Registering Custom Vocabulary with SiriKit

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

Overview

Apps must register any vocabulary that's used in a nonconventional way or that's completely unique to your app. Siri uses your custom vocabulary to improve the speech recognition process and to ensure that your app has correct information.

You must provide one or more example phrases for engaging your app through Siri. Example phrases correspond to the way that users speak about your app and should include any custom vocabulary terms defined by your app.

The terms you define from your iOS app are automatically shared with your Intents extension on watchOS. You don't need to register your user-specific or global vocabulary again from your Intents extension in watchOS.

Register Example Phrases and Global Vocabulary

Register your app's example phrases and global vocabulary using the Info.plist file of your Intents app extension. All apps should provide at least a few phrases that teach users how to use Siri to trigger your intents. Siri displays your phrases in the Siri Guide and also uses custom vocabulary to improve the accuracy of its recognition process. To create a global vocabulary file:

1. In your Xcode project, select New > File.
2. In iOS > Resource, select the Property List file type.
3. Click Next.
4. Set the name of the file to `AppIntentVocabulary.plist`, and add the file to your iOS app.

5. Click Create.

Place your `AppIntentVocabulary.plist` file in the language-specific (`.lproj`) directory of your iOS app that corresponds to your base development language. Include localized versions of the file in each of your app's language-specific project directories. In Xcode, you can create localized versions automatically from the File inspector.

The global vocabulary file contains two keys at the root level:

- The `IntentPhrases` key contains example phrases for invoking your services. Always include this key.
- The `ParameterVocabularies` key defines your app's custom terms that apply to all users of your app and the intent parameters to which they apply. You may omit this key if you don't have any custom vocabulary terms.

You may specify custom vocabulary terms for *ride options* and *workout names*. You must always associate custom vocabulary terms with a specific property of the corresponding intent object. You may associate the same term with multiple intents. For example, you may associate the same workout name with all of the workout-related intents.

> **Important**
>
> During development, Xcode forwards your global vocabulary to Siri but limits the availability of that vocabulary to your development device. Ingestion of your vocabulary data isn't instantaneous, so you may need to wait a few minutes before testing any custom vocabulary.

For information about how to fully specify example phrases and custom terms, see Global Vocabulary Reference.

Register User-Specific Vocabulary

You register user-specific vocabulary terms at runtime by calling the `setVocabularyStrings(_:of:)` method of an `INVocabulary` object. Unlike the global terms, which you declare in your app bundle, you declare user-specific terms at runtime from your iOS app. The terms you declare must belong to one of the following categories:

- Audiobook titles
- Audiobook author names
- Car names
- Contact names (only if they're app-specific and not managed by the Contacts framework)
- Contact groups

- Music artist names
- Notebook titles
- Notebook groups
- Payment organization names
- Payment account nicknames
- Photo tags
- Photo album names
- Podcast or radio show titles
- Playlist titles
- Workout names
- Vehicle profile names

When selecting the vocabulary to register, choose terms that could be misunderstood by someone not familiar with your app. For example, include terms whose literal meanings differ from your app's usage of those terms. Don't register terms that are easily understood, such as "My Photo Album" or "My Workout".

Each call to the `setVocabularyStrings(_:of:)` method replaces any previously registered terms of the same type. As a result, each call to that method must include all of the terms that you want to associate with the user. In addition, you must arrange terms based on priority, placing the most important terms first in the list you provide.

The listing below shows an example that registers a set of custom workout names for the user. The example assumes that the app provides a custom `sortedWorkoutNames` method that sorts the workout names based on how recently the user used them. You execute this code from your iOS app, and not from your Intents app extension.

Swift   Objective-C

```swift
let workoutNames = self.sortedWorkoutNames()
let vocabulary = INVocabulary.shared().
    setVocabularyStrings(workoutNames, of: .workoutActivityName)
```

Topics

Related Topics

Global Vocabulary Reference

Understand the keys and values included in your app’s global vocabulary file.

Specifying Synonyms for Your App Name

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

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.

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.

Localizing Your Vocabulary for Chinese Dialects

Apply emphasis markers to your pronunciation tips to assist Siri with Chinese dialects.