Article

Donating Shortcuts

Tell Siri about shortcuts to actions that the user performed in your app.

Overview

Siri can predict shortcuts to actions that a user may want to perform using your app, and suggest those shortcuts to the user in places such as Spotlight search and the Lock Screen. Siri learns about the shortcuts available for your app through donations that your app makes to Siri. Users can also use donated shortcuts to add personalized voice phrases to Siri. To learn more about adding phrases to Siri, see Shortcut-Related UI.

Donate Shortcuts at the Right Time

Donate a shortcut each time the user performs the action in your app. Make one, and only one, donation per action at the time the user performs the action. If the user performs the same action again, make another donation. For example, if the user can order soup from a restaurant using your app, donate a shortcut for the *order soup* action after the user places their order. If the user places another order, then make another *order soup* donation.

However, don't make more than one shortcut donation for an action at the time the user performs the action. Additionally, don't make donations for actions the user hasn't completed in your app; if the user never places an order for soup, don't donate a shortcut for the *order soup* action.

Your app can make donations using one of the following objects:

- `NSUserActivity`. Donate the shortcut using a user activity when the action involves a view within your app, such as displaying recent transactions in a banking app.
- `INInteraction`. Donate the shortcut using an interaction when the action involves a task the user accomplishes with your app, such as recording activity in a ski and snowboard tracking app.

Important

In iOS 17 and later, the Journal app encourages people to reflect and write about their day-to-day experiences. If your app donates activities or interactions to SiriKit, the subject matter of the interaction may appear as a suggestion in the Journal app, or other apps that utilize the Journaling Suggestions framework.

Donate a User Activity

`NSUserActivity` provides a lightweight approach for making a donation that also integrates with other Apple features such as Handoff and Spotlight search.

To make a donation using `NSUserActivity`, define the activity as a type in the `NSUserActivityTypes` array in your `Info.plist`. The activity type should be a reverse domain name that's unique within the list.

In your app, create an instance of `NSUserActivity` and set its `title`, `userInfo`, and `requiredUserInfoKeys` with information that your app needs to resume the activity at a later time. Also set the `isEligibleForPrediction` property to `true` and the `persistentIdentifier` to a unique string value, which you need in order to delete the donation (see Deleting Donated Shortcuts). You can also suggest the voice phrase that a user may want to use when adding a phrase to Siri by setting the `suggestedInvocationPhrase` property on the user activity.

Next, call the `becomeCurrent()` method on the user activity object to mark it as current, which donates the activity to Siri. Alternatively, you can attach the object to a `UIViewController` or `UIResponder` object, which also marks the activity as current.

To handle the action at a later time, implement the `application(_:continue:restorationHandler:)` method in your app delegate.

Donate an Interaction

You can also make a donation using an `INInteraction` object. For this type of donation, you provide the interaction with an intent that represents the action (or task) that the user wants to accomplish using your app. This type of donation has an advantage over ones made with a user activity in that Siri can deconstruct the donation into its intent parameters allowing for more intelligent predictions.

Before diving into the source code, look through the list of system-provided intents to see if one exists that's suitable for your action (see Standard Intents). Use an intent that best describes the action; otherwise, create a custom intent.

To donate the intent, create an instance of the intent class. Set its parameter values and add images to the parameters as needed. To recommend a voice phrase that the user may want to add to Siri, set the intent's `suggestedInvocationPhrase` property to a string containing the phrase.

> **Important**
>
> Siri reads the Intent Definition file to determine the intent and parameter combinations that an app supports for prediction. In order to help Siri make the best possible suggestions, add all the system-provided and custom intents your app supports and their parameter combinations to your Intent Definition file.

After creating the intent, create an instance of `INInteraction` and call its `donate(completion:)` method, passing in the intent. This tells Siri about the shortcut.

To handle the shortcut, implement the `application(_:continue:restorationHandler:)` method in your app delegate. When the time comes to handle the action, the system opens your app. To provide a better user experience—one where opening your app each time to handle the shortcut isn't necessary—provide an Intents App Extension with your app. The extension handles the shortcut in the background without needing to open your app, providing a richer experience that occurs within Siri. For more information on Intents App Extensions, see Creating an Intents App Extension.

Even if you provide an Intents App Extension, you should always implement the `application(_:continue:restorationHandler:)` method in your app delegate. Certain user activities—such as tapping a shortcut in Siri—open your app, and the user expects the app to handle that shortcut. Your app is also responsible for handling shortcuts that can't run in the background; for example, when your apps needs to ask the user for additional information before completing the action.

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.