

Exporting a Developer ID Network Extension

This thread has been locked by a moderator.



124

macOS allows you to independently distribute a Network Extension using Developer ID signing, but with an important wrinkle. This post explains that wrinkle, its affect on Xcode, and how you get around it.

If you have questions or comments, put them in a new thread here on DevForums. Tag it with *Network Extension* so that I see it.

Share and Enjoy

Quinn "The Eskimo!" @ Developer Technical Support @ Apple
let myEmail = "eskimo" + "1" + "@" + "apple.com"

Exporting a Developer ID Network Extension

macOS supports a variety of Network Extension (NE) provider types. Starting with macOS 10.15, it's possible to distribute an app containing NE providers independently, using Developer ID signing. See TN3134 [Network Extension provider deployment](#) for the full list of supported provider types.

For your NE provider to work when distributed independently, it must:

- Be packaged as a system extension.
- Use Developer ID specific entitlements

This post is focused on that second point, because it's common source of confusion.

This post assumes that you're building your app with Xcode; if you're building your app outside of Xcode, you'll have to adapt these steps to your build system.

Entitlement Matters

A Network Extension system extension and its container app must be signed with the [Network Extension entitlement](#) (`com.apple.developer.networking.networkextension`). That entitlement is an array, with a variety of different element values based on the provider type. For example, a standard NE content filter provider must include the `content-filter-provider` value.

There are two groups of these values: the standard ones and the ones with the `-systemextension` suffix. During development and for App Store distribution, use the appropriate standard value. For independent distribution using Developer ID, use the corresponding value with the `-systemextension` suffix. For example, a Developer ID signed NE content filter must use `content-filter-provider-systemextension` instead of `content-filter-provider`.

Xcode Issues

Xcode is currently not aware of this requirement. If you build your NE provider container app using Xcode, you might expect to export it for independent distribution using the Direct Distribution workflow in the Xcode organiser. This does not work (r. 108838909).

To get around this, manually export your app from your Xcode archive. Before attempting that, there's a few things to confirm:

- By default Xcode's Signing & Capabilities editor uses the standard values for the NE entitlement. Leave them that way. During day-to-day development it's best to use an Apple Development signing identity [1], and the standard values work with that.
- Continue to use Build > Archive [2] to create an Xcode archive for your product. The steps below replace the Direct Distribution workflow, and they assume you're starting with an Xcode archive.

[1] Don't use Developer ID for day-to-day development; see [The Care and Feeding of Developer ID](#) for more on that topic.

[2] Or, if you're automating this, the `archive` action in `xcodebuild`.

Assemble Your Assets

Imagine you're working on a content filter for the Mac called WaffleFilter. You've used Xcode to build the app into an Xcode archive:

```
% ls "WaffleFilter.xcarchive/Products/Applications"
WaffleFilter.app
```

That app is development signed:

```
% codesign -d -vvv "WaffleFilter.xcarchive/Products/Applications/WaffleFilter.app"
...
Authority=Apple Development: ...
...
```

IMPORTANT The steps in this section are based on the much more comprehensive instructions in [Creating Distribution-Signed Code for Mac](#). If anything is unclear, read that post for clarification.

To re-sign this app for independent distribution you'll need three things:

- A Developer ID application signing identity. This is named `Developer ID Application: TTT`, where `TTT` identifies your team.
- A Developer ID provisioning profile for the app. In this example I've called this `WaffleFilter_Dev_ID.provisionprofile`.
- A Developer ID provisioning profile for the system extension. In this example I've named this `WaffleFilter_WFProvider_DevID.provisionprofile`.

If you're not sure how to create these things, see [Developer Account Help](#).

