Article

Creating an App Clip with Xcode

Add an App Clip target to your Xcode project and share code between the App Clip and its corresponding full app.

Overview

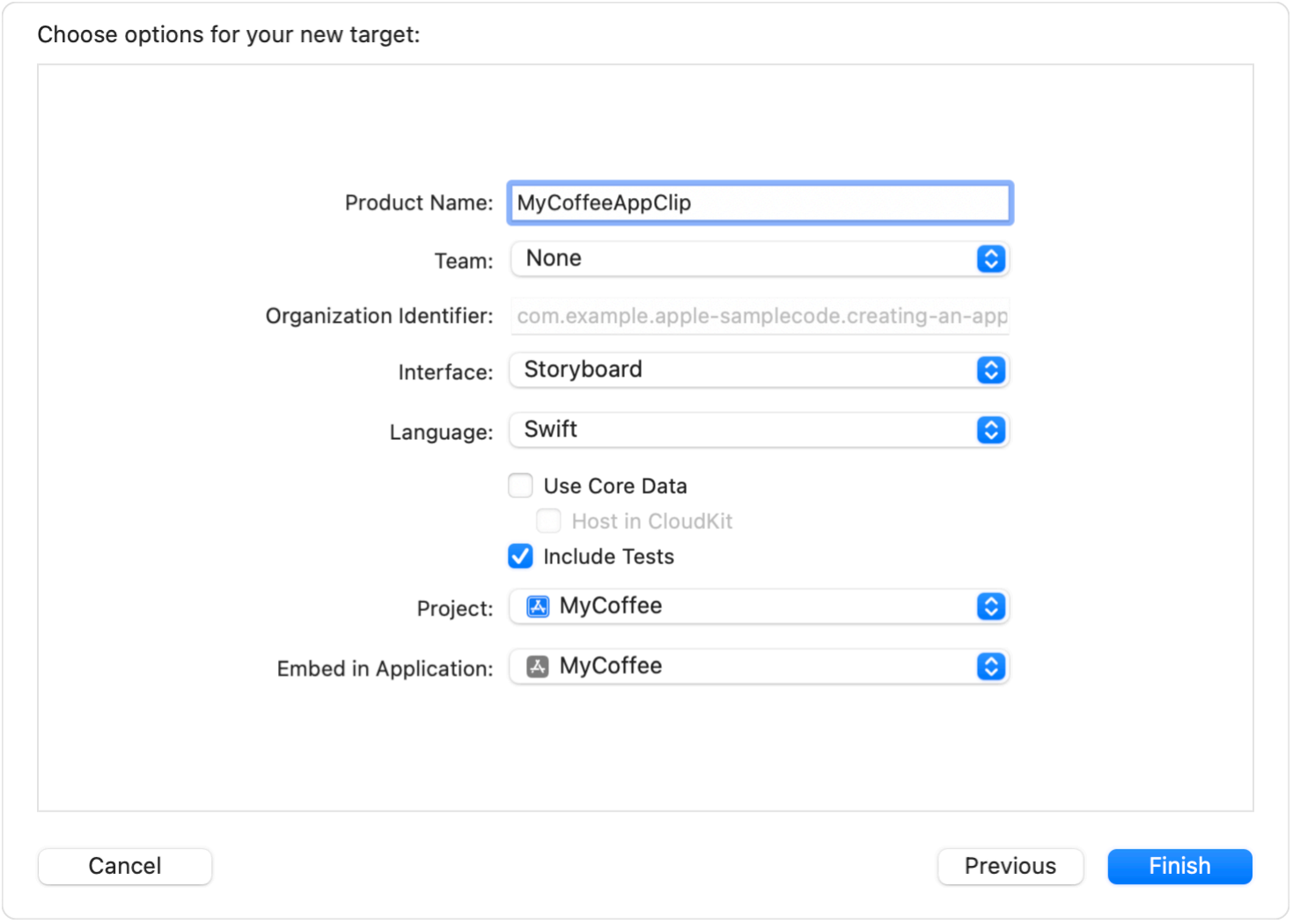
An *App Clip* is a lightweight version of your app that offers some of its functionality when and where people need it or that people use to try out your full app. With Xcode, you can add an App Clip target to your app's project, share code and assets between the App Clip and the full app, and build, run, and debug your App Clip.

Add an App Clip target

App Clips require a corresponding full app that offers at least the same functionality as the App Clip; you use the same Xcode project for your full app and your App Clip. If you're starting a new app project, first create a new iOS project with Xcode. If you want to add an App Clip to your existing iOS app, open its Xcode project. Then, add an App Clip target to the Xcode project:

1. Add a new target using the App Clip template.

2. Choose a product name, select applicable options for your App Clip, and click Finish.



Xcode creates all required files for the options you choose and adds a target for your App Clip with:

- A scheme to build and run your App Clip and its tests
- A new capability named On Demand Install Capable that adds the `com.apple.developer.on-demand-install-capable` entitlement
- The `Parent Application Identifiers Entitlement`
- An app identifier for the App Clip, using the full app's app identifier as its prefix, followed by a string. For example, if your full app's app identifier is `$(AppIdentifierPrefix)com.example.MyApp`, the app identifier for your App Clip would be `$(AppIdentifierPrefix)com.example.MyApp.Clip`
- The `_XCAppClipURL` environment variable for the scheme of your App Clip that allows you to debug invocations
- Support for the same devices as the full app, not including macOS

Additionally, Xcode creates a new build phase for the app target that embeds the App Clip in the app.

