Article

Creating an Intents App Extension

Add and configure an Intents app extension in your Xcode project.

Overview

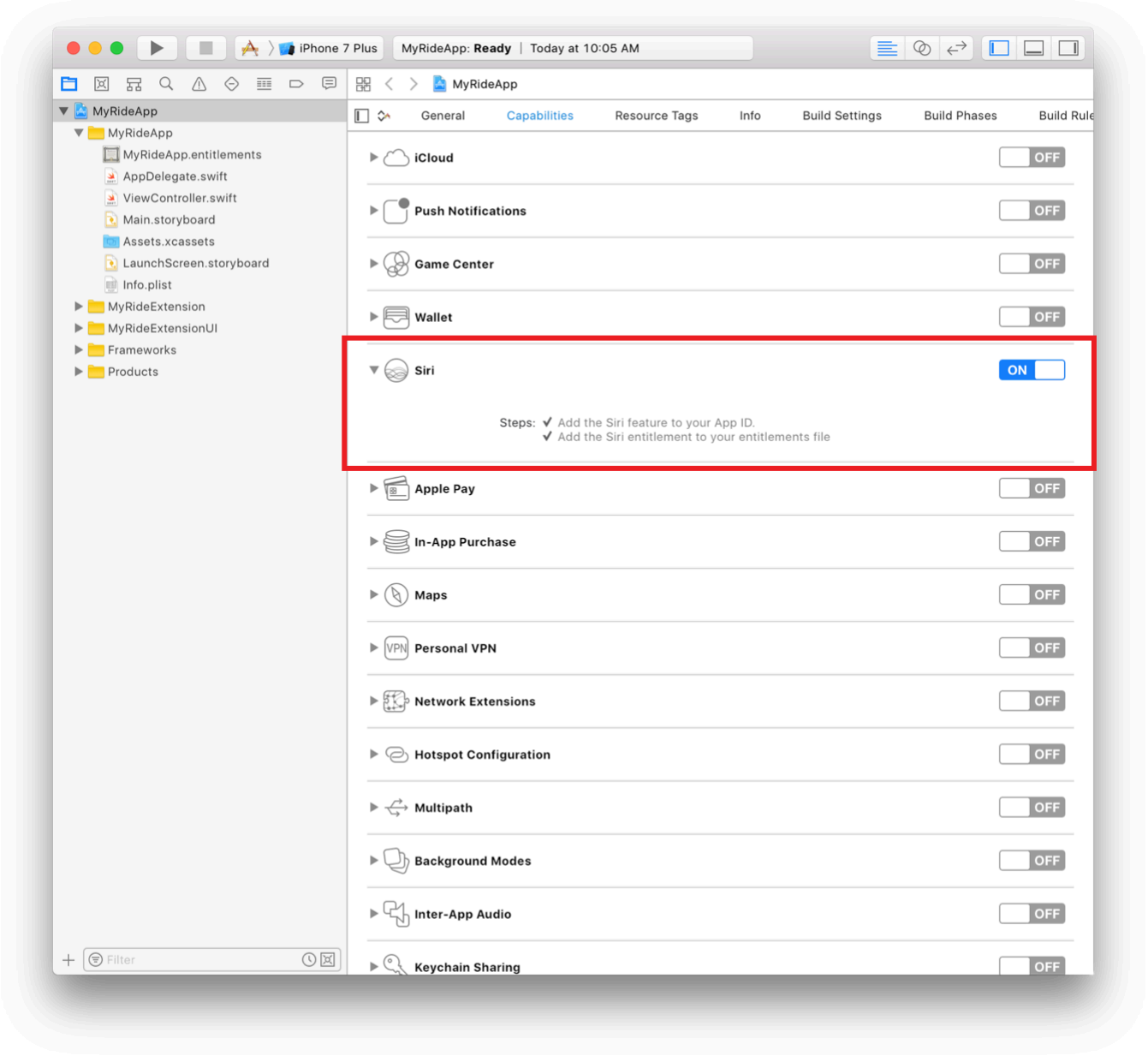
Interactions with SiriKit occur through your Intents app extension, which you deliver inside your iOS or watchOS app bundle. The Intents app extension handles most interactions with SiriKit. However, your app needs to be aware of those interactions and in some cases may have an active role in making them happen.

Configuring your Xcode project requires several steps. In addition to adding an Intents app extension target to your project, you must make some minor changes to your app as well.

Enable the Siri Capability

Enabling the Siri capability adds a set of entitlements to your app. The App Store requires the presence of these entitlements for any iOS app or watchOS app containing an Intents extension.

