Article

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.

Overview

When the user interacts with Siri to request media, the system may send your app an intent that represents the user's request. To effectively communicate the user's request to your app, the system relies on the intents your app supports, the names of media items and performers, and the user's listening habits. The system also uses this information to make proactive suggestions and provide relevant search results. Your app can contribute to this information with Core Spotlight and Intents to improve the accuracy, relevance, and convenience of playing media.

App Selection streamlines voice-driven interactions. As the system learns about the user's listening habits, it intelligently predicts when the user intends for your app to handle a request, even if the user didn't mention your app by name. By adopting these Intents and Core Spotlight APIs, your app also helps refine App Selection's predictions.

For information about how Apple protects user privacy when your app shares information with the system, see the Siri section of Privacy - Features.

Related Sessions from WWDC20

Session 10073: Empower Your Intents

Session 10087: Design for Intelligence: Make Friends with "The System"

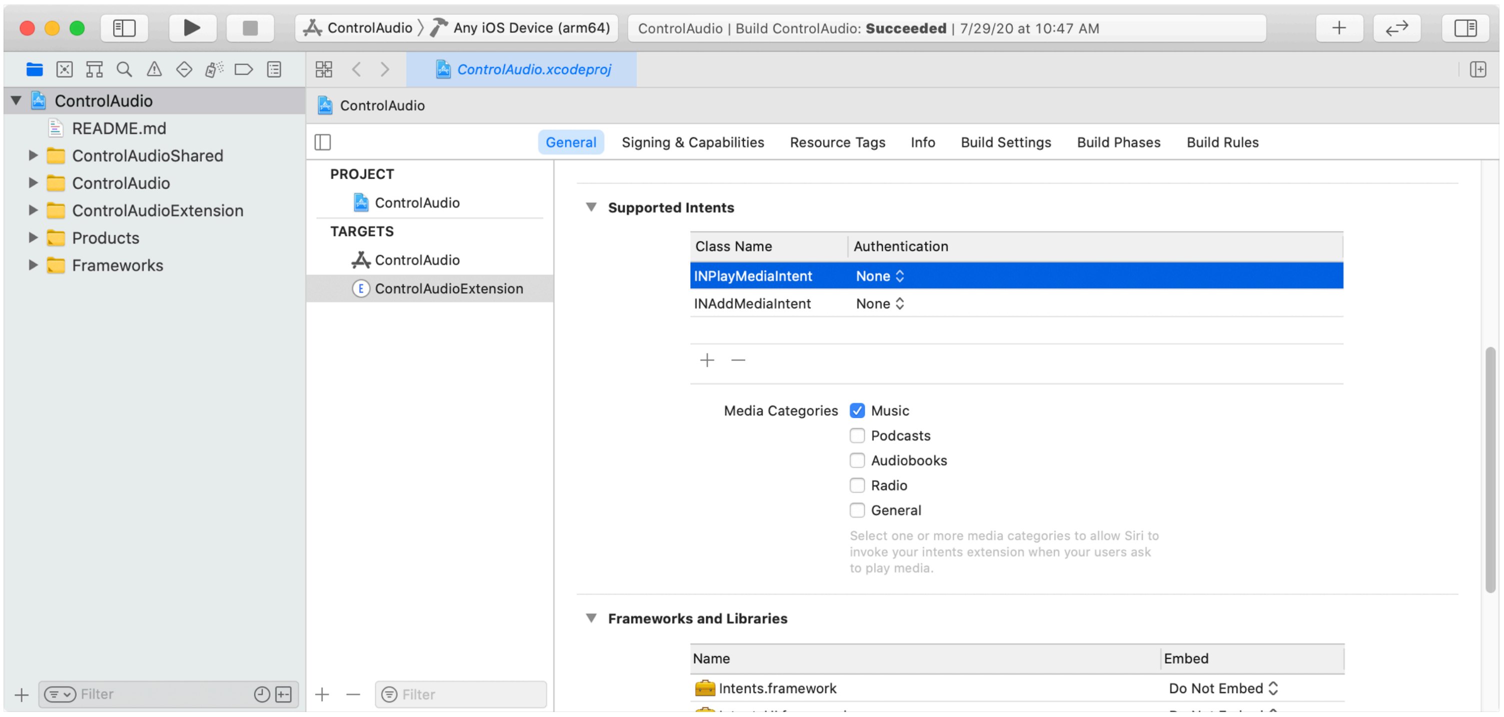
Session 10060: Design High Quality Siri Media Interactions

Support Media Intents

Use INPlayMediaIntent to receive media intents from Siri and Shortcuts. Support other system-defined media intents, such as INUpdateMediaAffinityIntent or INAddMediaIntent, that reflect interactions the user can perform directly in your app.

