Notarisation and the macOS 10.9 SDK

This thread has been locked. Questions are automatically locked after two months of inactivity, or sooner if deemed necessary by a moderator.

The notary service requires that all Mach-O images be linked against the macOS 10.9 SDK or later. This isn't an arbitrary limitation. The hardened runtime, another notarisation requirement, relies on code signing features that were introduced along with macOS 10.9 and it uses the SDK version to check for their presence. Specifically, it checks the SDK version using the `sdk` field in the `LC_BUILD_VERSION` Mach-O load command (or the older `LC_VERSION_MIN_MACOSX` command).

The best way to meet this requirement is to rebuild your code with modern tools. However, in some cases that's not possible. Imagine if your app relies on the closed source `libDodo.dylib` library. That library's vendor went out of business 10 years ago, and so the library hasn't been updated since then. Indeed, the library was linked against the macOS 10.6 SDK. What can you do?

The first thing to do is come up with a medium-term plan for breaking your dependency on `libDodo.dylib`. Relying on an unmaintained library is not something that's sustainable in the long term. The history of the Mac is one of architecture transitions — 68K to PowerPC to Intel, 32- to 64-bit, and so on — and this unmaintained library will make it much harder to deal with the next transition.

But what about the short term? Historically I wasn't able to offer any help on that front, but this has changed recently. Xcode 11 ships with a command-line tool, `vtool`, that can change the `LC_BUILD_VERSION` and `LC_VERSION_MIN_MACOSX` commands in a Mach-O. You can use this to change the `sdk` field of these commands, and thus make your Mach-O image 'compatible' with notarisation and the hardened runtime.

Before doing this, consider these caveats:

- Any given Mach-O image has only a limited amount of space for load commands. When you use `vtool` to set or modify the SDK version, the Mach-O could run out of load command space. The tool will fail cleanly in this case but, if it that happens, this technique simply won't work.
- Changing a Mach-O image's load commands will break the seal on its code signature. If the image is signed, remove the signature before doing that. To do this run `codesign` with the `--remove-signature` argument. You must then re-sign the library as part of your normal development and distribution process.
- Remember that a Mach-O image might contain multiple architectures. All of the tools discussed here have an option to work with a specific architecture (usually `-arch` or `--architecture`). Keep in mind, however, that macOS 10.7 and later do not run on 32-bit Macs, so if your deployment target is 10.7 or later then it's safe to drop any 32-bit code. If you're dealing with a Mach-O image that includes 32-bit Intel code, or indeed PowerPC code, make your life simpler by removing it from the image. Use `lipo` for this; see its man page for details.
- It's possible that changing a Mach-O image's SDK version could break something. Indeed, many system components use the main executable's SDK version as part of their backwards compatibility story. If you change a main executable's SDK version, you might run into hard-to-debug compatibility problems. Test such a change extensively.
- It's also possible, but much less likely, that changing the SDK version of a non-main executable Mach-O image might break something. Again, this is something you should test extensively.

This list of caveats should make it clear that **this is a technique of last resort**. I strongly recommend that you build your code with modern tools, and work with your vendors to ensure that they do the same. Only use this technique as part of a short-term compatibility measure while you implement a proper solution in the medium term.

For more details on `vtool`, read its man page. Also familiarise yourself with `otool`, and specifically the `-l` option which dumps a Mach-O image's load commands. Read its man page for details.

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@apple.com"`

Notarization

Asked 5 months ago by eskimo

Reply to this question