Inside Mac OS X

# Java 1.3.1 Update 1 Release Notes

February 2002

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010



Simultaneously published in the United States and Canada.

# Contents

# Java 1.3.1 Update 1 Release Notes

This document addresses issues pertinent to the Java implementation on Mac OS X 10.1 with Java 1.3.1 Update 1 installed. If you do not have the update installed you can obtain it through the Software Update control panel in System Preferences or you can download the installer independently from http://www.apple.com/downloads/macosx/apple/. You can determine programatically if this update is installed by looking at the `mrj.version` system property, whose value should be `3.2`. A user can determine whether they have the update installed by looking for the `Java1.3.1Update1.pkg` installer receipt in `/Library/Receipts`.

Release Notes for previous versions of Java on Mac OS X are available at http://developer.apple.com/techpubs/java/.

## Filing and Tracking Bugs

If you find issues with the implementation of Java that are not represented in this document or want to follow the resolution of an issue, you may do so online through Radar, Apple's bug tracking system. To access Radar, you will need an Apple Developer Connection (ADC) account. You can view the ADC membership options, including the free online membership at, http://developer.apple.com/membership/index.html. With an ADC membership, you can file and view bugs at http://bugreport.apple.com. When filing new bugs for Java on Mac OS X, please use `Java (new bugs)` for the Component and `X` as the Version.

CHAPTER 1

# Java 1.3.1 Update 1 Release Notes

# New Features in the Java 1.3.1 Update 1

This release is an update to the Java 1.3.1 implementation found in Mac OS X version 10.1. This update delivers improvements for text, mouse, and printing components, as well as improving overall stability and compatibility. It is a recommended update for both developers and your customers.

It includes a completely updated implementation of text components to resolve a number of issues found in the use of `TextAreas` and `TextFields`. Mouse behavior is improved, especially for mouse events that involve dragging in complex applications. Many printing issues have been resolved, including multi-page printing and clipping issues.

Other significant changes are overviewed below.

## mrj.version System Property

With this update, the `mrj.version` system property is set to 3.2.

The Java implementation in Mac OS X does not draw a distinction between the Java Development Kit (JDK) and the Java Runtime Environment (JRE) as on some platforms. Since both development components as well as the runtime environment are included with a default installation of Java on Mac OS X, the `mrj.version` system property is used to designate the specific overall Java implementation in Mac OS X.

# MRJ Preferences Handler

In addition to the `MRJAboutHandler` and `MRJQuitHandler` present in the previous release of Java, this release includes the `MRJPrefsHandler`. This allows your Java application's preferences panel to be opened from the Application Menu. To take advantage of this additional functionality you need to:

- Import `com.apple.mrj.MRJPrefsHandler`
- Implement the `MRJPrefsHandler` interface
- Define a `handlePrefs()` method with a return type of `void`. This method will be called when a user selects About *YourApplicationName* from the Application Menu. In this method you should display your preferences panel.

The `MRJPrefsHandler` behaves like the `MRJAboutHandler` and the `MRJQuitHandler`. Examples of their usage can be found in a new Project Builder Java Swing Application.

# Hardware Acceleration

Mac OS X allows you to take advantage of hardware graphics acceleration for your Java Swing graphics. If enabled, this technology passes swing graphics calls directly to the video card. This can result in significant speed increases for your graphics intensive Java applications. Java 1.3.1 Update 1 includes an implementation of hardware graphics acceleration that is much improved from previous releases of this technology. Along with increased robustness and effectiveness of the Java graphics hardware acceleration in this release, the procedure for turning it on has changed.

Previous releases of Java on Mac OS X used the `com.apple.hwaccel` flag to turn hardware acceleration on. In this release though, you use the `com.apple.hwaccellist` flag followed by a list of video cards that you want hardware acceleration turned on for. This gives you more flexibility in providing

the best end user experience for your application. If your application does not need hardware video acceleration, do not supply the `com.apple.hwaccellist` flag. If hardware acceleration only shows marked improvement on systems with higher end video cards, only turn it on for those systems.

