Article

# Animating data updates in widgets and Live Activities

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

## Overview

Animations bring your widgets and Live Activities to life and alert a person when new information is available. Widgets and Live Activities animate data updates with default animations or SwiftUI animations you choose, bringing a person's attention to updated data. In earlier OS versions, widgets don't animate, and Live Activities only use a subset of SwiftUI transitions and animations.

> **Note**
> Animations in widgets and Live Activities have a maximum duration of two seconds.

For example, text views animate content changes with blurred content transitions by default, and changes to images and SF Symbols animate with default content transitions. If you add or remove views from the interface based on timeline updates or other state changes, views fade in and out.

> Related session from WWDC23
> Session 10028: Bring widgets to life

To replace default animations and transitions:

- Configure built-in transitions like `opacity`, `move(edge:)`, `slide`, `push(from:)`, or combinations of them.
- Add `transition(_:)`, `contentTransition(_:)`, or `animation(_:value:)` to views.
- Request animations for timer text with `numericText(countsDown:)`.

> **Important**
> On devices that include an Always-On display, the system doesn't perform animations to preserve battery life in Always On. Check the `isLuminanceReduced` environment value to detect reduced luminance before animating content changes.

For Live Activities that appear on devices that run iOS 16 or earlier, the system ignores any animation modifiers — for example, `withAnimation(_:_:)` and `animation(_:value:)` — and uses the system's animation timing instead. However, you can use built-in transitions like `opacity`, `move(edge:)`, `slide`, `push(from:)`, or combinations of them.

For more information about SwiftUI animations, refer to Animations.

### Add transitions and animations to views that update their data

In addition to the default transitions and animations that the system performs when views update their data, you can choose other built-in SwiftUI transitions and animations. Widgets and Live Activities support all built-in SwiftUI transitions and animations. For example, you could configure a content transition for numeric text as shown in this example:

```
Text(totalCaffeine.formatCaffeine())
    .font(.title)
    .minimumScaleFactor (0.8)
    .contentTransition(.numericText())
```

Additionally, you could add a spring animation to the transition:

```
Text (totalCaffeine.formatCaffeine())
    .font(.title)
    .minimumScaleFactor (0.8)
    .contentTransition(.numericText())
    .animation(.spring(duration: 0.2), value: totalCaffeine)
```

To use custom text animations, use `contentTransition(_:)` as shown in the example above. To use the default text animation, use `transition(_:)`, and customize its speed and delay as shown in the following example:

```
Text("Some text with \(entry.value) that changes.")
    .animation(.default.speed(0.25).delay(0.5), value: entry.value)
```

### Add transitions and animations to additional views

In addition to adding transitions or animations to a view that changes its data, you can animate a view when other widget information changes. To animate a view when a certain value changes, first associate the view you want to animate with that value's data model object. This is easiest when your data model conforms to the `Hashable` protocol. If your data model doesn't conform to `Hashable`, change its code accordingly. Then, associate the view with the data model using the `id(_:)` view modifier. Finally, add a transition or animation.

The following example shows how the `LastDrinkView` adds a push transition when the associated `log` changes.

```
struct LastDrinkView: View {
    let log: CaffeineLog
    var dateFormatStyle: Date.FormatStyle {
        Date.FormatStyle(date: .omitted, time: .shortened)
    }

    var body: some View {
        VStack(alignment: .leading) {
            Text(log.drink.name)
                .bold()
            Text ("\(log.date, format: .dateformatStyle) • \(log.drink
        }
        .font (.caption)
        .id(log) // Associate the view with the data model.
        .transition(.push(from: .bottom))
    }
}
```

### Disable animations

If a content update changes many views in your widget or Live Activity, consider disabling transitions and animations for some views to direct a person's attention to the most important updates. To disable animations for a view, including default animations, pass `identity` to `transition(_:)` or `nil` to the `animation` parameter of `withAnimation(_:_:)` and `animation(_:value:)`.

> **Note**
> `Transaction` isn't available to widgets and Live Activities, so you can't cancel or deactivate an animation by setting the transaction's `animation` property to `nil`.

## 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.
- 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.