   */
  public Vector doInstall() throws SecurityException,IOException
  {
    Vector installedfilenames = new Vector();

    ClassLoader classloader = getClass().getClassLoader();

    String java_home = System.getProperty("java.home");
    if( java_home == null )
       throw new SecurityException("Unable to detect JAVA_HOME");

    String destinationfolder = java_home+File.separator+"lib"+File.separator+"security";
    for( int i=0; i < FILE_NAMES.length; i++ )
    {
      boolean needinstall = false;
      File file = new File(destinationfolder,FILE_NAMES);

      if( file.exists() && file.canRead() )
      {
        if( file.length() != FILE_SIZES )
          needinstall = true;

        GregorianCalendar olddate = new GregorianCalendar();
        olddate.setTime(new Date(file.lastModified()));
        olddate.set(Calendar.HOUR_OF_DAY,0);
        olddate.set(Calendar.MINUTE ,0);
        olddate.set(Calendar.SECOND ,0);
        olddate.set(Calendar.MILLISECOND,0);

        if ( FILE_DATES.after(olddate) )
          needinstall = true;
      }
      else
      {
        needinstall = true;
      }

      if( needinstall )
      {
        InputStream is = classloader.getResourceAsStream(FILE_NAMES);
        if( is != null )
        {
          // read-in
          ByteArrayOutputStream baos = new ByteArrayOutputStream();
          int j;
          while( (j = is.read()) != -1 )
          {
            baos.write(j);
          }

          // write-out
          file.createNewFile();
          FileOutputStream fos = new FileOutputStream(file);
          fos.write(baos.toByteArray());
          fos.flush();
          fos.close();

          // set the proper date
          file.setLastModified(FILE_DATES.getTime().getTime());
          installedfilenames.addElement(new String(file.getPath()));
        }
      }
    }
    return installedfilenames;
  }

  public static void main(String[] args)
  {
    try
    {
      UnlimitedCryptoPolicyInstaller installer = new UnlimitedCryptoPolicyInstaller();
      System.out.println("Installed files: "+installer.doInstall());
      System.exit(0);
    }
    catch( Exception e )
    {
      System.out.println( e.toString() );
    }
  }
}
```

You could use the code snippet below for a silent background installation:

```
Runnable r = new Runnable()
{
  public void run()
  {
    try
    {
      UnlimitedCryptoPolicyInstaller installer = new UnlimitedCryptoPolicyInstaller();
      Vector v = installer.doInstall();
      int n = v.size();

      String log = new String();
      if( n==0 )
        log = "Unlimited Strength Cryptograpy detected, OK.";
      else
        log = "Unlimited Strength Cryptograpy installed: "+v+" OK.";

      System.out.println(log);
    }
    catch ( Exception e )
    {
      ...
    }
  }
};

Thread t = new Thread( r, "MyApp: Unlimited Crypto Installer" );
t.start();
...
```

Rhys Parsons's RMI Callback GUI Threading Bug Workaround App-ContextSynch Class

```
import java.util.ArrayList;
import javax.swing.SwingUtilities;

/*
* Synchronize the SwingUtilities.invokeLater() with whichever thread created
* this object. Note that it is essential that this is created in the right
* thread (not an RMI / system thread) for this to work correctly.
*
* @author Rhys Parsons
*/

public class AppContextSynch
{
  /** The single instance. */
  private static AppContextSynch instance;

  /** List of runnable interfaces to be run. */
  private ArrayList list = new ArrayList();

  /** Semaphore for poking thread to run interface. */
  private Object lock = new Object();

  /**
   * Creates an AppContextSynch. Since this is a singleton, we
```

```
     * only allow our instance to be set once.
     */
   public AppContextSynch()
   {
      if( instance == null )
      {
         instance = this;

         ( new Thread( new Runnable() {
            public void run()
            {
               Runnable r;
               while( true )
               {
                  if( list.size() > 0 )
                  {
                     synchronized( list )
                     {
                       while( list.size() > 0 )
                       {
                          r = (Runnable) list.remove( 0 );
                          SwingUtilities.invokeLater( r );
                       }
                     }
                  }
                  else
                  {
                     try
                     {
                        Thread.sleep(10);
                     }
                     catch (InterruptedException ie) {}
                  }
               }
          }
        })).start();
     }
   }