"Supported Graphics Cards" (page 9) gives more information on determining the video card names to use in the list. "Enabling Hardware Acceleration From the Command Line" (page 10) and "Enabling Hardware Acceleration In Mac OS X Application Bundles" (page 10) give examples of how to turn it on in your application.

# Supported Graphics Cards

Java hardware graphics acceleration is supported on most of the video cards provided by default in Apple's computers. For the purpose of Java graphics hardware acceleration, each of these cards has been designated a distinct identifying string. When invoking hardware acceleration, you specify which graphics card's hardware acceleration should be turned on for by setting the `com.apple.hwaccellist` system property to the appropriate string or list of comma separated strings. Examples follow in "Enabling Hardware Acceleration From the Command Line" (page 10) and "Enabling Hardware Acceleration In Mac OS X Application Bundles" (page 10).

You can determine a specific system's designation by running the `hwaccel_tool`, viewing the list in `/Library/Java/Home/lib/glconfigurations.properties` or seeing what is specified inTable 2-1.

**Table 2-1** Java Hardware Graphics Acceleration Video Card Designation Strings

| Video Card Model | Memory | String |
|---|---|---|
| ATI Rage 128 | 16MB | ATIRage128_16777216 |
| ATI Radeon | 16MB | ATIRadeon_16777216 |
| ATI Rage 128 | 32MB | ATIRage128_33554432 |
| ATI Radeon | 32MB | ATIRadeon_33554432 |

**Table 2-1** Java Hardware Graphics Acceleration Video Card Designation Strings (continued)

| Video Card Model | Memory | String |
|---|---|---|
| NVidia GeForce2 | 32MB | NVidia11_33554432 |
| NVidia GeForce3 | 64MB | NVidia20_67108864 |
| NVidia GeForce4 | 64MB | NVidia20_134217728 |

# Enabling Hardware Acceleration From the Command Line

To run your application from the command line with hardware acceleration turned on, you simply pass in the `com.apple.hwaccellist` flag followed by a list of video card designation strings. An example of turning on hardware acceleration for a single video card follows:

```
java -Dcom.apple.hwaccellist=NVidia20_67108864 -jar App.jar
```

For multiple cards:

```
java -Dcom.apple.hwaccellist=ATIRage128_16777216,ATIRage128_33554432 -jar
App.jar
```

# Enabling Hardware Acceleration In Mac OS X Application Bundles

If you distribute your Java application within a Mac OS X application bundle you need to modify the application properties to include the hardware acceleration flags. If you are modifying a previously built application:

1. Navigate to the appropriate property file in your applications bundle. For an application built with MRJAppBuilder this file is in `YourApplicationBundle.app/Contents/Info.plist`. For an application built in Project Builder it is in`YourApplicationBundle.app/Contents/Info.plist`.

**Note:** The Finder hides the contents of an application bundle, as well as the `.app` extension. To show an application bundle's contents in the Finder, you can control click on the application icon and choose Show Package Contents.

2. Add a `com.apple.hwaccellist` key with a comma separated list of values from Table 2-1 (page 9). Both of these keys are set under the Java dictionary in the Application dictionary hierarchy. You may use any text editor or `/Developer/Applications/Property List Editor`. A simple example of a modified property list follows:

```
<dict>
    <string>????</string>
    <key>CFBundleVersion</key>
    <string>0.1</string>
    <key>Java</key>
    <dict>
        <key>ClassPath</key>
        <string>$JAVAROOT/SwingTest.jar</string>
        <key>MainClass</key>
        <string>SwingTest</string>
        <key>Properties</key>
        <dict>
            <key>com.apple.macos.useScreenMenuBar</key>
            <string>true</string>
            <key>com.apple.mrj.application.apple.menu.about.name</key>
            <string>SwingTest</string>
            <key>com.hwaccellist</key>
            <string>ATIRage128_16777216,ATIRage128_33554432</string>
        </dict>
    </dict>
</dict>
</plist>
```

If you are building a new application in Project Builder you can set these values in the Applications Settings tab.

1. With your project open in Project Builder click the Targets tab.
2. Select Edit Active Target from the Project menu.
3. Click the Application Settings tab.
4. In the top right corner of the Application Settings pane is the option to edit in Simple or Expert mode. Click Expert.
5. Near the bottom of the list should be a Java disclosure triangle. Click it.
6. This will reveal a Properties disclosure triangle. Click it.

