Mac OS X Port Project Status

This page outlines the status of high-level feature areas to bring from Apple’s Java SE 6 to the Mac OS X port of JDK 7u4.

7u4 is now complete; 7u6 is in active development in jdk7u-dev.

A list of popular open issues is available at the JIRA bug reporter.

Features in the macosx-port src repository

- Build
  - build using clang
  - buildjdk bundle (for CommandLine and embedded use only)
  - build embeddable jre bundle
    - build JavaAppLauncher stub
- HotSpot
  - remove ALWAYS_PASS_TEST GAMMA hack
  - register new threads as Obj-C-GC capable
  - DTrace v2-style probes
- Core Libraries
  - Load .jnilib (in addition to .dylib) - fixed for 7u4!
  - Precomposed/Decomposed File path translation
  - NIO
    - kqueue based Selectors
    - WatchService implementation (kqueue-based)
    - WatchService implementation (polling-based)
    - Async I/O API
  - Proxy Server handling
  - Locale support
    - Path fixup (vendor name, etc)
  - Resolve "Darwin" vs. "Mac OS X" checks
- JNI interface
- Networking
- X11
- Universal binaries
- JObjC
  - get tests passing
  - self-generate BridgeSupport full files
- Cocoa AWT
  - 2D drawing (OpenGL based)
    - OpenGL layer needs to be vended from an NSOpenGLLayer
  - Cocoa key/text events
    - cleanup to use JRSEvent API
  - merge JDK6 CWIndow -> LWWindow changes and style flags
  - update CImage
  - Transparent window support
  - Headless
    - remove $AWT_TOOLKIT=CToolkit, switch to default
  - threading cleanup
- Aqua LaF
  - Requires an image tile cache
  - Screen Menu Bar
    - Needs cleanup to use JRSMenu API
  - Needs refresh from Apple trunk
- Fonts/Text drawing
  - Needs cleanup to use JRSFont API
- Accessibility
- Clipboard
- Drag & Drop
- InputMethod support
- AppleScript javax.script engine
- Sound
- Printing
- Keychain Provider
- JAWT (for SWT/3D embedding)
- Smartcard
- Fix Kerberos code to read config info from SCDynamicStore (needed on Lion) [http://java.net/jira/browse/MACOSX_PORT-566]
- AWT Desktop API
- System Tray

Features not yet ported from Apple's Java SE 6

- Bundled app launching
Features not in public macosx-port ("deploy" is private to Oracle). You'll have to trust us on these – deploy is not ready to be open-sourced.

- ✔️ Applets/Plugin2
  - ✔️ Need to create .plugin bundle
  - ✔️ Need to embed slimmed JRE into .plugin bundle
  - ✔️ Need NPAPI/Cocoa event -> Java event conversion
  - ✔️ Need the LWAWT to render into NSOpenGLLayer (see above)
  - ✔️ Need cross-process CALayer connection mechanism to be exposed
  - ✔️ Need to determine update mechanism (Sparkle.framework?)
  - ✔️ JavaFX support

- ✔️ Web Start
  - ✔️ Need to determine delivery vehicle (inside Applet plug-in, or some other form) – Java Web Start.app will find javaws in the deployment bundle and execute that. Requires at least 10.7.3.
  - ✔️ Shortcut support

- ❌ Java Preferences
  - ❌ UI cleanup: Look better on Mac OS X
  - ✔️ Preference pane

Non-feature tasks

- ❌ Merge BSDPort into mainline jdk so that Mac OS X Port can be inserted into mainline:
  - ❌ Merge BSDPort hotspot, patch and description at: BSDPort, Description of jdk7 changes

- ❌ Move internal bugs to http://java.net/jira/browse/MACOSX_PORT

- ✔️ Write up development wiki pages for
  - ✔️ Quickstart
  - ✔️ Status
  - ✔️ Compiler changes/issues
  - ✔️ Cocoa & Java crash course
  - ✔️ Threading manifesto
  - ✔️ Memory management manifesto
  - ✔️ AppKit Threading & Cocoa Memory Management best practices examples
  - ✔️ Internal "Adopt-an-app" testing workflows

- ❌ Port private JUnit harness tests to jtreg
  - ✔️ Detailed status

Non-Features

These features will not be ported to OpenJDK from the Apple Java SE 6 port

- ✔️ MacRoman default encoding
  - ✔️ UTF-8 only, for all locales/languages

- ✔️ NSView-based AWT heavyweight widgets
  - ✔️ LWAWT Swing-based widgets only

- ✔️ NSView-based JAWT embedding
  - ✔️ CALayer embedding instead

- ✔️ CocoaComponent
  - ✔️ Use CALayer-based JAWT embedding instead

- ✔️ SWT Carbon EventLoop support
  - ✔️ No 32-bit only technologies, Cocoa only

- ✔️ JNI instantiation through the JavaVM.framework
  - ✔️ Embed your own JRE instead

- ✔️ Apple JavaApplicationStub
  - ✔️ Use the new JavaAppLauncher, loads an embedded .jre, JavaApplicationStub compatible

- ✔️ Apple native crypto
  - ✔️ Use standard Java crypto

- ✔️ HotSpot Shared Archive Generation
  - ✔️ Automatic JSA creation is of limited use until HotSpot supports class sharing in 64-bit, and in all garbage collectors

Status icon meanings

- ✔️ Done
- 🔴 In progress
- ✗ Not done
- ❓ Unknown