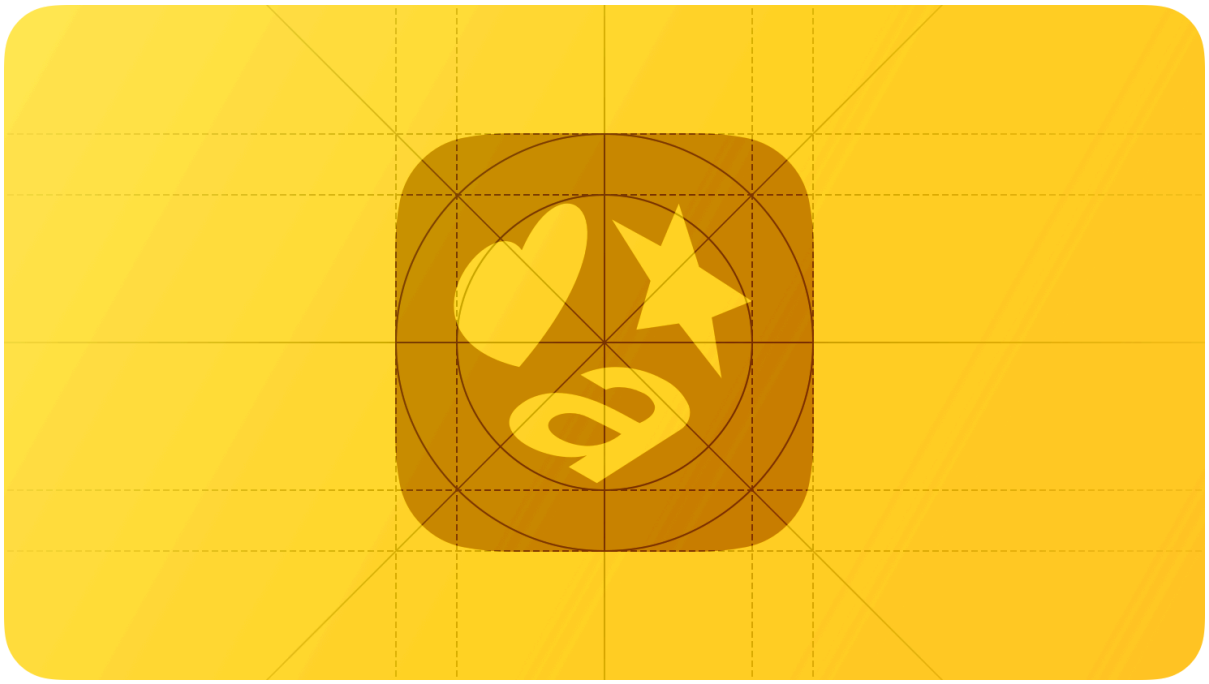
SF Symbols

SF Symbols provides thousands of consistent, highly configurable symbols that integrate seamlessly with the San Francisco system font, automatically aligning with text in all weights and sizes.



You can use a symbol to convey an object or concept wherever interface icons can appear, such as in navigation bars, toolbars, tab bars, context menus, and within text.

Availability of individual symbols and features varies based on the version of the system you're targeting. Symbols and symbol features introduced in a given year aren't available in earlier operating systems.

Visit SF Symbols to download the app and browse the full set of symbols. Be sure to understand the terms and conditions for using SF Symbols, including the prohibition against using symbols — or images that are confusingly similar — in app icons, logos, or any other trademarked use. For developer guidance, see Configuring and displaying symbol images in your UI.

Supported platforms

SF Symbols
Rendering modes
Variable color
Weights and scales
Design variants
Animations
Custom symbols
Platform considerations
Resources
Change log

Rendering modes

SF Symbols provides four rendering modes — monochrome, hierarchical, palette, and multicolor — that give you multiple options when applying color to symbols. For example, you might want to use multiple opacities of your app's accent color to give symbols depth and emphasis, or specify a palette of contrasting colors to display symbols that coordinate with various color schemes.

To support the rendering modes, SF Symbols organizes a symbol's paths into distinct layers. For example, the `cloud.sun.rain.fill` symbol consists of three layers: the primary layer contains the cloud paths, the secondary layer contains the paths that define the sun and its rays, and the tertiary layer contains the raindrop paths.

Primary Secondary Tertiary

Depending on the rendering mode you choose, a symbol can produce various appearances. For example, Hierarchical rendering mode assigns a different opacity of a single color to each layer, creating a visual hierarchy that gives depth to the symbol.

To learn more about supporting rendering modes in custom symbols, see Custom symbols.

SF Symbols supports the following rendering modes.