   /**
   * Adds code to the interface list.
   * @param r The runnable interface to call in the AWT event queue thread.
   */
   private void addInterface(Runnable r)
   {
    synchronized( list )
    {
        list.add(r);
    }
   }

   /**
   * Runs SwingUtilities.invokeLater() using the thread context that
   */
   public static void invokeLater( Runnable r )
   {
    if( instance != null )
    {
        instance.addInterface(r);
    }
    else
    {
       throw new RuntimeException( "No AppContextSync created." );
    }
   }
 } // end of class
```

paernoud's Session/Cookie Code Snippets

Try this in your Web Start app:

```
String jsessionid = null;
URL servletURL = "url to servlet";

URLConnection connServletInfo = servletURL.openConnection();
connServletInfo.setDoOutput( true );
if( jsessionid != null )
{
  connServletInfo.setRequestProperty( "Cookie", jsessionid );
}

PrintWriter out = new PrintWriter( connServletInfo.getOutputStream() );
out.print( "some parameters here" );
out.close();

String key = null;
for( int i=1; (key=connServletInfo.getHeaderFieldKey( i )) != null; i++)
{
   if( key.equals( "Set-Cookie" ) == true && jsessionid == null )
   {
     jsessionid = connServletInfo.getHeaderField( i );
     break;
   }
}
```

On the server try this:

```
public void service( HttpServletRequest req, HttpServletResponse res)
   throws IOException, ServletException
{
  Cookie userCookie = null;

  try
  {
    // some stuff to do here

    userCookie = setSession( req );
    setResponse( res, userCookie, "some stuff here in Map class" );
  }
  catch( Exception ex )
  {
    throw new ServletException( ex.getMessage(), ex.getCause() );
  }
}

private Cookie setSession( HttpServletRequest req )
{
  boolean isNewCookie = false;
  Cookie[] cookies = req.getCookies();
  Cookie userCookie = null;

  // must be declared before any 'out' declarations.
  HttpSession session = req.getSession( true );
  if( session.isNew() == true )
  {
    isNewCookie = true;
    userCookie = new Cookie( "jsessionid", session.getId() );
  }
  else
  {
    for( int i=0; i < cookies.length; i++)
    {
      if( cookies.getName().equals( "jsessionid") == true )
      {
        isNewCookie = false;
        userCookie = cookies;
        break;
      }
    }
  }
  return( userCookie );
}

private void setResponse( HttpServletResponse res, Cookie userCookie, Map userInfo )
  throws IOException
{
  StringBuffer sb = new StringBuffer( 1024 );

  // some stuff to put into sb etc....

