Java 1.4.2 for Mac OS X v10.4 Release Notes

Java

2005-09-08

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

Simultaneously published in the United States and Canada.

Contents

Introduction to Java 1.4.2 for Mac OS X v10.4 Release Notes

Mac OS X v.10.4 features a release of Java 1.4.2 that fixes issues with Apple's Java implementation. It features a version of Sun's Java 2 Platform, Standard Edition Version 1.4.2_07.

What is Java 1.4.2 for Mac OS X v10.4?

Java 1.4.2 for Mac OS X v10.4 provides numerous improvements to threading, security, and graphics, in addition to other bug fixes. This release includes compatibility with Sun's Java 2 Platform Standard Edition, version 1.4.2_07. For general information about Java changes in Java 1.4.2_07, see *Release Notes - Java 2 SDK, Standard Edition Version 1.4.2_07* at http://java.sun.com/j2se/1.4.2/ReleaseNotes.html.

Who Should Read This Document?

Any developer that currently distributes Java 1.4.2 applications for Mac OS X should read this document since the changes to Java 1.4.2 for Mac OS X v10.4 may affect your application. Anyone interested in new Java development (either J2SE or Cocoa Java) should read this document for the most current information on new features and outstanding issues with Java on Mac OS X.

Organization of This Document

This document contains the following chapters:

- "New Features" (page 7) details significant improvements present in Java for Mac OS X v10.4. This chapter is broken down by category and provides a brief description of the feature and its impact.
- "Resolved Issues" (page 11) highlights a selection of high-visibility bugs that have been addressed in this release. This chapter is broken down by the category where the bug occurs and provides a brief description of what the issue was and how it was resolved.
- "Outstanding Issues" (page 21) presents a selection of high-visibility bugs that you may need to work around with this release. This chapter is broken down by the category where the bug occurs and provides a brief description of what the issue is and often provides a workaround for the issue.

This document also contains a revision history.

If you are just beginning Java development for Mac OS X, you can probably just read the "Outstanding Issues" (page 21) chapter. Otherwise, it is recommended that Java developers read all chapters.

See Also

The Following Apple Java documentation may be helpful:

- *Java Development Guide for Mac OS X*
- *Java 1.4 System Properties*
- *Java 1.4 Virtual Machine Options*
- Previous Java Release Notes
- *Java on Mac OS X Frequently Asked Questions* (http://developer.apple.com/java/faq/)

New Features

This chapter details significant improvements present in Java 1.4.2 for Mac OS X v10.4.

Java AWT

Radar #3753953

Improved AWT Threading.

Description:

Previous versions of Java on Mac OS X used a threading model that varied from standard Java threading.

Resolution:

AWT threading in Java 1.4.2 for Mac OS X v10.4 more closely follows the standard Java threading model. As a result, you may need to examine your Java applications and revise any Mac-specific threading code.

Java Graphics

Radar #3986449

Enabling Quartz 2D Acceleration.

Description:

Quartz 2D acceleration is a feature included in Mac OS X v.10.4 for development purposes. It uses hardware acceleration to speed up rendering of simple primitives like images, lines, rects, and simple characters.

Resolution:

To enable Quartz 2D acceleration for your Java application, follow these steps:

- Turn on Quartz 2D acceleration in Quartz Debug
- Use the `apple.awt.graphics.EnableQ2DX` system property:

```
java -Dapple.awt.graphics.EnableQ2DX=true
```

Note that this is strictly a developer option. Java applications intended for use on Mac OS X v.10.4 should not rely on the presence of Quartz 2D acceleration.

Radar #3987145

Enabling Deferred Drawing Updates.

Description:

Mac OS X v.10.4 uses deferred drawing updates, which eliminates visual tearing, but blocks those applications that flush too often.

