

[UIKit](#) / Adding accessible descriptions to widgets and Live Activities

Article

Adding accessible descriptions to widgets and Live Activities

Describe the interface elements of your widgets and Live Activities to help people understand what they represent.

Overview

Designing with accessibility in mind is a foundational principle when creating an app. It also applies to widgets and Live Activities. To allow people to customize how they interact with your widget or Live Activity, to verify VoiceOver works correctly for them, and to help people understand what each interface element represents, add accessibility labels to the SwiftUI views you create for each widget and Live Activity presentation.

Provide accessibility labels

Add accessibility labels for each SwiftUI view you use as needed and make sure your accessibility labels fit the widget or Live Activity content. To review API that allows you to add accessible descriptions to SwiftUI views, see [Accessible descriptions](#).

The example below shows how the [Emoji Rangers: Supporting Live Activities, interactivity, and animations](#) app uses the `accessibilityLabel(_:)` modifier to add accessibility labels for minimal, compact leading, and compact trailing presentations.

```
import SwiftUI
import WidgetKit

struct AdventureActivityConfiguration: Widget {
    var body: some WidgetConfiguration {
        ActivityConfiguration(for: AdventureAttributes.self) { context in
```

```

        // Create the presentation that appears on the Lock Screen and as a
        // banner on the Home Screen of devices that don't support the
        // Dynamic Island.
        // ...
    } dynamicIsland: { context in
        // Create the presentations that appear in the Dynamic Island.
        DynamicIsland {
            // Create the expanded presentation.
            // ...
        } compactLeading: {
            // Create the compact leading presentation.
            Avatar(hero: context.attributes.hero, includeBackground: true)
                .accessibilityLabel("The avatar of \(context.attributes.hero.name)")
        } compactTrailing: {
            // Create the compact trailing presentation.
            ProgressView(value: context.state.currentHealthLevel, total: 1) {
                let healthLevel = Int(context.state.currentHealthLevel * 100)
                Text("\(healthLevel)")
                    .accessibilityLabel("Health level at \(healthLevel) percent.")
            }
            .progressViewStyle(.circular)
            .tint(context.state.currentHealthLevel <= 0.2 ? Color.red : Color.green)
        } minimal: {
            // Create the minimal presentation.
            ProgressView(value: context.state.currentHealthLevel, total: 1) {
                Avatar(hero: context.attributes.hero, includeBackground: false)
                    .accessibilityLabel("The avatar of \(context.attributes.hero.name)")
            }
            .progressViewStyle(.circular)
            .tint(context.state.currentHealthLevel <= 0.2 ? Color.red : Color.green)
        }
    }
    .supplementalActivityFamilies([.small, .medium])
}
}

```

If you provide a content description for an image you update while a Live Activity is active and the image conveys status information or similar, make sure to also update the accessibility label to match the updated status. For example, if an image indicates a delivery status, make sure the accessibility label changes as the delivery status that the image indicates changes. Similarly, update accessibility labels when a widget updates displayed images or SwiftUI views. For guidance on providing content descriptions, see [Human Interface Guidelines > Accessibility > VoiceOver > Content descriptions](#).

See Also

User interface

Creating custom views for Live Activities

Create reusable custom views and layouts that support each Live Activity presentation.

Launching your app from a Live Activity

Use deep links to enable people to open your app's scene that matches the data of you Live Activity.