Monochrome — Applies one color to all layers in a symbol. Within a symbol, paths render in the color you specify or as a transparent shape within a color-filled path.

Hierarchical — Applies one color to all layers in a symbol, varying the color's opacity according to each layer's hierarchical level.

Palette — Applies two or more colors to a symbol, using one color per layer. Specifying only two colors for a symbol that defines three levels of hierarchy means the secondary and tertiary layers use the same color.

Multicolor — Applies intrinsic colors to some symbols to enhance meaning. For example, the `leaf` symbol uses green to reflect the appearance of leaves in the physical world, whereas the `trash.slash` symbol uses red to signal data loss. Some multicolor symbols include layers that can receive other colors.

Regardless of rendering mode, using system-provided colors ensures that symbols automatically adapt to accessibility accommodations and appearance modes like vibrancy and

Dark Mode. For developer guidance, see renderingMode(_:).

Confirm that a symbol's rendering mode works well in every context. Depending on factors like the size of a symbol and its contrast with the current background color, different rendering modes can affect how well people can discern the symbol's details. You can use the automatic setting to get a symbol's preferred rendering mode, but it's still a good idea to check the results for places where a different rendering mode might improve a symbol's legibility.

Variable color

With variable color, you can represent a characteristic that can change over time — like capacity or strength — regardless of rendering mode. To visually communicate such a change, variable color applies color to different layers of a symbol as a value reaches different thresholds between zero and 100 percent.

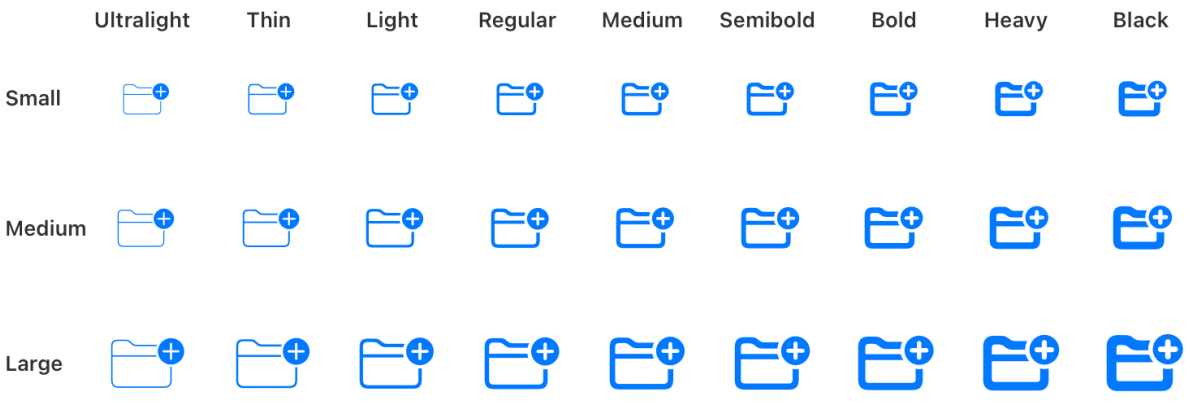
For example, you could use variable color with the `speaker.wave.3` symbol to communicate three different ranges of sound — plus the state where there's no sound — by mapping the layers that represent the curved wave paths to different ranges of decibel values. In the case of no sound, no wave layers get color. In all other cases, a wave layer receives color when the sound reaches a threshold the system defines based on the number of nonzero states you want to represent.

Sometimes, it can make sense for some of a symbol's layers to opt out of variable color. For example, in the `speaker.wave.3` symbol shown above, the layer that contains the speaker path doesn't receive variable color because a speaker doesn't change as the sound level changes. A symbol can support variable color in any number of layers.

Use variable color to communicate change — don't use it to communicate depth. To convey depth and visual hierarchy, use Hierarchical rendering mode to elevate certain layers and distinguish foreground and background elements in a symbol.

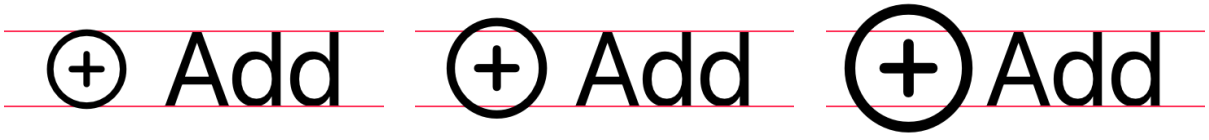
Weights and scales

SF Symbols provides symbols in a wide range of weights and scales to help you create adaptable designs.



Each of the nine symbol weights — from ultralight to black — corresponds to a weight of the San Francisco system font, helping you achieve precise weight matching between symbols and adjacent text, while supporting flexibility for different sizes and contexts.

