Article

Linking to specific app scenes from your widget or Live Activity

Add deep links to your widgets and Live Activities that enable people to open a specific scene in your app.

Overview

People interact with a widget or Live Activity to launch a scene in the corresponding app with matching content and functionality. For example, when people click or tap a Stocks widget, the Stocks app opens to a page that displays information about that stock price.

When you create widgets and Live Activities, think about how people interact with them. Make sure interactions launch the scene in your app that fits the widget's content or the Live Activity.

Launch a specific screen in your app

By default, tapping or clicking your widget or Live Activity opens its corresponding app. To provide a good experience and not make people navigate to get to the right place in your app, open the app at a scene that matches the content of the widget or Live Activity. To open a specific screen in your app, add the `widgetURL(_:)` modifier to a view in the view hierarchy of your widget or Live Activity.

Important

If the view hierarchy includes more than one `widgetURL` modifier, the behavior is undefined.

For example, the following code snippet from the Emoji Rangers: Supporting Live Activities, interactivity, and animations sample code project shows how the small widget uses `widgetURL(_:)` to allow people to open the app and show a character's detail information:

```
struct EmojiRangerWidgetEntryView: View {
    var entry: SimpleEntry

    @Environment(\.widgetFamily) var family

    @ViewBuilder
    var body: some View {
        switch family {
        case .systemSmall:
            AvatarView(entry.hero)
                .foregroundStyle(.white)
                .widgetBackground()
                .widgetURL(entry.hero.url)

        // Code for other widget sizes.
    }
}
```

For widgets with enough space for more than one interaction target — `WidgetFamily.accessoryRectangular`, `WidgetFamily.systemSmall`, and larger system family sizes — add one or more `Link` controls to your view hierarchy. You can use one `widgetURL` and additional `Link` controls. If an interaction targets a `Link` control, the system uses the URL in that control. For interactions anywhere else in the widget, the system uses the URL you specify in the `widgetURL(_:)` view modifier.

For example, the leaderboard widget of the Emoji Rangers: Supporting Live Activities, interactivity, and animations app displays a list of characters. Each item in the list uses a `Link` control to launch the scene in the app for the specific character that the item represents.

> **Note**
>
> When the widget or Live Activity receives an interaction, the system activates the containing app and passes the URL to `onOpenURL(perform:)`, `application(_:open:options:)`, or `application(_:open:)`, depending on the life cycle your app uses.

Detect the originating widget by accessing the user activity object

If a widget doesn't specify a deep link URL with `widgetURL(_:)` or `Link` and a person interacts with it, the system opens the containing app and passes an `NSUserActivity` to `onContinueUserActivity(_:perform:)`, `application(_:continue:restorationHandler:)`, or

`application( _:continue:restorationHandler:)`. The user activity's `userInfo` dictionary contains details about the widget the person interacted with. Use the keys in `WidgetCenter.UserInfoKey` to access these values from Swift code. To access the `userInfo` values from Objective-C, use the keys `WGWidgetUserInfoKeyKind` and `WGWidgetUserInfoKeyFamily` instead. Then, update your app's interface to match the widget so people don't have to navigate to the right place in your app.

> **Note**
>
> If you use an `AppIntentConfiguration` to configure your widget, use the `widgetConfigurationIntent(of:)` function to access the widget's intent. Similarly, if you use an `IntentConfiguration`, the user activity's `interaction` property contains the associated `INIntent`.

Review linking behavior in CarPlay

In CarPlay, linking from your widget to your app works differently to match the specific context of using your app while using your car. For more information about supporting CarPlay, see Adding StandBy and CarPlay support to your widget.

See Also

Interactivity

- Adding interactivity to widgets and Live Activities

  Include buttons or toggles in a widget or Live Activity to offer app functionality without launching the app.

- Animating data updates in widgets and Live Activities

  Use SwiftUI animations to indicate data updates in your widgets and Live Activities.