Resolution:

Deferred drawing updates are not supported for Java applications. If you want to enable deferred drawing throughout your application, use this system property:

```
java -Dapple.awt.graphics.EnableDeferredUpdates=true
```

Note that this is strictly a developer option. Java applications intended for use on Mac OS X v.10.4 should not rely on deferred drawing updates.

Java Security

Radar #2701070

Improved Cryptographic Services.

Description:

As part of Java, Sun provides a cryptographic service provider. Apple also provides a library of cryptographic algorithms and services through the CDSA architecture.

Resolution:

In Java 1.4.2 for Mac OS X v10.4, Apple now includes a service provider based on the Java Cryptography Architecture. Currently, the following algorithms are supported:

- Mac: MD5, SHA1
- Message Digest: MD5, SHA1
- Secure Random: YarrowPRNG

This release offers an implementation of `KeyStore` that uses the Mac OS X Keychain as its permanent store. You can get an instance of this implementation by using code like this:

```
keyStore = KeyStore.getInstance("KeychainStore", "Apple");
```

See the reference documentation on `java.security.KeyStore` for more usage information.

Radar #3586965

Improved Certificate Integration.

Description:

Previously, Java on Mac OS X used its own certificate trust system.

Resolution:

With Mac OS X v10.4, Java 1.4.2 uses Mac OS X's security framework to handle certificates on behalf of Java applications.

Radar #3658926

Improved Kerberos Support.

Description:

Java 1.4.2 was unable to correctly locate the credentials cache or tickets on the system.

Resolution:

Java 1.4.2 for Mac OS X v.10.4 accesses the credentials cache and tickets correctly, giving Java applications access to the Mac OS X Kerberos system.

Resolved Issues

This chapter lists high-visibility bugs that have been addressed in this release. It is not a complete listing of all of the bugs addressed. If you still have issues with any of these bugs, please file a new bug at http://bugreport.apple.com/ under the `Java (new bugs)` component, Version `X`. Refer to the bug number indicated below in your new bug if you believe it is the same issue.

Java Accessibility

Radar #3795198

Accessible Java applets.

Description:

Java applets weren't separated out into a unique logical grouping within Safari. This caused Safari to crash when trying traverse past an applet using various accessibility features.

Resolution:

Each applet is now a separate `AXGroup`. Safari no longer crashes while trying to traverse past an applet using accessibility features.

Radar #3854210

Accessibility and Java tooltips.

Description:

Java tooltips were treated as accessible objects. This led to a lot of confusion because the information is redundant and differs in behavior from native applications.

Resolution:

Java tooltips no longer respond to accessibility requests.

Radar #3856139

Accessibility and JLists.

Description:

A `JList` was opaque to accessibility requests in Mac OS X. It didn't return any information and it was possible for accessibility to halt trying to traverse past a `JList`.

Resolution:

A `JList` now responds to accessibility requests.

Radar #3938280

Accessibility and text components.

Description:

Java text components didn't respond to accessibility requests. Therefore, text couldn't be accessed word by word or character by character.

Resolution:

Java text components are now accessible at the word and character level.

Java Applets

Radar #3530879

LiveConnect threading issues.

Description:

LiveConnect calls frequently invoked Java methods on a non-Java thread. This caused a number of problems as methods are invoked on the wrong thread and in the wrong context.

Resolution:

LiveConnect calls now invoke Java methods on the correct `EventDispatch` thread and block the non-Java thread while waiting for the method invocation to finish.

Radar #3562703

LiveConnect calls within onload handlers.

Description:

If a LiveConnect call was made in the `onload` handler of a page, the applet would sometimes not be ready to accept the call.

Resolution:

LiveConnect calls are now blocked until the Java applet is loaded and has finished its `init()` method.

Radar #3923005

Some LiveConnect calls cause an exception or crash.

Description:

LiveConnect calls didn't support invocation of static Java methods.

Resolution:

LiveConnect now supports invocation of static Java methods.

Java AWT

Radar #3114971

Peer disposal memory leak.

Description:

In some cases, disposing of windows caused a memory leak due to undisposed peers.

