Packaging a Daemon with a Provisioning Profile

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

986

Some APIs can be used from a daemon but require that the client have an entitlement that's allowlisted by a provisioning profile. For example, the EndpointSecurity API can be called from daemon but requires that the daemon have the com.apple.developer.endpoint-security.client entitlement. macOS does not support this directly, but you can get things working by packaging your daemon in an app-like structure. Read this document to learn how to do this with Xcode.

Note If the API you're using supports system extensions, you can avoid all of this rigmarole by packaging your client as a system extension. System extensions support provisioning profiles directly, and Xcode will automatically do the right in that case.

The basic idea here is to create an app target, rather than a command-line tool target, and then remove all of the app-specific stuff and replace it with your daemon code. The following example assumes Xcode 11.4 on macOS 10.15.4, but the technique should work on any version of Xcode or macOS.

To start, create a new app target (File > New > Project > macOS > App). Set the language popup to Objective-C and the User Interface popup to Storyboard.

Note This example uses Objective-C but this approach also works just fine with Swift.

In the General tab of the target editor, make sure that the Bundle Identifier is set correctly. This is important because a provisioning profile is tied to an App ID and the bundle identifier is a key part of that App ID.

Also set the App Icon > Source popup to "Don't use asset catalogs".

Switch to the Signing & Capabilities tab and configure it appropriately. In this example I:

- Removed the App Sandbox capability — The App Sandbox is, as the name suggests, not appropriate for daemons.
- Added the Hardened Runtime capability — You'll need this to notarise your product, so you might as well set it up front.
- Added a Keychain Sharing capability — This requires a provisioning profile, which is critical to this exercise.

In the Project navigator, select the `Info.plist` and then delete all the app-specific items (`NSPrincipalClass`, `NSMainStoryboardFile`, `NSSupportsSuddenTermination` and `NSSupportsAutomaticTermination`).

Also remove the `AppDelegate.{h,m}`, `ViewController.{h,m}`, `Assets.xcassets`, and `Main.storyboard` files.

Replace the code in `main.m` with your daemon code. For this example I used:

```
@import Foundation;

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        SecCodeRef me;
        OSStatus err = SecCodeCopySelf(kSecCSDefaultFlags, &me);
        assert(err == errSecSuccess);
        CFDictionaryRef infoCF;
        err = SecCodeCopySigningInformation(me, kSecCSDefaultFlags, &infoCF);
        assert(err == errSecSuccess);
        NSDictionary * info = CFBridgingRelease(infoCF);
        NSDictionary * entitlements = info[(__bridge NSString *) kSecCodeInfoEntitlementsDict];
        NSLog(@"entitlements: %@", entitlements);
    }
}
```

This logs the current process's entitlements, which is a good way to confirm that things are set up correctly.

Build and run from Xcode. The program prints:

```
2020-02-26 09:21:31.142904+0000 DaemonInAppsClothing[7020:9340817] entitlements: {
    "com.apple.application-identifier" = "SKMME9E2Y8.com.example.apple-samplecode.DaemonInAppsClothing";
    "com.apple.developer.team-identifier" = SKMME9E2Y8;
    "com.apple.security.get-task-allow" = 1;
    "keychain-access-groups" =     (
        "SKMME9E2Y8.com.example.apple-samplecode.DaemonInAppsClothing.shared"
    );
}
```

Now add your daemon's code to the project and call it from `main.m`. You can also replace `main.m` with a plain C or C++ main function if you wish.

The final structure of your daemon should look something like this:

```
% find DaemonInAppsClothing.app
DaemonInAppsClothing.app
DaemonInAppsClothing.app/Contents
DaemonInAppsClothing.app/Contents/_CodeSignature
DaemonInAppsClothing.app/Contents/_CodeSignature/CodeResources
DaemonInAppsClothing.app/Contents/MacOS
DaemonInAppsClothing.app/Contents/MacOS/DaemonInAppsClothing
DaemonInAppsClothing.app/Contents/embedded.provisionprofile
DaemonInAppsClothing.app/Contents/Info.plist
DaemonInAppsClothing.app/Contents/PkgInfo
```

Now modify your `launchd` property list file so that the `Program` property (or the first item of the `ProgramArguments` array) points to the executable within `Contents/MacOS/`.

Finally, I should stress that you don't have to build your daemon using Xcode; this is just an easy way to get started. If you're using another build system, I recommend that you first use these instructions to create an example daemon, and then update your build system based on that.

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

Change history:

- 26 Feb 2020 — Newly written today.
- 9 Apr 2020 — Changed the test app name as aide-memoire. Updated the Xcode and macOS versions.
- 26 Feb 2021 — Fixed the formatting.

XPC

Asked 1 year ago by eskimo

Reply to this question

This site contains user submitted content, comments and opinions and is for informational purposes only. Apple disclaims any and all liability for the acts, omissions and conduct of any third parties in connection with or related to your use of the site. All postings and use of the content on this site are subject to the Apple Developer Forums Participation Agreement.

Developer › Apple Developer Forums

Discover
macOS
iOS
watchOS
tvOS
Safari and Web
Games
Business
Education
WWDC

Design
Human Interface Guidelines
Resources
Videos
Apple Design Awards
Fonts
Accessibility
Internationalization
Accessories

Develop
Xcode
Swift
Swift Playgrounds
TestFlight
Documentation
Videos
Downloads

Distribute
Developer Program
App Store
App Review
Mac Software
Apps for Business
Safari Extensions
Marketing Resources
Trademark Licensing

Support
Articles
Developer Forums
Feedback & Bug Reporting
System Status
Contact Us

Account
Certificates, Identifiers & Profiles
App Store Connect

To view the latest developer news, visit News and Updates.

Copyright © 2021 Apple Inc. All rights reserved. Terms of Use | Privacy Policy | License Agreements