7. Select one of the items already there and click New Sibling.
8. In the Property List column, enter `hwaccelllist`. In the Value column append the appropriate strings from Table 2-1 (page 9).

If you are using MRJAppBuilder to wrap an existing Java application as a Mac OS X executable bundle:

1. Click on the Java Properties tab.
2. Click Add.
3. In the property field of your new property add `com.apple.hwaccellist`.
4. Set the value field set to a string or a comma separated list of strings from Table 2-1 (page 9).

# java.awt.Robot

If you were using `java.awt.Robot` in a previous release of Java on Mac OS X, you were required to turn it on explicitly. The issues that required this workaround have since been resolved, and with this release it is turned on by default. This is important to note only if you previously had code that explicitly turned the Robot on.

# Resolved Issues

The following issues have been addressed between the Mac OS X version 10.1 Java 1.3.1 release and Java 1.3.1 Update 1. If you still have problems with any of the issues listed below, please file a bug. More information on filing bugs can be found in "Filing and Tracking Bugs" (page 5).

A description of this chapter's fields follows:

- Reference supplies you with the bug number that is used by Apple to track the issue internally. Some of these bugs are visible to external developers as well.
- The Problem field states the issue.
- Description gives more details about the issue.
- The Resolution field tells you what Apple has done in these individual cases. Many times this field just says "Software changed." This means that the source code was modified to correct the issue. In some cases, the Resolution field gives you additional information about the fix or may advise you on steps you should take in implementing your code.

## Java AWT

**Reference:** 2774013
**Problem:** AWT problems if the display is asleep.
**Description:** If the display is asleep and the machine is not, AWT may throw exceptions.
**Resolution:** Software changed.

**Reference:** 2780949

**Problem:** Calls to get the screen size do not distinguish between actual screen size and usable screen size.

**Description:** According to the specification, `Toolkit.getScreenSize` returns the size of the screen including the menu bar. `Toolkit.getScreenSize`, does not take Mac OS X's menu bars into account. The same is true for `GraphicsConfiguration`. This can cause problems if you only want the usable screen size.

**Resolution:** This will be handled by JDK1.4.

**Reference:** 2786861

**Problem:** Server-based usage of AWT broken.

**Description:** A server application that uses AWT for image processing will get a dock icon even though no user interface was presented to the user. The application will also be quit when the user logged out.

**Resolution:** The notion of a headless AWT isn't officially supported until Java 1.4, but there is a workaround. If you are launch the Java process from the command line, add `-Dcom.apple.backgroundOnly=true`. If you have packaged the application as a double-clickable application, you can set the Info.plist key `LSBackgroundOnly` to a string value of `1`. These do not prevent the application from being quit when the user logs out, but they will suppress the dock entry and the menu bar.

**Reference:** 2787462

**Problem:** Changing page orientation for printing doesn't always work correctly.

**Description:** Changing the page orientation in the Page Setup dialog results in page formatting information being corrupted.

**Resolution:** Software Changed.

**Reference:** 2789038

**Problem:** The page range in the Print dialog doesn't reflect reality.

**Description:** Java programs that indicate page range do not properly reflect this in the Print dialog. Changes made to the page range in the Print dialog, are not respected by the application.

**Resolution:** Software changed.

# Java Classes

**Reference:** 2610387

**Problem:** The Java application launcher assumes that double-clickable applications should be foreground applications.

**Description:** If you double click the application icon of a background application, one with no user interface, an icon appears in the dock. This is confusing since applications are visible in the Finder that are not accessible through the Finder.

**Resolution:** Setting the `LSBackgroundOnly` key to true in the application's Info.plist now automatically sets the `com.apple.backgroundOnly` system property to `true` as well.

**Reference:** 2771640

**Problem:** Incorrect exception messages in alerts.

**Description:** Exceptions thrown from the `main` method of a double-clickable application display an `java.lang.reflect.InvocationTargetException` in the alert instead of showing the actual exception type and message.