Workaround:

All of a window's peers are now fully disposed.

Radar #3258043

A modal dialog during a modal dialog leaves application modal.

Description:

Showing a modal dialog while another modal dialog is showing or showing two modal dialogs in rapid succession left the application in a state where it appeared as if the application was still modal.

Resolution:

This problem is now fixed.

Radar #3664093

Extra mouse events reported.

Description:

For various controls, some mouse events were passed to a container in addition to the original subcomponent.

Resolution:

The extra events are no longer reported.

Radar #3775638

Unnecessary paint calls in CocoaComponent.

Description:

Sometimes a `CocoaComponent` would prompt Java applications to do their own drawing, resulting in the component being drawn over.

Resolution:

Only the `CocoaComponent`'s `NSView` is prompted to draw.

Radar #3823263

ForceSafeProgrammaticPosition set to true.

Description:

Be default, all Java applications were run with the `ForceSafeProgrammaticPosition` runtime property set to `true`. This caused issues with applications that attempted to run in full screen mode.

Resolution:

The `ForceSafeProgrammaticPosition` option is now set to `false`.

Radar #3863542

AWT and Swing Frame ExtendedState Issues.

Description:

The return value of `getExtendedState` was wrong; also, setting the extended state often wouldn't work. Setting the extended state was done as a constant comparison, rather than a bitmask, which meant that some state would be lost periodically. Changes to state while minimized often didn't work.

Resolution:

State changes are now handled correctly. The state change is now interpreted as a change in potentially multiple states (for instance maximizing a minimized window is now allowed). Hiding and showing of minimized Frames is disallowed.

Minimizing hidden windows takes effect when the window is next shown. Changing the maximized state of a window while minimized takes effect when next unminimized. Note that for `java.awt.Frame`, a maximized window can still be moved or resized. This contrasts the behavior of `javax.swing.JFrame` and the standard implementation of `java.awt.Frame`.

Radar #3902468

Some non-editable TextFields are also non-selectable.

Description:

If a text field was created non-editable, it also became non-selectable.

Resolution:

All text fields are now selectable, regardless of editability.

Radar #3957755

FocusEvents were incorrect.

Description:

`FocusEvents` frequently lacked opposites and were frequently set to be temporary.

Resolution:

`FocusEvents` now have opposites where possible and should accurately be set as temporary or not.

Radar #3971095

Text area default sizing.

Description:

The default size for an AWT text area was too small, causing applications that depended on default sizing to display oddly.

Resolution:

The default size for an AWT text area has been changed to 10 lines high and 60 characters wide.

Java Events

Radar #2846648

Time reporting changes.

Description:

An `InputEvent` stores a timestamp of when it was created. This timestamp should be comparable to the result of `System.currentTimeMillis()`. Instead, it returned the number of milliseconds since system boot.

Resolution:

An `InputEvent` is now stamped with a creation timestamp indicating the number of milliseconds since midnight, January 1, 1970 UTC. Note that this is the time the `InputEvent` itself is created, not the time the underlying native event was created.

Radar #3284378

Using Robot to generate button-2 and button-3 clicks failed.

Description:

The Robot implementation flipped the 2 buttons, so that requesting a button-2 click generated a button-3 click and vice-versa.

Resolution:

The appropriate button event is now generated.

Radar #3872468

Drag-and-drop default action issues.

Description:

During drag-and-drop, the `DropTarget` interprets the default action as `MOVE` even if `COPY` was specified.

Resolution:

The default action as specified is used.

Java Graphics

Radar #3532846

PNG image display issues.

Description:

Under various circumstances, PNG images would not display properly.

Resolution:

Java 1.4.2 for Mac OS X v10.4 offers broader support for PNG images.

Radar #3749983

Exception while exiting full screen mode.

Description:

Sometimes exiting full screen mode caused an exception.

Resolution:

Full screen mode has been fixed so that it no longer causes the exception to be thrown.