1. Open your app project in Xcode.
2. In the project settings, select the appropriate target. For iOS, select your iOS app target. For watchOS, select your WatchKit Extension target.
3. Select the Capabilities tab.
4. Enable the Siri capability.



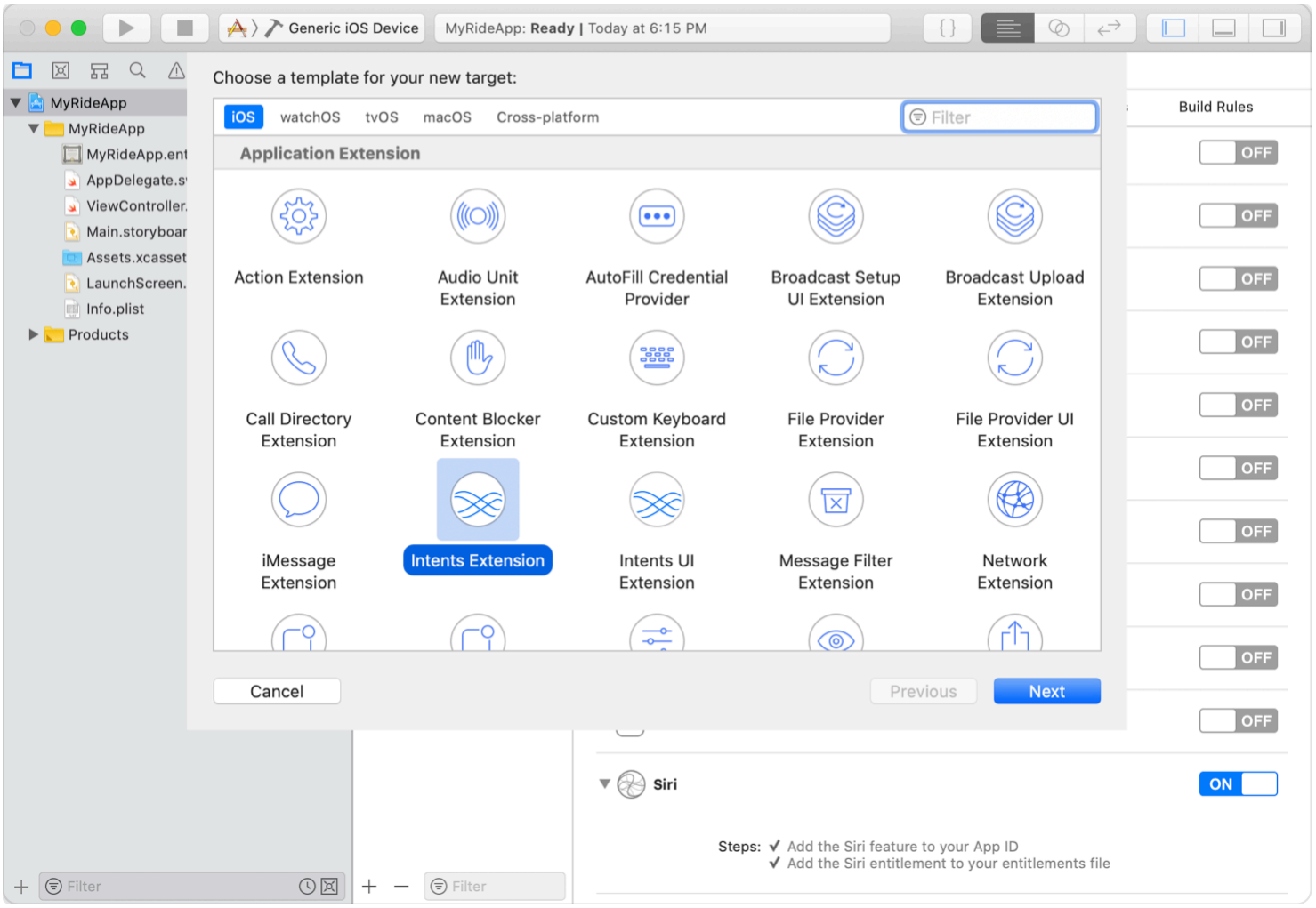
Add an Intents App Extension to Your Project

Adding an Intents app extension target provides the initial files you need to build your Intents extension and configures your Xcode project to build that extension and include it in your app's bundle.