  res.addCookie( userCookie);
  res.setContentType( "text/bin");
  ServletOutputStream srvOut = res.getOutputStream();
  svrOut.write( sb.toString().getBytes() );
  svrOut.close();
  return;
}
```

What's new?

Q: July 22nd, 2002

Contributors

Jean-Baptiste Bugeaud, Mike Callahan, dietz, David Geary, Kevin Herrboldt, Andrew Herrick, Jean-Francois Morin, paernoud, Rhys Parsons, Andreas Weder, Marc van Woerkom

New Entries

- Can I create my own Java runtime installers without breaching Sun's license?
- How can I change the splash screen?
- How can I turn off Web Start's splash screen?
- How can I create a shortcut on the first launch?
- How can I auto-download an international Java runtime?
- What's the best way to move my Web Start apps to a different server?
- How can I install my own authenticator?

- Where can I get a free, zero-dollar Web Start key certificate?
- Do signed jars with a certificate that expired still work?
- Unable to sign jar. How come?
- How can I remove a signature from a jar?
- Why doesn't 1.3.1_03 work as a Java runtime platform version?
- What properties can I pass on to the Java runtime?
- How does a Web Start app differ from a plain-old app?
- How can I tell Web Start to start my apps offline?
- Two Event Queues. How come?
- How can I find out why Web Start fails to parse my JNLP startup file?

Q: May 19th, 2002

Contributors

Kyle Chalupa, dietz, Thomas Gulden, Agnes Juhasz, moa, Michael Nadel, Harald Sack, Franklin Schmidt

New Entries

- Is Java Dead On The Desktop?
- What's the catch?
- What alternatives exist to Web Start and Java for rich cross-platform, zero-admin desktop apps?
- Who is using Web Start?
- How does Web Start differ from Applets?
- Why does Web Start use two javaws.cfg files?
- How can I add arguments to an installer?
- Why Can't Web Start Create Desktop Shortcuts?
- Where can I get JnlpDownloadServlet's source?
- How can I autoinstall Web Start?
- What autodownloadable Java Runtimes does Sun offer?

Updated Entries

- Why can't Web Start do ...?
- Where can I find more information about Web Start programming?
- Can I use native libraries for my apps?

Q: April 1st, 2002

Contributors

Dean Cording, Thomas Ernst, Christian Menne, Manfred Riem, Masahiro Takatsuka

New Entries

- Has Web Start won any Awards?
- Can I use my own custom ClassLoader?
- How can I protect my code in jars?
- Where can I find certificate authorities?
- Where can I find more info about Web Start programming in German?
- Where can I find more info about Java's class loading architecure?

Updated Entries

- Why can't Web Start do ...?
- How to share cookies/sessions between Web Start apps and servlets?
- How can I use Web Start and JSSE together?
- Are there any open-source Web Start/JNLP projects?
- Where can I find JNLP/Web Start apps?
- Where can I find more information about Web Start programming?

Q: February 26th, 2002

Contributors

Bruce Houghton, Brett Humphreys, Dale King, Jon A. Maxwell, Robert Mohn, Ferghil O'Rourke

New Entries

- What's New In Web Start?
- Can I distribute Web Start apps without putting them on a web server (aka CD installers)?
- Can I run Web Start Apps on a headless (monitor-less) UNIX system?
- How can I install Web Start on Unix as root once for all users?
- Can I use Windows UNC names (aka shared network drives) in file:// URLs?
- Which trusted root certificates ship with Web Start?

Updated Entries

- How can I add JNLP MIME-types to my ISP's Apache Web Server?
- How can I use Web Start and Comm API together?
- Are there any open-source Web Start/JNLP projects?

Q: January 24th, 2002

Contributors

Patric Bechtel, Todd Dunst, Robert Mohn, Oleg Ryaboy, Dale Searle, Tony Sze, Marc van Woerkom

New Entries

- Why does the Java runtime linger on after my app is gone?
- How can I associate file extensions with Web Start apps under Windows?
- Where can I find JNLP's DTD?

Updated Entries

- Can I use my own URLs for downloading JREs?
- Why can't Web Start and Microsoft Proxy Server get along?
- Can Web Start handle multiple XML parsers at once?

- How can I debug apps under Web Start?
- Where can I find more information about Web Start programming?
- Are there any open-source Web Start/JNLP projects?

December 14th, 2001

Contributors

Todd Dunst, Mathias Gronert, Kevin Herrboldt, Scott Hughes, Amit Jnagal, Marco Maier, Fredrik Skanberg, Mik Tuver

New Entries

- How can I start other Web Start apps from my Web Start app?
- How can I start a browser without Web Start?
- Is Web Start a general installer?
- How does a muffin differ from a cookie?
- How do I know that jardiff is working?
- Can I disable jardiff for selected jars?
- How can I use jars signed by someone else?
- Where can I download Web Start's source?
- How can I sign jars?
- Where do I find the javax.jnlp classes?

Updated Entries

- Can I use Web Start to deploy server apps?
- Can I use Web Start to deploy apps to mobile devices (J2ME)?