Radar #3750520

Component's graphics object empty.

Description:

If you obtained a component's graphics object, the resulting object was empty and any calls on it were ignored.

Resolution:

A real object is now returned. This allows you to paint on top of any component.

Radar #3779947

Overridden paint methods not called.

Description:

In previous versions of Java on Mac OS X, an overridden component's paint method was not called.

Resolution:

Overridden paint methods are now called.

Radar #3824084

TIFF images displayed incorrectly.

Description:

TIFF images may not be displayed correctly in low memory conditions.

Resolution:

TIFF handling has be adjusted to take low memory conditions into account.

Java Aqua Look-and-Feel

Radar #3547670

JPopupMenus and the Aqua look-and-feel.

Description:

In the Aqua look-and-feel, a `JPopupMenu` was completely opaque.

Resolution:

Now all popup menus created in the Aqua look-and-feel use transparency matching that of native popup menus.

Java Networking

Radar #3651645

ServerSocket allows multiple bindings.

Description:

TCP sockets in Java should not be allowed to bind to the same port.

Resolution:

TCP sockets are no longer allowed to bind ports already in use.

Radar #3887227

NetworkInterface and IPv6 addresses.

Description:

Calling `NetworkInterface.getByInetAddress` on an IPv6 address returned `null` instead of the correct `NetworkInterface` object.

Resolution:

`NetworkInterface.getByInetAddress` now properly handles IPv6 addresses.

Java Swing

Radar #3936658

Small windows report incorrect sizes.

Description:

If a decorated Frame is requested to be smaller than the OS-defined minimum size, it is forced to comply with the OS-defined minimum size. Previously, the Frame bounds values weren't updated to match the actual size.

Resolution:

The frame bounds values are now set correctly.

Java Virtual Machine

Radar #3740164

MaxDirectMemorySize is not supported.

Description:

The maximum memory that could be allocated using NIO direct buffers was 64 MB and wasn't variable.

Resolution:

The property `-XX:MaxDirectMemorySize` is now supported and allows variable allocation sizes.

Radar #3783330

Server process crashing during JIT-compilation of Java methods.

Description:

Issues with certain instructions in the JIT Compiler's internal representation were causing a crash during JIT compilation.

Resolution:

JIT Compiler fixed to no longer crash in this scenario.

Radar #3840618

Enhanced JVM crash reports.

Description:

When a Java process crashes, there is a wide variety of crash diagnostics that are generated, but there was no centralized location to find out what data had been generated.

Resolution:

All HotSpot error reports are now reported to the Console. Instructions are printed there on where to find the error report files and how to report a bug:

```
<date> <machine> java: An unexpected Java error has  been detected by HotSpot
 Virtual Machine:
<date> <machine> java: An error report file has been  written to:
<date> <machine> java:   ~/Library/Logs/Java/JavaNativeCrash_pid<num>.crash.log
<date> <machine> java: If this error is reproducible,  please report it with
the  following information:
<date> <machine> java:   1. Provide the steps to reproduce,  a test case, and
any  relevant information
<date> <machine> java:   2. The corresponding JavaNativeCrash_pid<num>.crash.log
  (Java state)
<date> <machine> java:   3. The corresponding <name>.crash.log  (native state
 generated by CrashReporter)
<date> <machine> java: File report at: http://bugreport.apple.com/
<date> <machine> crashdump: java crashed
<date> <machine> crashdump: crash report written to:
~/Library/Logs/CrashReporter/ java.crash.log
```

Radar #3788362

java.io.RandomAccessFile does not allow reopening files in "rws" mode.

Description:

`java.io.RandomAccessFile` did not allow reopening files in "rws" mode if they have just been created. If a file was created with "rw" and then a new `RandomAccessFile` object is instantiated for the same file in "rws" mode, the JVM threw a `FileNotFoundException`.

Resolution:

`FileNotFoundException` no longer thrown when opening a `RandomAccessFile` in "rws" mode.