**Resolution:** Software changed.

**Reference:** 2771644

**Problem:** Failure alerts are blank in some applications.

**Description:** In double-clickable Java applications, if the application is incorrectly packaged or is configured such that the `main` cannot be run, the alert only displays a quit button with no descriptive text.

**Resolution:** Software changed.

**Reference:** 2789297

**Problem:** Info.plist Arguments key does not work.

**Description:** If a double-clickable Java application uses the Arguments key of the Info.plist, when the application is launched in the Finder the arguments are not passed to the application's `main(String[])`. If the application is launched from the command line, the arguments are passed to the application's `main(String[])`.

**Resolution:** Software changed.

**Reference:** 2791329
**Problem:** Some Swing Components' print methods do not work.
**Description:** The Swing renderer in AWT does not understand printing.
**Resolution:** Software changed.

**Reference:** 2792935
**Problem:** Java programs using the Java2 Printing API in uncommon ways may encounter printing problems.
**Description:** Calling the Java 2 printing API in non standard ways can result in erratic printing behavior. Calling `setPrintable` (or its related methods) after calling `printDialog`, or using a single `PrintJob` to call `pageDialog` and `printDialog` have resulted in these types of problems.
**Resolution:** Java programs that follow the API examples from Sun do not run into this problem.

**Reference:** 2797762
**Problem:** JNI headers may become outdated with changing JDK versions.
**Description:** With each JDK release, there is the possibility that JavaSoft will update the JNI (and related headers). Mac OS X uses the JavaVM framework to allow binary compatibility between JDK releases with no work required by the developer. However, it may sometimes be necessary to provide more than one version of JDK headers to support old versions of software.
**Resolution:** JDK headers have been moved to Header directories that are versioned by JDK version and not the framework version. The existing paths continue to work by virtue of symbolic links being added. Developers should update their native source code and projects to adapt to the new layout, where necessary.

**Reference:** 2799878
**Problem:** The meaning of Landscape and Reverse Landscape in Java is inverted from the Mac OS X meaning.
**Description:** If a Java program sets the page orientation to `PageFormat.LANDSCAPE` via code and then displays the Page Setup dialog, the orientation displayed is Java's `REVERSE_LANDSCAPE`. The same problem occurs in the inverse case. There is no problem with the resulting page; it is oriented correctly.
**Resolution:** Landscape and Reverse now have the same meaning in Mac OS X as they do in Java.

**Reference:** 2813320

**Problem:** Mac OS X does not support the `-server` option to the Java command line.

**Description:** Other Java implementations provide a `-server` option. The lack of this option can confuse existing scripts that expect it to work.

**Resolution:** A `-server` option is now available with this update. It is important to note that this is not the server version of the JIT compiler. The Java VM is the same Java HotSpot Client VM technology. Using the `-server` flag however, does change a few default settings to values considered more appropriate to a server type program. For example, the default max heap size is set to128MB. These settings may change or be tuned differently in later releases. On Mac OS X Server `-server` will be the default. On Mac OS X, `-client` will be the default.

**Reference:** 2808084

**Problem:** Swing text components do not support drag and drop.

**Description:** Without minor configuration changes, Swing text components are not able to implement drag and drop functionality.

**Resolution:** To provide drag and drop text for a Swing text component, the developer must install the `MacDnDCaret` into the `JTextComponent` (or any text class which extends from `JTextComponent`). This is done as follows:

```
JTextComponent
myTextComponent=...myTextComponent.setCaret(new
com.apple.mrj.swing.MacDnDCaret());
```

**Reference:** 2831422

**Problem:** The Robot does not function properly if display sleep kicks in.

**Description:** If your machine's monitor goes to sleep while the robot is active, the Robot may not function as expected. Specifically, upon display wake-up, the position of the cursor may be returned as (0,0) for the position of all components.

**Resolution:** Software changed.

**Reference:** 2840061

**Problem:** An active Robot does not prevent sleep.

**Description:** The system may still fall asleep if only the Robot is active.

**Resolution:** The robot now more closely emulates user actions. Robot commands (generated events) now reset the various sleep timers the same way as user actions do.

# Java Engine

**Reference:** 2718625
**Problem:** Requesting a large heap crashes the Java VM.
**Description:** `java -Xmx595m -version` results in a crash, rather than the expected behavior.
**Resolution:** This bug has been fixed. The maximum heap limit for Mac OS X Java is now about 980MB (due to other restrictions).