- How can I change the Web Start application folder/cache?
- How can I use Web Start and Comm API together?
- Can I use my own URLs for downloading JREs?
- Why can't Web Start do ...?

- Where can I find more information about Web Start programming?

November 12th, 2001

Contributors

Cristian Bledea, Richard Colvin, Brent Fisher, Michael Garrahan, Chris Lambert, Benoit Marchal, Marc Prud'hommeaux, Austin Shelton, Sergey Starosek, Patrick Taylor, Anthony Vito, Cory Wandling

New Entries

- How can I list all resources in a jar?
- How can I talk to servlets from my Web Start app?
- How can I use Web Start and RMI together?
- How can I detect if my app is offline?
- Can I use Web Start to deploy server apps?
- Can I use Web Start to deploy apps to mobile devices (J2ME)?
- Why can't I start Web Start's app manager under Windows? Why does the app manager's preferences dialog fail to appear?
- Why doesn't Web Start reprompt for proxy logins?
- Can Web Start handle multiple XML parsers at once?

Updated Entries

- How can I add JNLP MIME-types to my ISP's Apache Web Server?
- Is Web Start available for Macintosh?
- How can I connect to a database?

October 15th, 2001

Contributors

Jay Colson, Gavin Everson, Thomas Guelden, Kevin Herrboldt, Brian Larson, John Munsch, Ivan Ooi, Marc van Woerkom, Ivan Zhidov

New Entries

- Is Web Start available for Macintosh?
- Does Web Start support downloading jars using FTP?
- How can I start Web Start's app manager (aka Web Start Player) automatically at boot time/restart time?
- How can I start Web Start's app manager from a web page?
- How can I tell Web Start to use either a hotspot or classic virtual machine?
- How can I connect to a database?
- How can I reference a Java Help helpset?
- How can I change the proxy configuration in javaws.cfg?
- How can I retrieve the proxy settings from Web Start?
- How can I call other programs or shell scripts in Java?
- Why doesn't my app run under Web Start?

September 18th, 2001

Contributors

John Archer, Thomas Gylden, David Harvey, Magnus Kessler , Paul Lucassen, Marc van Woerkom

New Entries

- How can I use Web Start and JCE together?
- How can I use Web Start and JAAS together?
- How can I use Web Start and JSSE together?
- How can I use Web Start and Comm API together?
- How can I use Web Start and Java 3D together?
- Can I use native libraries for my apps?
- Can I make Web Start use my own policy file?
- Can I change my apps look and feel?
- Why doesn't Sun's Javascript Web Start detection script work for my household browser?
- Does Web Start support Mozilla?

- Can I use my own URLs for downloading JREs?
- Why can't Web Start and Microsoft Proxy Server get along?
- Can I install Web Start on Windows without admin privileges?
- How can I use Web Start and Jini together?

August 2nd, 2001

Contributors

Jenna Tottenham

New Entries

- How can I turn off the sandbox?
- Web Start returns a Bad MIME Type error. What's wrong?

July 27th, 2001

New Entries

- What does the upcoming support of Web Start in Java 1.4 look like?
- Can I use Web Start for command line/batch apps?
- Can Web Start update itself?
- Does Web Start support SSL?
- Can I use Web Start for Intranet apps?

Updated Entries

- How can I pass in arguments to my app using a HTML hyperlink?

July 24th, 2001

Contributors

Kevin Dean, Stan Jordan, Richard Robinson, Dian Xu

New Entries

- How to share cookies/sessions between Web Start apps and servlets?
- How can I control Web Start's update policy?
- Can I use Sun's J2EE Reference Implementation EJBs?
- How can I connect to Borland's AppServer?

Updated Entries

- How can I debug apps under Web Start?

Contributors

Thanks to all the JDC members posting their insights at Sun's Java Web Start/JNLP discussion forum. If you think you deserve credit, but I forgot you, please mail to comments@vamphq.com.
Contributors

John Archer, Gerald Bauer, Patric Bechtel, Cristian Bledea, Jean-Baptiste Bugeaud, Mike Callahan, Kyle Chalupa, Jay Colson, Richard Colvin, Dean Cording, Kevin Dean, dietz, Todd Dunst, Thomas Ernst, Gavin Everson, Brent Fisher, Michael Garrahan, David Geary, Mathias Gronert, Thomas Gulden, David Harvey, Kevin Herrboldt, Andrew Herrick, Bruce Houghton, Scott Hughes, Brett Humphreys, Amit Jnagal, Stan Jordan, Agnes Juhasz, Magnus Kessler, Dale King, Chris Lambert, Brian Larson, Paul Lucassen, Marco Maier, Benoit Marchal, Jon A. Maxwell, Christian Menne, moa, Robert Mohn, Jean-Francois Morin, John Munsch, Michael Nadel, Ivan Ooi, Ferghil O'Rourke, John Ott, paernoud, Rhys Parsons, Marc Prud'hommeaux, Andreas Rammelt, Manfred Riem, Richard Robinson, Oleg Ryaboy, Harald Sack, Franklin Schmidt, Dale Searle, Austin Shelton, Fredrik Skanberg, Sergey Starosek, Tony Sze, Masahiro Takatsuka, Patrick Taylor, Paul Tetley, Jenna Tottenham, Mik Tuver, Anthony Vito, Cory Wandling, Andreas Weder, Marc van Woerkom, Dian Xu, Ivan Zhidov