Article

Creating accessory widgets and watch complications

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

Overview

WidgetKit allows you to extend the reach of your app to the Lock Screen on iPhone and iPad, and to the Smart Stack on Apple Watch as *accessory widgets*. Accessory widgets also appear as watch complications on Apple Watch. They display your app's most relevant, glanceable content in highly visible contexts and let people quickly access your app for more details.

Widgets and watch complications use WidgetKit and SwiftUI views, enabling you to:

- Update your existing widgets' code to support accessory widgets.
- Offer WidgetKit complications for your watchOS app, replace ClockKit complications, and share more code between your iOS and watchOS apps.
- Create watch complications and Lock Screen widgets simultaneously.
- Add support for iOS or watchOS to your app — in addition to platforms you already support — and offer widgets or complications.

Your path to creating accessory widgets and WidgetKit complications depends on your project. Before you start coding, make sure to review options available to you and carefully plan your software development project. If you're new to WidgetKit development, see Developing a WidgetKit strategy.

Related sessions from WWDC22

Session 10050: Complications and widgets: Reloaded and Session 10051: Go further with complications in WidgetKit

Related sessions from WWDC23

Session 10309:Design widgets for the Smart Stack on Apple Watch and Session 10027: Bring widgets to new places

Create a new app that offers widgets and watch complications

If you create a new app, support iOS and watchOS and offer widgets and complications right from the start. This allows you to create a shared code base and reduces the need for costly code changes later in the app development process.

To get started:

1. Open Xcode and create a new project.
2. Choose the watchOS app template and select Watch App with New Companion iOS App in the "Choose options for your new project" sheet.
3. Add a new watchOS widget extension target to your project.
4. Add a new iOS widget extension target to your project.
5. Add code for your iOS and watchOS apps, and create iOS widgets — including for the Lock Screen — and watchOS complications simultaneously using SwiftUI views and WidgetKit.

To learn more about creating widgets and building apps with SwiftUI, see Creating a widget extension and SwiftUI. For more information on creating watchOS apps, see watchOS apps.

Add accessory widgets to your existing iOS App

If your app supports widgets, adding accessory widgets works like supporting any other widget size. To learn more, see Supporting additional widget sizes.

If you don't offer widgets in your app, add a widget extension to your app and support accessory widgets from the start. To learn more about adding a widget extension to your project, see Creating a widget extension.

Additionally, consider adding support for watchOS with WidgetKit complications by adding a watchOS app target and a watchOS widget extension target. To learn more about creating a watchOS app, see watchOS apps.

Add WidgetKit complications to your existing watchOS app

If your existing watchOS app doesn't offer complications, add a watchOS widget extension to your project in Xcode, and create complications with WidgetKit. Implementing WidgetKit complications works like creating widgets. For more information about creating a widget extension, see Creating a widget extension.

If your watchOS app offers ClockKit complications, continue to use these to support watchOS 8 and earlier. However, plan to migrate your existing ClockKit complications to WidgetKit as described in Migrating ClockKit complications to WidgetKit.

Important

As soon as you offer a widget-based complication, the system stops calling ClockKit APIs. For example, it no longer calls your `CLKComplicationDataSource` object's methods to request timeline entries. The system may still wake your data source for migration requests.

Additionally, consider adding support for iOS by adding an iOS app target to your Xcode project if your watchOS app doesn't come with an iOS companion app.

See Also

Accessory and watchOS widgets

`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.

Copyright © 2025 Apple Inc. All rights reserved. Terms of Use | Privacy Policy | Agreements and Guidelines