WidgetKit / Widgets and watch complications

API Collection

# Widgets and watch complications

Allow people to personalize their devices, view relevant information, and perform interactions with widgets and watch complications.

## Overview

Use WidgetKit to create widgets and watch complications that elevate a small amount of timely, personally relevant information from your app and allow people to perform quick actions without launching your app:

- On iPhone and iPad, people put widgets in Today View, on the Home Screen, and on the Lock Screen.
- On Mac, people place macOS widgets and widgets from a paired iPhone in locations like the Mac desktop and in Notification Center.
- On Apple Watch, widgets appear in the Smart Stack and as watch complications.
- On Apple Vision Pro, widgets become three-dimensional objects that people pin to horizontal and vertical surfaces.
- In CarPlay, people configure iPhone widgets to appear on the widgets screen.

## Topics

### Widget creation

Creating a widget extension

Display your app’s content in a convenient, informative widget on various devices.

Developing a WidgetKit strategy

Explore features, tasks, related frameworks, and constraints as you make a plan to implement widgets, controls, watch complications, and Live Activities.

{} Emoji Rangers: Supporting Live Activities, interactivity, and animations

Offer Live Activities, controls, animate data updates, and add interactivity to widgets.

Preparing widgets for additional platforms, contexts, and appearances

Create widgets that support additional platforms and adapt to their context.

`@MainActor @preconcurrency protocol Widget`

The configuration and content of a widget to display on the Home screen or in Notification Center.

`enum WidgetFamily`

Values that define the widget’s size and shape.

`struct StaticConfiguration`

An object describing the content of a widget that has no user-configurable options.

## Configurable widgets

Making a configurable widget

Give people the option to customize their widgets by adding a custom app intent to your project.

Migrating widgets from SiriKit Intents to App Intents

Configure your widgets for backward compatibility.

`struct AppIntentConfiguration`

An object describing the content of a widget that uses a custom intent to provide user-configurable options.

`struct WidgetInfo`

A structure that contains information about user-configured widgets.

## Layout and presentation

Supporting additional widget sizes

Offer widgets in additional contexts by adding support for various widget sizes.

Displaying the right widget background

Group your widget’s background views and mark them as removable to ensure your widget appears correctly for each context and platform.

Optimizing your widget for accented rendering mode and Liquid Glass

Make your widget feel at home on Apple platforms and Liquid Glass by using accented rendering mode.

Adding StandBy and CarPlay support to your widget

Ensure that your small system family widget works well in StandBy and CarPlay.

`struct WidgetRenderingMode`

Constants that indicate the rendering mode for a widget.

`struct WidgetAccentedRenderingMode`

Constants that indicate the rendering mode for an `Image` in when displayed in a widget in `accented` mode.

`struct AccessoryWidgetBackground`

An adaptive background view that provides a standard appearance based on the the widget’s environment.

`struct WidgetLocation`

Values that indicate different widget locations.

## Timeline updates

Keeping a widget up to date

Plan your widget’s timeline to show timely, relevant information using dynamic views, and update the timeline when things change.

`protocol TimelineProvider`

A type that advises WidgetKit when to update a widget’s display.

`protocol AppIntentTimelineProvider`

A type that advises WidgetKit when to update a user-configurable widget’s display.

`protocol IntentTimelineProvider`

A type that advises WidgetKit when to update a user-configurable widget’s display.

`struct TimelineProviderContext`

An object that contains details about how a widget is rendered, including its size and whether it appears in the widget gallery.

`protocol TimelineEntry`

A type that specifies the date to display a widget, and, optionally, indicates the current relevance of the widget’s content.

`struct Timeline`

An object that specifies a date for WidgetKit to update a widget’s view.

`class WidgetCenter`

An object that contains a list of user-configured widgets and is used for reloading widget timelines.

## Push notification updates

📄 Updating widgets with WidgetKit push notifications

Use WidgetKit to receive push tokens and reload your widgets with remote push notifications.

`protocol WidgetPushHandler`

A type that can receive push information about widget refreshes and relevance refreshes.

`struct WidgetPushInfo`

A structure that contains information about the push token for updating widgets and widget relevances.

## Capabilities

📄 Accessing location information in widgets

Incorporate location information into your widget presentation to make it more relevant and contextual.

📄 Making network requests in a widget extension

Update your widget with new information you fetch with a network request.

## Debugging

📄 Previewing widgets and Live Activities in Xcode

Use Xcode previews to iteratively develop, fine-tune, and troubleshoot widgets and Live Activities.

Preview macros

Use Swift macros to create widget previews in Xcode.

```
struct WidgetPreviewContext
```

A specification for the context of a widget preview.

Debugging widgets

Set environment variables in Xcode to control your widget’s configuration in the debugger.

## visionOS widgets

Updating your widgets for visionOS

Choose widget styles specific to visionOS, support recessed and elevated appearances, and add proximity awareness to your widget.

```
@MainActor @preconcurrency func widgetTexture(_ material: WidgetTexture
) -> some WidgetConfiguration
```

Specifies the widget texture for this widget.

```
struct WidgetTexture
```

Values that define the texture of the widget’s coating layer.

```
@MainActor @preconcurrency func supportedMountingStyles(_ styles: [
WidgetMountingStyle]) -> some WidgetConfiguration
```

Specifies the mounting style for this widget.

```
struct WidgetMountingStyle
```

Values that define the widget’s supported mounting style.

```
struct LevelOfDetail
```

The level of detail the view is recommended to have.

## Accessory and watchOS widgets

Creating accessory widgets and watch complications

Support accessory widgets that appear on the Lock Screen and as complications on Apple Watch.

```
struct AccessoryWidgetGroup
```

A view type that has a label at the top and three content views masked with a circle or rounded square.

`struct AccessoryWidgetGroupStyle`

The style for an `AccessoryWidgetGroup` view.

Migrating ClockKit complications to WidgetKit

Leverage WidgetKit's API to create watchOS complications using SwiftUI.

## Smart Stacks

Increasing the visibility of widgets in Smart Stacks

Provide contextual information and donate intents to the system to make sure your widget appears prominently in Smart Stacks.

`struct TimelineEntryRelevance`

An object that describes the relative importance of a timeline entry compared to other entries in the current and past timelines.

`struct RelevanceConfiguration`

A type that describes the content of a widget that uses relevance clues.

`protocol RelevanceEntriesProvider`

A type that provides the content for a widget that uses relevance clues to display information in the Smart Stack.

`protocol RelevanceEntry`

A type that specifies the information to render a widget at a specific relevance configuration.

`struct WidgetRelevance`

A type collecting the relevances for a widget kind.

`struct WidgetRelevanceAttribute`

A type that describes when a specific widget could be relevant.

`struct WidgetRelevanceGroup`

A type for configuring widget behavior in the watchOS Smart Stack.

`struct AppIntentRecommendation`

An object that describes a recommended intent configuration for a user-customizable widget.

`struct IntentConfiguration`

An object describing the content of a widget that uses a custom intent definition to provide user-configurable options.

```
struct IntentRecommendation
```

An object that describes a recommended intent configuration for a user-customizable widget.

# See Also

## System experiences

Live Activities

Let people track updates from your app with Live Activities.

Controls

Offer controls that people place in Control Center, on the Lock Screen, and on the Action button to quickly perform an action from your app.