Java System Property Reference for Mac OS X

Tools & Languages: Other Languages

2009-11-17

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

Simultaneously published in the United States and Canada.

Contents

Java Runtime System Properties 5

Integration with the Native Application Environment 5
Full-Screen Java 6
Quartz Rendering Hints 6
Quartz Graphics Drawing Performance 7
Pixel Conversion Performance 8

Document Revision History 9

Introduction 11

See Also 11
Filing and Tracking Bugs 11

Java Runtime System Properties

The Java runtime system properties described here are supported in J2SE 5.0 and Java SE 6. The properties are grouped according to their functionality. Note that these properties may not all be supported in future versions of Java for Mac OS X.

Most runtime properties accept a Boolean value. If a property accepts a different value, this is noted. It is recommended that you include these property settings in your bundled Java application's `Info.plist` file. For more information on `Info.plist` files for Java applications, read *Java Info.plist Key Reference for Mac OS X*. You can also set system properties from the command line (with the `-D` flag) or in your code (with the `System.setProperty` method). For example, you could set your Swing application to use the Mac OS X menu bar with either

```
java -Dapple.laf.useScreenMenuBar="true" com.example.yourApp
```

or

```
System.setProperty("apple.laf.useScreenMenuBar", "true");
```

When setting a property within your application (using `System.setProperty`), make sure that it is one of the first statements made inside of your `main` method. Doing so sets the property before AWT is loaded, ensuring that it takes effect.

Integration with the Native Application Environment

`apple.laf.useScreenMenuBar`

When using the Aqua look and feel, this property puts Swing menus in the Mac OS X menu bar. Note that JMenuBars in JDialogs are not moved to the Mac OS X menu bar.

The default value is `false`.

`apple.awt.brushMetalLook`

Allows you to display your main windows with the unified Aqua window appearance. This property should be applied only to the primary application window, and should not affect supporting windows like dialogs or preference windows.

The default value is `false`.

`apple.awt.fileDialogForDirectories`

By default, the AWT File Dialog lets you choose a file. Under certain circumstances, however, it may be proper for you to choose a directory instead. If that is the case, set this property to allow for directory selection in a file dialog.

The default value is `false`.

Full-Screen Java

`apple.awt.fakefullscreen`

Causes full-screen applications to be displayed in a window. You might want to use this property during development of full-screen Java applications.

This is strictly a developer option.

The default value is `false`.

`apple.awt.fullscreencapturealldisplays`

When you have multiple displays, entering full-screen mode normally darkens the secondary screens. Setting this property to `false` overrides the default behavior and secondary screens are not darkened. You might want to override the default behavior for development purposes like debugging.

This is strictly a developer option.

The default value is `true`.

`apple.awt.fullscreenhidecursor`

Hides the mouse cursor when in full-screen mode.

The default value is `true`.

`apple.awt.fullscreenusefade`

If you change the screen resolution when entering full-screen mode, the screen transitions by fading out of the old resolution and back in with the new resolution. If you do not change screen resolution, you normally do not see this fade effect. This property enables that fade effect regardless of whether you have changed the screen resolution.

The default value is `false`.

Quartz Rendering Hints

`apple.awt.antialiasing`

Causes graphic primitives such as line, arc, rectangle, and so on, to be painted with antialiasing. Even with this property set to `on` from the command line, you may still set the `KEY_ANTIALISING` rendering hint for specific objects.

This option accepts either `on` or `off` for its value. This property is automatically set to `on` when you use the Aqua look and feel. This makes the behavior more consistent with the native Mac OS X user interface. Note that even if you set this to `off` for an application that uses the Aqua look and feel, Aqua user interface elements themselves are still drawn with antialiasing.

`apple.awt.textantialiasing`

Causes text to be drawn with antialiasing. Sets the `KEY_TEXT_ANTIALIASING` rendering hint. Although this property inherits the same setting as `apple.awt.antialiasing`, you can override that setting explicitly.

This option accepts either `on` or `off` for its value. The default value is `off` unless you are using the Aqua look and feel.

`apple.awt.rendering`

Determines whether Graphics2D objects prioritize speed or quality. This option accepts either `speed` or `quality` for its value. It sets the `KEY_RENDERING` hint so that it accepts either `VALUE_RENDER_SPEED` or `VALUE_RENDER_QUALITY` as an argument.

`apple.awt.interpolation`

Determines which algorithm is used in image transformations, by setting the `KEY_INTERPOLATION` rendering hint. This option accepts either `nearestneighbor`, `bilinear`, or `bicubic` for its value. Setting this option passes `VALUE_INTERPOLATION_NEAREST_NEIGHBOR`, `VALUE_INTERPOLATION_BILINEAR`, or `VALUE_INTERPOLATION_BICUBIC`, respectively, to `KEY_INTERPOLATION`.