> **Tip**
>
> In iOS 14 and later, you can set up handling for INPlayMediaIntent directly in your app. To continue supporting earlier versions of iOS, resolve and confirm the intent in an extension, then respond with INPlayMediaIntentResponseCode.continueInApp or INPlayMediaIntentResponseCode.handleInApp so the system sends the intent to the app to play the media.

Tell the system what types of media your app supports in Xcode's Project editor. In the Supported Intents section of the Project editor, select each Media Category that describes your app's content. Select `General` if your app plays media that doesn't fit any of the other categories, like spooky sound effects or white noise. To opt in to Siri media control, you must choose at least one category for your app. With the settings shown here, users can say "Play some music in ControlAudio" to play music in the ControlAudio app. When App Selection is available, they can simply say "Play some music" and Siri intelligently picks the user's preferred music app with App Selection.



Supported Intents

Class Name	Authentication
INPlayMediaIntent	None
INAddMediaIntent	None

Media Categories
- [x] Music
- [ ] Podcasts
- [ ] Audiobooks
- [ ] Radio
- [ ] General

Select one or more media categories to allow Siri to invoke your intents extension when your users ask to play media.

Frameworks and Libraries

Name	Embed
Intents.framework	Do Not Embed

When a user searches for a musician or band on iOS 15 or later, Spotlight may include a suggestion to start listening to content from that artist. To include your app in the list of apps available to listen on, support INSearchForMediaIntent. If the user chooses your app from the list, or if your app is the user's preferred music app, the system sends your app an intent so you can display content by that artist.

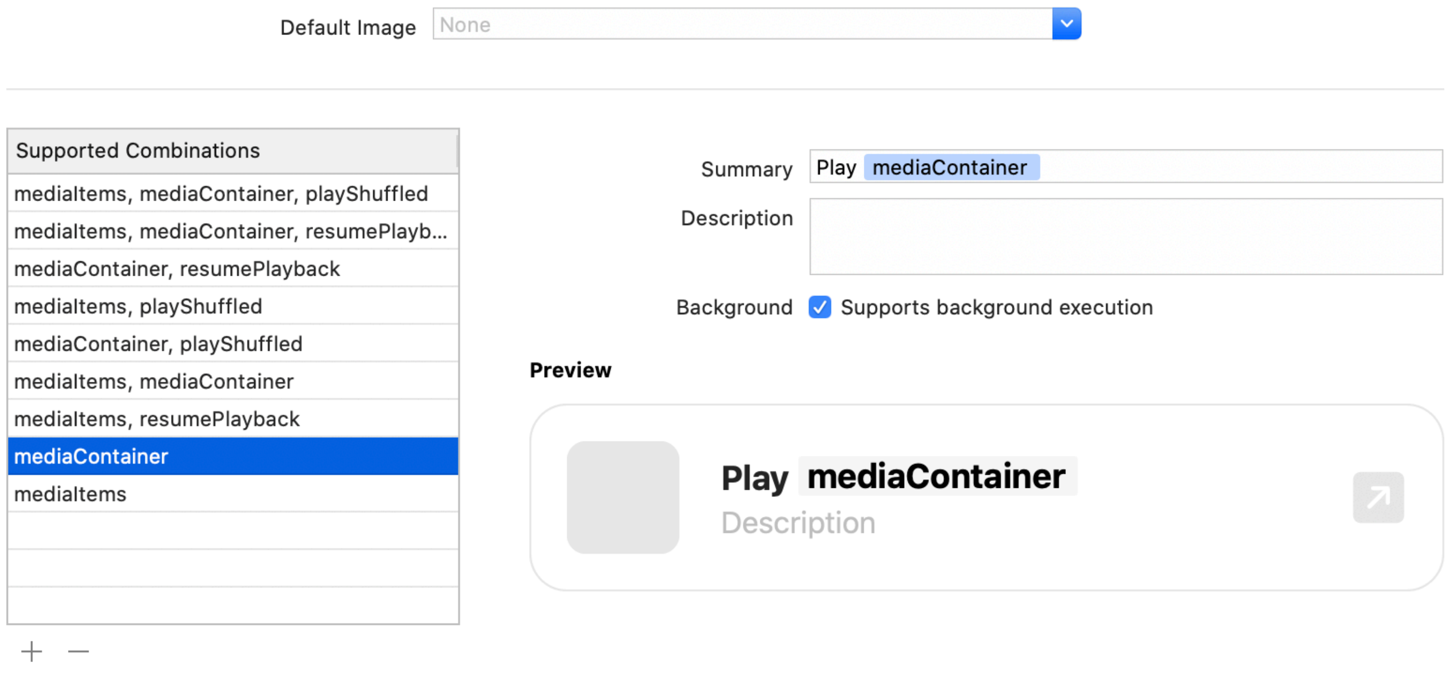
The system can also use your app's `INPlayMediaIntent` donations to offer personalized audio suggestions on Lock Screen, in Control Center, and in the Home app. To opt-in to these suggestions, you need to explicitly indicate that your app can play media container intents with no additional parameters. If you haven't already customized the PlayMedia system intent, follow these steps to set that up:

1. If your project doesn't already have an `.intentDefinition` file, use File > New File and then choose SiriKit Intent Definition File to add one to your project.

