Article

Resolving the Parameters of an Intent

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

Overview

You must resolve the parameters of an intent to verify that you have the information you need to fulfill the user's request. When the user talks to Siri, Siri converts the user's spoken commands into actionable data and places that data in an intent object. During resolution, SiriKit prompts you to verify each parameter individually by calling the resolution methods of your handler object. In each method, you validate the provided data and create a resolution result object indicating your success or failure in resolving the parameter. SiriKit uses your resolution result object to determine how to proceed. For example, if your resolution result asks the user to disambiguate from among two or more choices, SiriKit prompts the user to select one of those choices.

The example below shows how a ride booking app might resolve the drop-off location of the user's ride. In this example, the user must provide a valid drop-off location before the app can fulfill the ride request. If the user supplies a drop-off location that is within the app's service area, the method returns a successful resolution result. If the drop-off location is outside the valid service area, the resolution result indicates that the location is unsupported. If the user did not specify a drop-off location at all, the method creates a resolution result indicating that the value is needed.

```
func resolveDropOffLocation(
            forRequestRide intent: INRequestRideIntent,
            with completion: @escaping (INPlacemarkResolutionResult) -> Void) {
   var result: INPlacemarkResolutionResult

   if let location = intent.dropOffLocation {
      // If the location is valid, use it; otherwise,
      // let the user know it is not supported
      if self.locationIsInsideServiceArea(location) {
```

```
            result = INPlacemarkResolutionResult.success(with: location)
        } else {
            result = INPlacemarkResolutionResult.unsupported()
        }
    } else {
        // Ask for the drop-off location.
        result = INPlacemarkResolutionResult.needsValue()
    }

    // Return the result back to SiriKit.
    completion(result)
}
```

Each resolvable parameter has a corresponding resolution result class that you use to communicate your results back to SiriKit. For example, resolving the `CLPlacemark` object in the preceding example requires the creation of an `INPlacemarkResolutionResult` object. All resolution result classes inherit from the `INIntentResolutionResult` base class, which defines the common ways to resolve parameters of any type.

When returning a resolution to SiriKit, always include as much information as you can about the actual result. After each resolution attempt, SiriKit updates the intent object with the information you provide. If resolution was unsuccessful and SiriKit needs to call your resolution method again, you can use the extra information to reach a resolution more quickly. For example, if you find two contacts with the same name during resolution, you might populate the `customIdentifier` property of each `INPerson` object with a unique string that you can use to look up the selected contact later. Your resolution method can then look for that identifier first and bypass any other contact-matching logic.

Choosing the Correct Resolution Result

The resolution result object you give back to SiriKit determines whether any further user interactions will occur. You must resolve all parameters successfully or let SiriKit know that the parameter is not required. All other resolutions require Siri to gather more information from the user.

To create a resolution result object, choose the appropriate class method. Each specific resolution result class has methods defining the possible resolutions for the corresponding type. The `INIntentResolutionResult` base class also defines a set of common resolutions. This table lists the resolution types and their meaning:

Resolution	Meaning
Success	A successful resolution means that you can use the value to handle the user's request. Use this resolution even if it means making an educated guess about what the user meant. A resolution of this type does not require further user interactions. *Example:* If the user specified the name "John" for a person and has only one contact with that name, you could resolve the parameter successfully to that contact.
Not required	Specify this resolution for parameters that your service does not use. A resolution of this type does not require further user interactions. *Example:* You might specify this resolution for workout goals that your fitness app does not support.
Needs disambiguation	Specify this resolution when you identify two or more possible results and cannot choose one definitively. A resolution of this type causes SiriKit to prompt the user to select one of the provided options. *Example:* If the user specified the name "John" for a person and there are two equally likely contacts with that name, you could ask the user to choose one of them.
Needs confirmation	Specify this resolution when you want the user to confirm the value that you choose. This resolution is for cases where you can choose a single value, but are not completely confident that it is the correct value. A resolution of this type causes SiriKit to prompt the user to confirm the value. *Example:* If the there are two contacts with the name "John", one of whom is marked as a favorite and one of whom is used infrequently, select the favorite and ask the user to confirm the choice.
Value required	Specify this resolution when no value was specified for the parameter, but you require one. A resolution of this type causes SiriKit to prompt the user for a value. *Example:* Return this resolution if the user requests a payment from another person but does not specify the payment amount.
Unsupported	Specify this resolution when the value supplied by the user is invalid or conflicts with the value of other parameters. A resolution of this type causes SiriKit to notify the user that the value is not supported.

Resolution	Meaning
	Example: Return this resolution if the user wants to make a payment in Canadian Dollars but your app supports only Euros or US Dollars.

Try to reach a resolution of “Success” or “Not required” as quickly as possible. Other resolution results lead to additional user interactions and additional calls to your handler. You should also avoid repeatedly asking for clarification of the same parameter, which could frustrate the user. Whenever possible, choose reasonable values based on the user’s patterns and habits, and ask for disambiguation or confirmation only as needed.

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.

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.