1. Open your existing app project in Xcode.
2. Choose File > New > Target.
3. Select Intents Extension from the Application Extension group of the iOS or watchOS platform.
4. Click Next.
5. Specify the name of your extension and configure the language and other options.
6. For an iOS app, enable the Include UI Extension option if you plan to customize portions of the Siri interface.

7. Click Finish.

> **Note**
>
> You may add more than one Intents extension to your app, but each extension must support different intents. Create multiple extensions only if doing so provides a performance advantage or reduces the memory footprint of each extension.



Specify the Intents Your Extension Supports

After adding your Intents extension target to your project, specify the intents the extension supports. SiriKit uses the information in your extension's `Info.plist` file to determine which intents to route to your extension.

1. In Xcode's Project editor, select the extension target and expand the Supported Intents section in the General pane.
2. Add the class name of each intent you support.
3. If you support the `INPlayMediaIntent`, also select each type of media your app plays in the Media Categories list. For more details about these categories, see `SupportedMediaCategories`.

4. For each intent you support, select the appropriate Authentication requirement.

Choose Restricted While Locked to specify that the intent requires an unlocked device. Some intents, such as those involving financial transactions, always require the user's device to be unlocked. For those intents, Siri automatically asks the user to unlock the device, even if you leave the Authentication set to None. In watchOS, all intents require the device to be unlocked.

> **Important**
>
> Unless your watchOS app has `WKRunsIndependentlyOfCompanionApp` enabled, your iOS app must support all of the intents your watchOS app supports, either directly in the app or in an app extension.

When an ambiguous user utterance resolves to multiple intents, SiriKit uses the order of the intents in the Supported Intents table to determine which intent to send to your app. Organize your list of intents by putting the most relevant ones at the top of the table. Prioritizing your intents is especially important when your Intents extension supports multiple domains with similar semantics. For example, an app that supports calling and messaging intents might choose to prioritize sending a message over initiating a call.

Some intents may require additional configuration steps for your Xcode project or your app. For example, when implementing a ride-booking app, Maps expects you to provide a GeoJSON file that describes the coverage area for your service. See the reference documentation for information about any special requirements.

Test Your Intents App Extension

Xcode supports launching your Intents app extension directly from your Xcode project and debugging the extension while it runs in the simulator or on a device. To run and debug your Intents extension:

1. Select the build scheme for your Intents extension. Xcode automatically creates a build scheme when you create an Intents extension target.
2. Select the target (simulator or device) on which to run your code.
3. Select Product > Run to begin your debugging session.
4. When prompted by Xcode, select Siri or Maps as the main app to run. Xcode builds your app and extension, installs them on the devices, and launches the app you selected.

When installing your extension for the first time, Siri may not immediately recognize your app extension and you may need to wait several minutes before you can issue any relevant commands. Similarly, when updating your extension's `Info.plist` file, you may need to wait several minutes before Siri recognizes any changes.

See Also

Articles

- Adding User Interactivity with Siri Shortcuts and the Shortcuts App

  Add custom intents and parameters to help users interact more quickly and effectively with Siri and the Shortcuts app.

- Defining Relevant Shortcuts for the Siri Watch Face

  Inform Siri when your app's shortcuts may be useful to the user.

- Deleting Donated Shortcuts

  Remove your donations from Siri.

- Dispatching intents to handlers

  Provide SiriKit with an intent handler capable of handling a specific intent.

- Improving Siri Media Interactions and App Selection

  Fine-tune voice controls and improve Siri Suggestions by sharing app capabilities, customized names, and listening habits with the system.

- Improving interactions between Siri and your messaging app

  Donate app-specific content, use Siri's contact suggestions, and adopt the latest platform features to create a more consistent messaging experience.

- Registering Custom Vocabulary with SiriKit

  Register your app's custom terminology, and provide sample phrases for how to use your app with Siri.

- Confirming the Details of an Intent

  Perform final validation of the intent parameters and verify that your services are ready to fulfill the intent.

- Handling an Intent

  Fulfill the intent and provide feedback to SiriKit about what you did.

- Resolving the Parameters of an Intent

  Validate the parameters of an intent and make sure that you have the information you need to continue.

- Generating a List of Ride Options

  Generate ride options for Maps to display to the user.

- Handling the Ride-Booking Intents

  Support the different intent-handling sequences for booking rides with Shortcuts or Maps.

- Donating Reservations

  Inform Siri of reservations made from your app.

- Specifying Synonyms for Your App Name

  Provide alternative names for your app that are more familiar or easier for users to speak.

- Intent Phrases

  The keys that you include in your global vocabulary file to show how users engage your app from Siri.