Outstanding Issues

This chapter lists bugs that you may need to work around in your Java code for Mac OS X. Where possible, workarounds are provided.

Java Accessibility

Radar #3967454

JMenu accessibility issues.

Description:

Menus that are not placed in the screen menu bar are not properly recognized by the accessibility interfaces in Mac OS X.

Workaround:

There are no known workarounds.

Java AWT

Radar #3909714

Setting apple.laf.useScreenMenuBar appears to have no effect.

Description:

Applications that set the property `apple.laf.useScreenMenuBar` during program execution (as opposed to on the command line or application's `Info.plist`) may find that the property appears to be ignored. A change in how the Aqua look-and-feel loads causes the property to be examined earlier than in past releases.

Workaround:

The best way to set this property is either by using `-Dapple.laf.useScreenMenuBar=true` as an argument to the application's command line invocation or by setting it as a property in the Java dictionary of the application's `Info.plist`. Doing so ensures that the property is set before your code runs.

Radar #3839311

Threading changes to CocoaComponent.

Description:

The threading model of the AWT native implementation has been revised. As a result, the AppKit thread can no longer block against the AWT Event Dispatching thread. This results in deadlock.

Workaround:

To provide compatibility between the improved threading model and older implementations of `CocoaComponent`, a compatibility mode has been introduced. This compatibility mode is active the first time a `CocoaComponent` is used in an application and defaults to `true`. A developer can turn this mode off by setting a System Property:

```
com.apple.eawt.CocoaComponent.CompatibilityMode=false
```

While this compatibility mode is active, the AWT does not deadlock for longer than 0.5 seconds. However, this also means that any AWT peer request that takes longer than 0.5 seconds returns asynchronously (with possibly incorrect results for a "get"). The intent is for older applications to work better on Mac OS X v10.4. However, it's strongly recommended that newer applications revise their threading model and disable this compatibility mode. By default, applets always have the compatibility mode enabled. At this time there is no supported way to exclude specific applets from the compatibility mode.

Radar #4205435

Nested Dialogs.

Description:

Nested modal dialogs don't behave properly if they all use a common parent.

Workaround:

Use the previous modal dialog as the current dialog's parent.

Java Embedding Framework

Radar #3669337

The Java Embedding framework is deprecated.

Description:

The Java Embedding framework is deprecated in Mac OS X v10.4 and should no longer be used.

Workaround:

Carbon applications can use the Netscape 4.0 style plugin in `Java Applet.plugin`, found in `/Library/Internet Plug-Ins/`. Cocoa applications should use the `JavaPluginCocoa.bundle` with the Web Kit plugin interface available in Mac OS X v10.4.

Java Developer

Radar #4124800

Java 1.4.2 symbols not visible in Xode Documention window.

Description:

After installing a documentation update in Xcode, Java symbols are no longer present in the Xcode Documentation window.

Workaround:

You need to manually run the `pbhelpindexer` utility in order for Java symbols to appear in the Xcode Documentation window after a documentation update. In Terminal, run this command:

```
sudo /Developer/Tools/pbhelpindexer
```

Java Text

Radar #3909984

Non-antialiased text can't have fractional metrics.

Description:

There is no matching native font-rendering mode for the case where text is not antialiased but fractional metrics are requested. Java on Mac OS X used to be able to set text antialiasing and fractional metrics separately via `CGFontRendering` methods. However, `CGFontRendering` has been deprecated for Mac OS X v10.4.

Workaround:

None. If you want text with fractional metrics, make your text antialiased.

Document Revision History

This table describes the changes to *Java 1.4.2 for Mac OS X v10.4 Release Notes*.

Date	Notes
2005-09-08	Includes information on nested dialogs.
2005-06-04	Added information on using the Java 1.4.2 reference documentation with Xcode.
2005-04-29	First version of Java 1.4.2 for Mac OS X v10.4 Release Notes.

Document Revision History