App Groups: macOS vs iOS: Fight!

This thread has been locked by a moderator.

13

I regularly see folks confused by the difference in behaviour of app groups between macOS and iOS. One day I'll have time to write this up for the official docs (r. 92322409) but, in the meantime, here's a quick overview.

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

App Groups: macOS vs iOS: Fight!

The app groups mechanism works differently on macOS and iOS. On iOS:

- Group IDs start with `group.`.
- To use a group ID, allocate it on the Developer website. This associates the group ID with your team.
- To use an app group at runtime, list the group ID in your App Groups entitlement (`com.apple.security.application-groups`).
- Like all entitlements on iOS, that claim must be authorised by a provisioning profile. A profile will only authorise a group ID that's allocated by your team.

For more background on provisioning profiles, see TN3125 Inside Code Signing: Provisioning Profiles.

In contrast, on macOS:

- Group IDs typically start with your Team ID.
- They can't be explicitly allocated on the Developer website.
- Code that isn't sandboxed doesn't need to declare the group ID in the App Groups entitlement.
- To use an app group in a sandboxed app, list the group ID in the App Groups entitlement.
- The App Groups entitlement is not *restricted*, meaning that this claim does not need to be authorised by a provisioning profile.
- The App Store submission process checks that your group IDs make sense.

IMPORTANT In this context I'm using *macOS* to refer to a standard macOS app. In Mac Catalyst things behave as they do on iOS. Likewise for iOS Apps on Mac. Also, anything I say about iOS also applies to tvOS and watchOS.

This difference is a product of the way that each platform protects app group content. On iOS the developer web site enforces group uniqueness, that is, the site prevents team B from using a group ID that's assigned to team A. In contrast, on macOS.

- Group IDs are prefixed with the Team ID solely to prevent collisions.
- The Mac App Store prevents you from publishing an app that uses a group ID that's in use by another team.

Crossing the Streams

[… and mixing my pop culture metaphors!]

In some circumstances you might need to share an app group between iOS and macOS code. For example, you might have a Mac app that needs to share an app group with:

- A Mac Catalyst app
- An iOS app that runs on macOS via iOS Apps on Mac

The solution is to use an iOS-style group ID in your Mac app. This breaks down as follows:

- If your Mac app is not sandboxed, you're free to use any app group ID you like, including one using the `group.` prefix you allocated for your iOS app.
- If your Mac app is sandboxed, that app group ID must be declared in the App Groups entitlement.
- If you submit that app to the Mac App Store, the submission process checks that your app group IDs make sense, that is, they either follow the macOS convention (use a prefix of the Team ID) or the iOS convention (allocate a group ID, with the `group.` prefix, on the Developer website).

App Groups and the Keychain

The differences described above explain an oddity associated with keychain access. Consider this quote from Sharing Access to Keychain Items Among a Collection of Apps:

> **Application groups**
> When you collect related apps into an application group using the App Groups entitlement, they share access to a group container, and gain the ability to message each other in certain ways. Starting in iOS 8, the array of strings given by this entitlement also extends the list of keychain access groups.

There are three things to note here:

- Using a group ID as a keychain access group only works on iOS; it's not supported on macOS because doing so would be insecure.
- The App Groups entitlement must be allowlisted by a provisioning profile on iOS, and that process is what protects the keychain from unauthorised access.
- The required `group` prefix means that these keychain access groups can't collide with other keychain access groups, which all start with an App ID prefix (there's also Apple-only keychain access groups that start with other prefixes, like `apple`).

In contrast, standard keychain access groups are protected the same way on both platforms, using the Keychain Access Groups entitlement, `keychain-access-groups`.

Not Entirely Unsatisfactory

When you launch a Mac app that uses app groups you might see this log entry:

```
type: error
time: 10:41:35.858009+0000
process: taskgated-helper
subsystem: com.apple.ManagedClient
category: ProvisioningProfiles
message: com.example.apple-samplecode.Test92322409: Unsatisfied entitlements: com.apple.security.application-groups
```

This is kinda worrying, and many folks interpret it to mean that there's something wrong with their code signing. In reality, it's [log noise][refOLN]. It's telling you that:

- Your app has claimed the App Groups entitlements.
- That entitlement is not authorised by your provisioning profile.

On iOS that would be a big deal; indeed, the trusted execution system would block your app from launching in this case. However, on macOS it's absolutely normal.

Note The exact format of that log entry, and the circumstances under which it's generated, varies by platform. On macOS 13.0.1 I was able to generate it by running a sandboxed app that claims the App Group entitlement and also claims some other restricted entitlement.

Entitlements Code Signing

Reply

Posted 8 hours ago by eskimo

Add a Comment