2. Open your `.intentDefinition` file.

3. Choose Customize System Intent > Media > Play Media from the + menu at the bottom of the intent definition editor.

Once you have the PlayMedia system intent in a SiriKit Intent Definition file, make sure that its list of Supported Combinations includes the entry with "mediaContainer" as the only parameter.



Default Image: None

Supported Combinations
mediaItems, mediaContainer, playShuffled
mediaItems, mediaContainer, resumePlayb...
mediaContainer, resumePlayback
mediaItems, playShuffled
mediaContainer, playShuffled
mediaItems, mediaContainer
mediaItems, resumePlayback
mediaContainer
mediaItems

Summary: Play mediaContainer

Description:

Background: ✓ Supports background execution

Preview

Play mediaContainer
Description

Note

The system offers the user audio suggestions based on donations the system can play in an audio-only situation. The system excludes the following media types when generating audio suggestions: `INMediaItemType.news`, `INMediaItemType.musicVideo`, `INMediaItemType.movie`, `INMediaItemType.tvShow`, `INMediaItemType.tvShowEpisode`, and `INMediaItemType.unknown`.

Donate Interactions

Improve App Selection, Shortcuts, and Shortcut Suggestions by donating interactions from your app. Each time the user starts playing media directly in your app, create and donate an `INInteraction` containing an `INPlayMediaIntent` to tell the system what the user is listening to. Provide as much detail as possible in each donation to help the system interact more accurately with the user in the future. If the user starts a playlist or radio station, donate that information instead of each individual song.

Important

Suggestions improve as the system accumulates donated interactions. If some media becomes irrelevant or unavailable, use `delete(with:completion:)` to remove interactions associated with those specific media items. Don't call `deleteAll(completion:)` unless you really mean to reset what the system has learned about the way the user interacts with your app.

Set the Current User Context

When your app launches, create an `INMediaUserContext`, add the user's information, and make the context current. The system uses this information to optimize the user experience for App Selection. The following code listing shows how to set the current context for a user with a paid subscription and a few hundred items in their library:

```
let context = INMediaUserContext()
context.numberOfLibraryItems = 345
context.subscriptionStatus = .subscribed
context.becomeCurrent()
```

Populate the Spotlight Database

Add metadata about playlists and individual media items from the user's local library to the Core Spotlight database. When the user searches for the item, the system can offer to play the media in your app alongside any other search results.

For Swift, choose the `completeUntilFirstUserAuthentication` level of protection for your Spotlight database so the system can access this information even when your app isn't running. In Objective-C, choose the `completeUntilFirstUserAuthentication` level of protection for your Spotlight database so the system can access this information even when your app isn't running.

```
let attributes = CSSearchableItemAttributeSet(contentType: .audio)
attributes.title = "Some Cool Song Title"
attributes.artist = "Hot New Artist"
attributes.album = "Hot New Artist's Exclusive Album"
attributes.genre = "Pop"
attributes.playCount = 5
attributes.lastUsedDate = Date()

let item = CSSearchableItem(uniqueIdentifier: "<identifier>", domainIdentifier: doma

CSSearchableIndex.default().indexSearchableItems([item]) { ... }
```

Warning

If you need to delete items from the index, use `deleteSearchableItems(withDomainIdentifiers:completionHandler:)`. Calling `deleteAllSearchableItems(completionHandler:)` deletes not only the searchable items you've added directly to the searchable index, but also every `INInteraction` your app has donated or Siri has donated on behalf of your app.

Define Relevant Vocabulary

Define vocabulary for Siri, both specific to the individual user and relevant for all users but exclusive to your app. Providing vocabulary helps Siri match a user's request to media items you've added to the Spotlight database or included in donated intents and interactions. Take care to include terms that include numbers or have other unusual spellings. There are two ways to provide vocabulary, and each one helps Siri interpret and route user requests:

- Global vocabulary relevant to all users of your app.
- User-specific vocabulary that you update as the user interacts with your app.

Define general terms in the Global Vocabulary Reference. Add an `AppIntentVocabulary.plist` file to your project and provide any vocabulary that's unique to your app but relevant for all users in the Parameter Vocabularies section. For more details, see Registering Custom Vocabulary with SiriKit.

Provide additional vocabulary programmatically with `INVocabulary`, like names that are unique or particularly important to the user, such as playlist titles. Put terms that are most important for this user or that Siri most often misunderstands first. You should update the vocabulary as needed, such as when the user renames or deletes a playlist, or begins listening to a particular artist more frequently.

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 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.

Localizing Your Vocabulary for Chinese Dialects

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