`apple.awt.fractionalmetrics`

Sets metrics to use floating-point values instead of using (the default) integer values, by setting the `KEY_FRACTIONALMETRICS` rendering hint. Options include `on` and `off`, corresponding to `VALUE_FRACTIONALMETRICS_ON` and `VALUE_FRACTIONALMETRICS_OFF`, respectively.

Quartz Graphics Drawing Performance

`apple.awt.graphics.OptimizeShapes`

Determines whether graphics primitives are used in place of the relatively more complex shape objects if there is an appropriate mapping. For example, a call to draw a simple shape like `draw(new Rectangle2D.Float(0, 0, 10, 10)` is mapped to `drawRect(0, 0, 10, 10)`. If you do not want this automatic optimization, set this value to `false`.

The default value is `true`.

`apple.awt.graphics.EnableLazyDrawing`

Determines whether graphics primitives are queued before being sent to the renderer. Doing so keeps the renderer primed and improves the graphics performance for rendering simple primitives like lines, rectangles, arcs, and ovals. This is referred to as lazy drawing. You may disable lazy drawing by setting this value to `false`.

The default value is `true`.

`apple.awt.graphics.EnableLazyDrawingQueueSize`

If lazy drawing optimization is enabled, this option sets the size of the queue used. This property takes an integer value which represents the number of graphics primitives cached. Each graphics primitive requires about 10 entries. Each entry requires 4 bytes (1 entry = 4 bytes).

The default value is `2`.

`apple.awt.graphics.EnableQ2DX`

Determines whether hardware acceleration is used to speed up rendering of simple primitives like images, lines, rects, and simple characters. In addition to using this flag, you need to enable Quartz 2D acceleration in the Quartz Debug application, included with the Xcode Developer Tools for Mac OS X.

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

The default value is `false`.

`apple.awt.graphics.EnableDeferredUpdates`

Determines whether drawing updates are deferred. Deferred updates eliminate visual tearing, but block those applications that flush too often. Deferred drawing updates are not supported for Java applications. If you want to enable deferred drawing throughout your application, use this system property.

This is strictly a developer option. Java applications intended for use on Mac OS X should not rely on deferred drawing updates.

The default value is `false`.

`apple.awt.graphics.UseQuartz`

Determines whether Apple's Quartz renderer is used instead of Sun's 2D renderer.

The default value is `true` for J2SE 5.0 and `false` for Java SE 6.

Pixel Conversion Performance

`apple.awt.graphics.EnableLazyPixelConversion`

Determines whether pixel conversion is optimized for image formats that are not natively supported by the underlying operating system. Image formats that are not supported natively by Core Graphics include:

- `TYPE_3BYTE_BGR`
- `TYPE_4BYTE_ABGR`
- `TYPE_4BYTE_ABGR_PRE`
- `TYPE_BYTE_BINARY`
- `TYPE_BYTE_INDEXED`
- `TYPE_CUSTOM`
- `TYPE_INT_ARGB`
- `TYPE_INT_BGR`
- `TYPE_USHORT_565_RGB`
- `TYPE_USHORT_GRAY`

You may override the default behavior by setting this property to `false`.

The default value is `true`.

Document Revision History

This table describes the changes to *Java System Property Reference for Mac OS X*.

Date	Notes
2009-11-17	Updated the default setting information for the apple.awt.graphics.UseQuartz property.
2009-03-04	First version as a separate document. Formerly part of "Java Property, VM Option, and Info.plist Key Reference for Mac OS X".

Introduction

This document describes the system properties that are specific to J2SE 5.0 and Java SE 6 in Mac OS X. It is written for Java developers who want to write Java applications for Mac OS X version 10.5. Although the audience is primarily developers of pure Java applications, it may also be useful for developers using Apple's web application framework WebObjects.

See Also

For more information on Java Development on Mac OS X, read *Java Development Guide for Mac OS X*.

This document and other Java documentation for Mac OS X, including the Javadoc API reference, are available in the Java Reference Library. A subset of this documentation is installed in `/Developer/Documentation/` on a Mac OS X system with the Xcode Tools installed. You can view this documentation through a web browser or through Xcode (from Xcode's Help menu, choose Documentation and then click Java).

General information on previous versions of Java for Mac OS X can be found in the Java Release Notes.

The main Apple website for Java technology, http://developer.apple.com/java/, contains links to information about Java development in Mac OS X.

The `java-dev` mailing list is a great source of information on a wide range of Java development topics in Mac OS X. You can sign up for this list at http://lists.apple.com/mailman/listinfo/java-dev.

Sun's Java web site, http://java.sun.com/ is the essential reference point for Java development in general.

Filing and Tracking Bugs

If you find issues with the implementation of Java that are not covered in this document or you want to follow the resolution of an issue, you may do so online through Radar, Apple's bug tracking system. To access Radar, you 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/. With an ADC membership, you can file and view bugs at http://bugreport.apple.com/. When filing new bugs for Java in Mac OS X, please select `Java` from the Product field and enter `X` in the Version/Build Number field.