**Reference:** 2785078
**Problem:** Calling `GetDefaultJavaVMInitArgs` with JNI 1.1-style arguments causes a crash.
**Description:** Calling the JNI function `GetDefaultJavaVMInitArgs` with old JNI 1.1-style arguments crashes the process. The process should not crash, but instead print an error message and exit.
**Resolution:** The Java VM now detects this situation and correctly prints an error message and exits.

# Java Graphics

**Reference:** 2756936
**Problem:** The printed page's Graphics2D is clipped incorrectly.
**Description:** When printed, the page is offset and incorrectly clipped for the printer.
**Resolution:** Software changed.

**Reference:** 2789524
**Problem:** AWT crashes accessing ColorSync information.

**Description:** Random crashes in AWT, usually when starting up. AWT shouldn't have more than one thread in Carbon at a time, and ColorSync uses Carbon, so AWT's usage of ColorSync needs to mind the CarbonLock.

**Resolution:** Software changed.

# Java HotSpot

**Reference:** 2781914

**Problem:** Thread Local allocation is not available on Mac OS X.

**Description:** HotSpot has a technology known as Thread Local allocation which allows more scalable allocation for heavily threaded applications. This was not available in Mac OS X 10.0 or 10.1.

**Resolution:** Thread Local allocation (known as Thread Local Eden in HotSpot terminology) can now be activated using a runtime flag: `-XX:+UseTLE`.

**Reference:** 2782713

**Problem:** `java.ext.dir` system property contains non-existent directories.

**Description:** The `java.ext.dir` system property contained directories that were not required to exist on any given Mac OS X system (`/Network/Library/Java/Extensions`, `~/Library/Java/Extensions`). This could cause errors in programs that assumed they would exist.

**Resolution:** The Java VM now checks for the existence of extension directories before adding them to the `java.ext.dir system` property.

**Reference:** 2786528

**Problem:** `String.equals()` could return an incorrect result.

**Description:** With the Java VM shipped in Mac OS X 10.1, `String.equals()` occasionally would return an incorrect result when used with interned String objects.

**Resolution:** This bug was fixed in Mac OS X software update (10.1.1). The fix is also in this update.

**Reference:** 2790571

**Problem:** Java VM would crash when debugging was attempted, but not previously enabled.

**Description:** The `-Xdebug` runtime flag is required to inform the Java VM that it should enable debugging via JVMDI. If this was omitted, and then a JVMDI call was made, the Java VM would crash. It should not; it should return an error.

**Resolution:** This bug was fixed. The Java VM now returns the `EVERSION` error code under these circumstances.

**Reference:** 2792716

**Problem:** Native code (JNI) crashes are hard to track down.

**Description:** If a Java program crashes in native code, unless it is running under `gdb`, it's very hard to get useful information about what caused the crash.

**Resolution:** A new runtime option has been provided to enable Java backtraces to be generated when a crash occurs in native code. This is not guaranteed to work all the time (the crash may destroy some critical data structures) but can be enabled with a runtime flag: `-XX:+PrintJavaStackAtFatalState`.

**Reference:** 2795695

**Problem:** When stressed, the Java VM would occasionally crash with an `Illegal Hardware Instruction`.

**Description:** There was a narrow race condition in dealing with objects returned from native (JNI) functions called from compiled (JIT) code. This could lead to garbage collected objects being referenced inside compiled methods.

**Resolution:** The race condition was eliminated.

**Reference:** 2797340

**Problem:** Looking up the package for a class sometimes resulted in a `null` result.

**Description:** With sharing of system classes enabled (the default), `Package.getPackage(class)` would return `null` for any preloaded classes.

