

## Documentation

< All Technologies

### App Intents

Parameter resolution

Intent parameters

> [IntentParameter](#)

> [IntentParameterDependency](#)

> [IntentParameterContext](#)

> [InputConnectionBehavior](#)

Parameter choices

> [DynamicOptionsProvider](#)

> [AppEnum](#)

Shortcuts support

> [ParameterSummary](#)

> [IntentParameterSummary](#)

> [ParameterSummaryString](#)

> [ParameterSummaryWhenCondition](#)

> [ParameterSummarySwitchCondition](#)

> [ParameterSummaryCaseCondition](#)

> [ParameterSummaryDefaultCaseCondition](#)

> [App entities](#)

> [Entity queries](#)

[Filter](#)

/

[App Intents / Parameter resolution](#)

## API Collection

# Parameter resolution

Define the required parameters for your app intents and specify how to resolve those parameters at runtime.

## Overview

Parameters represent input arguments to your app intents and offer additional metadata to the system.

When you define an app intent, add the `@Parameter` property wrapper to any properties you use as input. For example, an app intent that sends a message might include a parameter for the recipient and message string. The system collects and resolves the relevant parameter information before it performs your app intent.

The following partial example shows how to declare parameters for a custom app intent that enables someone to order soup from your app. Configure the `parameter` property wrapper with any additional details that help the system infer extra information about your parameter.

```
struct OrderSoupIntent: AppIntent {
    @Parameter(title: "Soup")
    var soup: Soup

    @Parameter(title: "Quantity", inclusiveRange: (1, 10))
    var quantity: Int

    // Other properties
}
```

## Topics

### Intent parameters

`class IntentParameter`

A property wrapper that indicates the associated property is an input argument of the app intent.

`class IntentParameterDependency`

A property wrapper that represents an app intent dependency you use to provide dynamic options.

`struct IntentParameterContext`

A type that provides information about an associated parameter during value resolution.

`enum InputConnectionBehavior`

Describes the input behaviors for connecting a parameter to the output of the previous App Intent.

### Parameter choices

`protocol DynamicOptionsProvider`

An interface for providing a dynamic list of options for a parameter of your app intent.

`protocol AppEnum`

An interface to express that a custom type has a predefined, static set of valid values to display.

### Shortcuts support

`protocol ParameterSummary`

An interface for defining the visual representation of an app intent's parameters.

`struct IntentParameterSummary`

A type that describes the user interface configuration of an app intent's parameters.

`struct ParameterSummaryString`

A human-readable string that interpolates parameter key paths to provide user-configurable placeholders in the Shortcuts app.

`struct ParameterSummaryWhenCondition`

A type that represents a conditional statement in a parameter summary.

`struct ParameterSummarySwitchCondition`

A type that represents a switch statement in a parameter summary.

`struct ParameterSummaryCaseCondition`

A type that represents an individual case of a switch statement in a parameter summary.

`struct ParameterSummaryDefaultCaseCondition`

A type that represents the default case of a switch statement in a parameter summary.

## See Also

### Parameters, custom data types, and queries

[Adding parameters to an app intent](#)

Enable people to configure app intents with their custom input values.

[Integrating custom data types into your intents](#)

Provide the system with information about the types your app uses to model its data so that your intents can use those types as parameters.

[App entities](#)

Make core types or concepts discoverable to the system by declaring them as app entities.

[Entity queries](#)

Help the system find the entities your app defines and use them to resolve parameters.

[Resolvers](#)

Resolve the parameters of your app intents, and extend the standard resolution types to include your app's custom types.

Apple > Developer > Documentation

#### Platforms

iOS

iPadOS

macOS

tvOS

visionOS

watchOS

#### Tools

Swift

SwiftUI

Swift Playground

TestFlight

Xcode

Xcode Cloud

SF Symbols

#### Topics & Technologies

Accessibility

Accessories

App Extension

App Store

Audio & Video

Augmented Reality

Design

Distribution

Education

Fonts

Games

Health & Fitness

In-App Purchase

Localization

Maps & Location

Machine Learning & AI

Open Source

Security

Safari & Web

#### Resources

Documentation

Tutorials

Downloads

Forums

Videos

Support

Support Articles

Contact Us

Bug Reporting

System Status

Account

Apple Developer

App Store Connect

Certificates, IDs, & Profiles

Feedback Assistant

#### Programs

Apple Developer Program

Apple Developer Enterprise Program

App Store Small Business Program

MFi Program

News Partner Program

Video Partner Program

Security Bounty Program

Security Research Device Program

#### Events

Meet with Apple

Apple Developer Centers

App Store Awards

Apple Design Awards

Apple Developer Academies

WWDC

To submit feedback on documentation, visit [Feedback Assistant](#).

Light Dark Auto