Re-sign the App

To start, make a copy of the app:

```
% ditto "WaffleFilter.xcarchive/Products/Applications/WaffleFilter.app" "WaffleFilter.app"
```

Dump the entitlements of the app and its embedded system extension:

```
% codesign -d --entitlements "WaffleFilter.entitlements" --xml "WaffleFilter.app"
% codesign -d --entitlements "WaffleFilter_WFProvider.entitlements" --xml "WaffleFilter.app/Contents/Library/SystemExtensions/com.example.apple-samplecode.WaffleFilter.WFProvider.systemextension"
```

And reformat them to make them more readable:

```
% plutil -convert xml1 "WaffleFilter.entitlements"
% plutil -convert xml1 "WaffleFilter_WFProvider.entitlements"
```

Now edit these files to add the `-systemextension` suffix. The result will look something like this:

```
% cat "WaffleFilter.entitlements"
...
<dict>
  ...
  <key>com.apple.developer.networking.networkextension< /key>
  <array>
    <string>content-filter-provider-systemextension< /string>
  </array>
  ...
</dict>
</plist>
% cat "WaffleFilter_WFProvider.entitlements"
...
<dict>
  ...
  <key>com.apple.developer.networking.networkextension< /key>
  <array>
    <string>content-filter-provider-systemextension< /string>
  </array>
  ...
</dict>
</plist>
```

Before you re-sign with these entitlements, replace the embedded provisioning profiles with their Developer ID profiles variants:

```
% cp "WaffleFilter_Dev_ID.provisionprofile" "WaffleFilter.app/Contents/embedded.provisionprofile"
% cp "WaffleFilter_WFProvider_DevID.provisionprofile" "WaffleFilter.app/Contents/Library/SystemExtensions/com.example.apple-samplecode.WaffleFilter.WFProvider.systemextension/Contents/embedded.provisionprofile"
```

Now re-sign the app and the system extension with their new entitlements, from the inside out:

```
% codesign -s "Developer ID Application" -f --entitlements "WaffleFilter_WFProvider.entitlements" --timestamp -o runtime "WaffleFilter.app/Contents/Library/SystemExtensions/com.example.apple-samplecode.WaffleFilter.WFProvider.systemextension"
WaffleFilter.app/Contents/Library/SystemExtensions/com.example.apple-samplecode.WaffleFilter.WFProvider.systemextension: replacing existing signature
% codesign -s "Developer ID Application" -f --entitlements "WaffleFilter.entitlements" --timestamp -o runtime "WaffleFilter.app"
WaffleFilter.app: replacing existing signature
```

If you have multiple Developer ID Application signing identities, you'll need to replace `Developer ID Application` with the name of the specific identity you want to use.

IMPORTANT If your app contains other code items, like frameworks or an app extension, re-sign those as well. For advice on how to manually re-sign a more complex app, see [Creating Distribution-Signed Code for Mac](#).

And you're done!

Manually Notarise

Xcode's Direct Distribution workflow also deals with notarisation. As you're not using that workflow, manually notarise your app. For advice on how to do that, see [Customizing the notarization workflow](#).

You should also look at [Packaging Mac Software for Distribution](#), which has a bunch of general info about packaging Mac apps.

Network ExtensionSystem ExtensionsCode SigningDeveloper ID

Reply

Posted 3 weeks ago by eskimo

Add a Comment

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.

Apple > Developer > Forums

Platforms

iOS
iPadOS
macOS
tvOS
watchOS
visionOS

Tools

Swift
SwiftUI
SF Symbols
Swift Playgrounds
TestFlight
Xcode
Xcode Cloud

Topics & Technologies

Accessibility
Accessories
App Extensions
App Store
Audio & Video
Augmented Reality
Business
Design
Distribution
Education
Fonts
Games
Health & Fitness
In-App Purchase
Localization
Maps & Location
Machine Learning
Security
Safari & Web

Resources

Documentation
Curriculum
Downloads
Forums
Videos

Support
Support Articles
Contact Us
Bug Reporting
System Status

Account
Apple Developer
App Store Connect
Certificates, IDs, & Profiles
Feedback Assistant

Programs

Apple Developer Program
Apple Developer Enterprise Program
App Store Small Business Program
MFI Program
News Partner Program
Video Partner Program
Security Bounty Program
Security Research Device Program

Events

App Accelerators
App Store Awards
Apple Design Awards
Apple Developer Academies
Entrepreneur Camp
Tech Talks
WWDC

To view the latest developer news, visit

[News and Updates](#)

Copyright © 2023 Apple Inc. All rights reserved. Terms of Use Privacy Policy License Agreements