**Resolution:** The sharing technology now stores the package info, so this call now works correctly.

**Reference:** 2800776

**Problem:** Occasional Java VM `internal error` failures.

**Description:** Infrequently, and generally, the Java VM would report an internal error and abort with an error. These were of the form:

```
## HotSpot Virtual Machine Error, Internal Error
```

```
# Please report this error at
# http://bugreport.apple.com/
## Error ID: /SourceCache/HotSpot/HotSpot-47.2/src/os/macosx/
vm/os_macosx.cpp, 1420
## Problematic Thread: prio=1 tid=0x64f6ce0 nid=0x6650cd0
runnable
#Internal ErrorFatal: mach_msg returned unexpected result.
```

**Resolution:** The error handling code, and the above case in particular, has been modified to print more information about this unexpected error.

**Reference:** 2826003

**Problem:** Heavy heap usage can cause excessive system paging.

**Description:** Java applications using large amounts of heap space that are then released, can still trigger system paging because the heap memory is seen to the operating system as "modified".

**Resolution:** When the garbage collection mechanism determines that the heap can be shrunk, i.e., a lot of memory that had been used is no longer needed by the program, the Java VM marks the memory as clean. The operating system is then able to reclaim pages for use by other processes without needing to page out the old memory content to disk. You can turn this behavior off with the runtime flag `-XX:-CleanPagesOnUncommit`.

**Reference:** 2852933

**Problem:** Initializers with deep loops in them may cause a crash.

**Description:** Some initializers contain deep loops. When the number of iterations is sufficiently large, a Trace/BPT trap error is sent to standard output.

**Resolution:** Software changed.

# Java Printing

**Reference:** 2801667

**Problem:** Choosing a custom paper size would print incorrectly.

**Description:** The printing process is incorrectly using the default paper size in lieu of the selected custom size to start printing, and then printing as if it were the chosen paper size, resulting in bad printouts.

**Resolution:** Custom page sizes are now recognized.

**Reference:** 2810728

**Problem:** Printer specific data not reflected by the Java2 Printing API was lost.

**Description:** If a user set some Java `PageFormat` information via the Page Setup dialog, and then that same `PageFormat` was displayed in a Page Setup dialog again, any printer specific data outside of the Java2 Printing definition would be lost. This is data such as the specific printer in a list of printers and printer specific page sizes.

**Resolution:** Software changed.

**Reference:** 2816583

**Problem:** If the user chooses a scale in the Page Setup dialog, it isn't reflected in the Graphics2D given to the Java2 Printing API.

**Description:** If the user chooses a scale in the Page Setup dialog, it should be reflected in the Graphics2D given to the Java2 Printing API. It prints correctly, but a Java developer won't be able to determine that the user has scaled printing.

**Resolution:** Software changed.

**Reference:** 2822980

**Problem:** Printing AWT components doesn't print anything.

**Description:** Trying to print AWT components with the `print` method doesn’t print anything.

**Resolution:** Software changed.

# Java Security

**Reference:** 2782248

**Problem:** Code using `java.security.SecureRandom` would see a noticeable slowdown when getting random bytes.

**Description:** `/dev/random` is not used in Mac OS X as an entropy generating device.

**Resolution:** Since Mac OS X now supports `/dev/random` for entropy generation, this release uses it and its relatives as the entropy generating device for random numbers. This is on by default; you don't have to do anything to your code to use `/dev/random` as the entropy generator. If, however, you have a need to use a method other than `/dev/random` you can specify it with the property `java.security.egd=<URL to entropy generator>`. In some circumstances you may see a 100x speedup in the time it takes to generate a random number.

**Reference:** 2845298

**Problem:** When using an applet on a secure connection (https) a `javax.net.ssl.SSLException: untrusted server cert chain` exception is thrown.

**Description:** This is caused by one of two things; first, the certificate may not be recognized by JSSE as valid. For example, if the certificate is out of date JSSE will not trust the certificate and will throw this exception. If this is the case contact whoever owns the server and ask them to correct the problem. Second, a bug in JSSE prevents some valid certificates from being recognized. Apple and Sun are working on the problem.

**Resolution:** You can export certificates from Internet Explorer for Windows and bring them over to Mac OS X. To do this:

1) Open the Security settings dialog and then go to the certificates window.
2) Drag the certificate you want to the desktop.
3) Transfer it to your Macintosh.
4) In a Terminal window, type: `sudo keytool -import -trustcacerts -keystore /Library/Java/Home/lib/security/cacerts -file /path/to/the/file`. The password for the cacerts file is `changeit`. The certificate is added to the JSSE certificate store. You should now be able to connect to the server in question.

