Article

Configuring Your Intents UI App Extension Target

Configure your Xcode project to include an Intents UI app extension that you use to customize the Siri and Maps interfaces.

Overview

An app may have one or more Intents UI app extensions to manage the custom interfaces for its intents. Each extension has a single view controller class that provides the interface for the supported intents. To set up each Intents UI app extension, you must add a custom target to your Xcode project and configure its `Info.plist` file with the intents that it supports.

Add an Intents UI App Extension to Your Xcode Project

When creating your Intents app extension, you can ask Xcode to create an Intents UI extension at the same time. If you didn't do that, you can add an Intents UI extension to your project later:

1. Open your iOS app project in Xcode.
2. Choose File > New > Target.
3. Select Intents UI extension from the iOS Application Extension group.
4. Click Next.
5. Specify the name of your extension and configure the language and other options.
6. Click Finish.

The Xcode template includes a storyboard and a single view controller. Siri and Maps always load and display the initial view controller in your extension's storyboard file, so configure that view controller with your content. You're responsible for managing all interactions between the initial view controller and the rest of your content. Typically, you add custom views to the initial view controller scene, but you might also embed child view controllers so that you can display different content for each of your intents.

Specify the Intents that Your Extension Supports

After adding your Intents UI app extension target to your project, configure the default `Info.plist` file provided by Xcode to specify which intents should use your custom interface. SiriKit uses the information in your extension's `Info.plist` file to determine when to load your custom view controller.

1. In Xcode, select the `Info.plist` file of your intents app extension.

2. Expand the `NSExtension` and `NSExtensionAttributes` keys.

3. In the `IntentsSupported` key, add a String item for each intent that your extension handles. Set the value of each item to the class name of the intent.

The `NSExtensionMainStoryboard` key in your `Info.plist` file contains the name of the storyboard containing your view controller. If you change the name of your storyboard file, remember to update the value of this key. If you prefer to create your view controller programmatically, replace this key with the `NSExtensionPrincipalClass` key, and set the value to the name of your view controller class.

Important

The `NSExtensionMainStoryboard` and `NSExtensionPrincipalClass` keys are mutually exclusive. If both keys appear in the extension's `Info.plist` file, the system won't load the extension.

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.