Before you add code to the App Clip target, run the App Clip in Simulator or on a device. At this point, the App Clip shows an empty white screen because you haven't yet added any code and

assets to the App Clip target.

> **Note**
>
> When you archive the app that comes with an App Clip, Xcode adds the `com.apple.developer.associated-appclip-app-identifiers` entitlement to your app. Together with the `Parent Application Identifiers Entitlement`, it associates your App Clip with your app.

Add code and assets

App Clips make use of the same frameworks as full apps, and adding code or assets to an App Clip target works just like it does for any other target. Create new source files and assets, or use existing source files and assets, and add them as members to the App Clip target. To ensure the project's maintainability, both the full app and the App Clip should share as much code as possible:

- If you create a new app, build it with creating an App Clip in mind, and follow best practices that promote a modular code base. For example, create reusable components, bundle them as Swift packages, and use the packages in both the full app and the App Clip. For more information, see Organizing your code with local packages.
- If you add an App Clip to an existing app, set aside time to refactor the app's code base to be modular and share code between the App Clip and the full app to avoid duplicating code.
- Add shared assets to a new asset catalog, and use the catalog in both the full app and the App Clip. For more information about asset catalogs, see Managing assets with asset catalogs.

Verify the size of your App Clip

Your App Clips must be small to launch instantly. Aim to keep your App Clip well below the applicable limits outlined in Choosing the right functionality for your App Clip.

To measure the size of your App Clip, create an app-size report for your App Clip:

1. In Xcode, archive the app that belongs to your App Clip, open the Organizer window, select the archive, and click Distribute App.
2. Export the App Clip as an Ad Hoc or Development build with App Thinning enabled.

The output folder for your exported App Clip contains its size report, a file named `App Thinning Size Report.txt`. Open the text file, note the uncompressed size of your App Clip for each variant, and then make adjustments to your project to keep the uncompressed size for each variant below the applicable size limit.

For more information on measuring your app's size, see Reducing your app's size.

Download additional assets

App Clips can use Background Assets to download additional content. If your App Clip offers an in-the-moment experience, ensure instant availability by keeping the App Clip as small as possible and avoiding usage of Background Assets.

If you create an App Clip to offer a demo version of your app or your game, and are confident that people have a reliable and fast network connection when they invoke your App Clip, use Background Assets to down additional content. For example, the App Clip demo of a game might include assets needed for people to start the demo and create their in-game hero. To keep the App Clip small, it might not include the assets needed to play the first three levels of the game but downloads them while people create their hero.

To download additional assets in the background, make sure you configure Background Assets for your app and App Clip targets. Note that your App Clip can't set a background asset download's priority to essential with `isEssential`.

Use active compilation conditions

Adding an App Clip to your app is a good opportunity to refactor your app's code to be modular and reusable. Most functionality and frameworks available to your full app are available to your App Clip. However, you may encounter cases where you can't use some of your app's code in the App Clip, and creating separate modules for your app and App Clip code isn't feasible. In these cases, take advantage of the Active Compilation Conditions build setting, where you can declare a condition to exclude code.

Start by navigating to your App Clip target's build settings and creating a new value for the Active Compilation Condition build setting (for example, `APPCLIP`). Then, add a check in your shared code where needed, to exclude code you don't want to use in your App Clip.

The following code checks for the `APPCLIP` value you added to the Active Compilation Conditions build setting:

```
#if !APPCLIP
// Code you don't want to use in your App Clip.
#else
// Code your App Clip may access.
#endif
```

Review next steps

Adding an App Clip target to your app's Xcode project and modifying the project are only the first steps in offering an App Clip. Next, plan to spend time designing the launch experience of your App Clip by:

- Reviewing how invocations work
- Identifying invocations you want to support
- Planning which URLs launch your App Clip
- Changing your code to respond to invocations

Based on the decisions you make, you'll use App Store Connect to:

- Configure the required default App Clip experience
- Use the default App Clip link or the App Clip demo link
- Configure optional advanced App Clip experience
- Add code to respond to different invocation URLs
- Create App Clip Codes

To learn more about the App Clip launch experience for your App Clip, see Configuring App Clip experiences and Responding to invocations.

You may also have to associate your App Clip with your website and make changes to your server. For more information, refer to Associating your App Clip with your website.

When it's time to test your App Clip, use Xcode to test the launch experience locally or test it with TestFlight. For more information, see Testing the launch experience of your App Clip.

See Also

Creation

{} Fruta: Building a feature-rich app with SwiftUI

Create a shared codebase to build a multiplatform app that offers widgets and an App Clip.

`Parent Application Identifiers Entitlement`

A list of parent application identifiers for an App Clip with exactly one entry.

`com.apple.developer.associated-appclip-app-identifiers`

A list of App Clip identifiers for an app with exactly one entry.

`com.apple.developer.on-demand-install-capable`

A Boolean value that indicates whether a bundle represents an App Clip.