# Java Web Start

**Reference:** 2780830

**Problem:** Java Web Start's cache is hidden in the Finder.

**Description:** Java Web Start's cache is hidden from the user's view because it is in `~/.javaws`, a folder which the Finder interprets as invisible. Running multiple Web Start applications fills this folder with program information that users have no simple way of cleaning out via the Finder. This can result in a perceived loss of disk space.

**Resolution:** The Web Start cache is now stored in `~/Library/Caches/Java Web Start`, which is visible in the Finder. You can easily check the size of the cache by using the Get Info command, command-I, in the Finder).

**Reference:** 2782803

**Problem:** Setting the look and feel in a Web Start application doesn't work.

**Description:** Setting a valid property flag in a JNLP file, for example `<property name="swing.defaultlaf" value="javax.swing.plaf.metal.MetalLookAndFeel"/>`, will be ignored by Swing. The main problem is that Swing has been initialized and the look and feel set long before the JNLP file is read. Other Swing-related properties may be affected by this problem as well.

**Resolution:** This issue is due to the vagueness of the JNLP specification as to when system properties should be set. This is not a Mac OS X specific issue. A bug has been filed with Sun.

# Known Issues

The following are known issues in this release of Java on Mac OS X. They represents only a few of the active bugs that Apple is evaluating for resolution in a later release of Java, but are the bugs that are the most significant to external developers.

## Java AWT

| | |
|---|---|
| **Reference:** | 2515533 |
| **Problem:** | Running applications that have a UI from a remote telnet session should fail gracefully. |
| **Description:** | If you log into a machine and launch a GUI based Java application, you get cryptic error information through the telnet session. |
| **Workaround:** | There currently is none. This issue should be addressed with JDK1.4's support for headless operation. |
| **Reference:** | 2722280 |
| **Problem:** | Servlets launch Java UI elements that, when closed, kill the Tomcat server. |
| **Description:** | When some kinds of servlets are launched, a Java icon appears in the Dock of whoever is logged in. Killing that application kills Tomcat, but the icon remains. This occurs whether or not the request is handed off from Apache via a context mapping. Logging out also crashes Tomcat, putting up a crash log dialog above the login panel. |
| **Workaround:** | There currently is none. This issue should be addressed with JDK1.4's support for headless operation. |

# Java Classes

**Reference:** 2525063

**Problem:** There is no API that lets a Java application determine if a folder is a bundle.

**Description:** Mac OS X has the concept of bundles, where a directory can appear as a single file for certain operations. There is no way for a Java application to determine if the Mac OS considers a directory to be a bundle or just a regular directory.

**Workaround:** There is no simple pure Java workaround, but you could use JNI to access Carbon calls that determine if a folder is a bundle.

**Reference:** 2720352

**Problem:** `MRJOpenDocumentHandler` not called if an application was launched by double-clicking a document in the Finder.

**Description:** Native Mac OS X applications claim certain file types in the system. If a file of that type is double-clicked, the appropriate application will be launched (if not already running) and sent an `AppleEvent` to open that file. Java applications use the `MRJOpenDocumentHandler` to receive that event. Double-clicking on a file that belongs to a Java application will launch the application if it is not running, but the `AppleEvent` will be lost. If the application is already running when document is double-clicked in Finder, the `AppleEvent` will be handled properly by calling `MRJOpenDocumentHandler`.

**Workaround:** There currently is none.

**Reference:** 2845214

**Problem:** The screen saver may consume robot generated events.

**Description:** When the screen saver is running, it may affect the dispatching of robot generated events.

**Workaround:** If you are using robot, you may want to disable the screen saver. In System Preferences choose Screen Saver. Click the Activation tab. Set the slider to Never.

# Java Embedding

**Reference:** 2477415
**Problem:** Need a Cocoa Java embedding model.
**Description:** Cocoa does not support the embedding of pure Java functionality, including applets.
**Workaround:** None at this time.

**Reference:** 2773786
**Problem:** Yahoo! Games may freeze the browser.
**Description:** There are number of Java applet games that can cause the browser to freeze.
**Workaround:** This site is in active development, although games may not have worked at the time Java 1.3.1 Update 1 was released, they may work later as issues are resolved.

**Reference:** 2817019
**Problem:** Cannot play games at http://www.games.com.
**Description:** If you try to load any of the games at http://www.games.com you will get a `javax.net.ssl.SSLException: untrusted server cert chain` exception. The main source of the problem is that the site certificate for the site is out of date.
**Workaround:** Apple is working with Sun for a long term solution that will handle this situation. You can also export certificates from Internet Explorer for Windows and bring them over to Mac OS X. To do this:

1) Open the Security settings dialog and then go to the certificates window.
2) Drag the certificate you want to the desktop.
3) Transfer it to your Macintosh.
4) In a Terminal window, type: `sudo keytool -import -trustcacerts -keystore /Library/Java/Home/lib/security/cacerts -file /path/to/the/file`. The password for the cacerts file is `changeit`. The certificate is added to the JSSE certificate store. You should now be able to connect to the server in question.

# Java Events

| | |
|---|---|
| **Reference:** | 2778552 |
| **Problem:** | Modal dialogs in drop methods may cause a freeze. |
| **Description:** | Displaying a modal dialog in the drop method of a `DragTargetListener` causes the Java application to freeze. It also stops the Dock from being able to launch or switch applications. |
| **Workaround:** | Move the dialog to a separate thread. |
| **Reference:** | 2832782 |
| **Problem:** | No click through with Java on Mac OS X. |
| **Description:** | On other Java platforms you have a click-through behavior. This is especially useful for palette windows. It would be nice to activate this feature via a Java property. |
| **Workaround:** | Currently a user must click twice in Java applications, once to obtain focus and then a second time to initiate the desired action. |

# Java Graphics

| | |
|---|---|
| **Reference:** | 2822976 |
| **Problem:** | Can't print java.awt.List. |
| **Description:** | Images of a java.awt.List may not be printed. |
| **Workaround:** | None. |

# Java Internationalization

| | |
|---|---|
| **Reference:** | 2826318 |
| **Problem:** | Double-byte characters won't be displayed correctly unless the language preference is set. |
| **Description:** | Using logical fonts for screen font or in an editing window, double-byte characters don't display correctly without forcing it to run under corresponding language environment. |
| **Workaround:** | Java applications can not support multiple-script support when running under a Roman system script. |

# Java Security

| | |
|---|---|
| **Reference:** | 2847933 |
| **Problem:** | Internet Explorer hangs when receiving a Java Security Alert. |
| **Description:** | If a web page contains more than one signed applet the VM will deadlock trying to load the two applets. |
| **Workaround:** | Sign no more than one applet per page. |

# Java Sound

| | |
|---|---|
| **Reference:** | 2588625 |
| **Problem:** | Audio input doesn't work. |
| **Description:** | Audio input doesn't work through JavaSound. |
| **Workaround:** | Don’t use JavaSound. |

# Java Toolkit

**Reference:** 2510039

**Problem:** Add `get/setCreationDate` methods to `MRJFileUtils`.

**Description:** There is currently no way to access the creation date of a file without going to native code or JDirect. `MRJFileUtils` already includes a `setModified` method; it should also include `getCreationDate` and `setCreationDate` methods.

**Workaround:** None.

**Reference:** 2585075

**Problem:** `MRJFileUtils.setDefaultFileType`, `setDefaultFileCreator` don't work in Mac OS X.

**Description:** Certain functions present in Mac OS 9's Java implementation no longer work in Mac OS X.

**Workaround:** In Mac OS X, applications behavior should not be dictated by a file type code only. These methods depend on a certain filesystem, HFS(+). Since Mac OS X can run on other file systems, these methods should not be used.

# Java Web Start

**Reference:** 2782802

**Problem:** `-Xdock:name=<name>:icon=<icon>` usage is being deprecated.

**Description:** The use of the multi-option `-Xdock` command-line option is being deprecated so that we can fix a problem that prevents the use of the colon in the application name. If you are using this option in this manner you can split it out into two options for the same effect; i.e., `-Xdock:name=<name> -Xdock:icon=<icon>`.

**Workaround:** None