Each symbol is also available in three scales: small, medium (the default), and large. The scales are defined relative to the cap height of the San Francisco system font.



Small Medium Large

Specifying a scale lets you adjust a symbol's emphasis compared to adjacent text, without disrupting the weight matching with text that uses the same point size. For developer guidance, see `imageScale( :)` (SwiftUI), `UIImage.SymbolScale` (UIKit), and `NSImage.SymbolConfiguration` (AppKit).

Design variants

SF Symbols defines several design variants — such as fill, slash, and enclosed — that can help you communicate precise states and actions while maintaining visual consistency and simplicity in your UI. For example, you could use the slash variant of a symbol to show that an item or action is unavailable, or use the fill variant to indicate selection.

Outline is the most common variant in SF Symbols. An outlined symbol has no solid areas, resembling the appearance of text. Most symbols are also available in a fill variant, in which the areas within some shapes are solid.

In addition to outline and fill, SF Symbols also defines variants that include a slash or enclose a symbol within a shape like a circle, square, or rectangle. In many cases, enclosed and slash variants can combine with outline or fill variants.

SF Symbols provides many variants for specific languages and writing systems, including Latin, Arabic, Hebrew, Hindi, Thai, Chinese, Japanese, Korean, Cyrillic, Devanagari, and several Indic numeral systems. Language- and script-specific variants adapt automatically when the device language changes. For guidance, see Images.

Symbol variants support a range of design goals. For example:

- The outline variant works well in toolbars, navigation bars, lists, and other places where you display a symbol alongside text.
- Symbols that use an enclosing shape — like a square or circle — can improve legibility at small sizes.
- The solid areas in a fill variant tend to give a symbol more visual emphasis, making it a good choice for iOS tab bars and swipe actions and places where you use an accent color to communicate selection.

In many cases, the view that displays a symbol determines whether to use outline or fill, so you don't have to specify a variant. For example, an iOS tab bar prefers the fill variant, whereas a navigation bar takes the outline variant.

Animations

SF Symbols provides a collection of expressive, configurable animations that enhance your interface and add vitality to your app. Symbol animations help communicate ideas, provide feedback in response to people's actions, and signal changes in status or ongoing activities.

Animations work on all SF Symbols in the library, in all rendering modes, weights, and scales, and on custom symbols. For considerations when animating custom symbols, see Custom symbols. You can control the playback of an animation, whether you want the animation to run from start to finish, or run indefinitely, repeating its effect until a condition is met. You can customize behaviors, like changing the playback speed of an animation or determining whether to reverse an animation before repeating it. For developer guidance, see Symbols and `Symbol Effect`.

SF Symbols 5 and later support the following animations.

Appear — Causes a symbol to gradually emerge into view.

Play

Disappear — Causes a symbol to gradually recede out of view.

Play

Bounce — Briefly scales a symbol with an elastic-like movement that goes either up or down and then returns to the symbol's initial state. The bounce animation plays once by default and can help communicate that an action occurred or needs to take place.

Play

Scale — Changes the size of a symbol, increasing or decreasing its scale. Unlike the bounce animation, which returns the symbol to its original state, the scale animation persists until you set a new scale or remove the effect. You might use the scale animation to draw people's attention to a selected item or as feedback when people choose a symbol.

Play

Pulse — Varies the opacity of a symbol over time. This animation automatically pulses only the layers within a symbol that are annotated to pulse, and optionally can pulse all layers within a symbol. You might use the pulse animation to communicate ongoing activity, playing it continuously until a condition is met.

Play

Variable color — Incrementally varies the opacity of layers within a symbol. This animation can be cumulative or iterative. When cumulative, color changes persist for each layer until the animation cycle is complete. When iterative, color changes occur one layer at a time. You might use variable color to communicate progress or ongoing activity, such as playback, connecting, or broadcasting. You can customize the animation to autoreverse — meaning reverse the animation to the starting point and replay the sequence — as well as hide inactive layers rather than reduce their opacity.

The arrangement of layers within a symbol determines how variable color behaves during a repeating animation. Symbols with layers that are arranged linearly where the start and end points don't meet are annotated as *open loop*. Symbols with layers that follow a complete shape where the start and end points do meet, like in a circular progress indicator, are annotated as *closed loop*.

In SF Symbols 6 and later, variable color animations for symbols with closed loop designs feature seamless, continuous playback.

Play ⊙

Replace — Replaces one symbol with another. The replace animation works between arbitrary symbols and across all weights and rendering modes. This animation features three configurations:

- Down-up, where the outgoing symbol scales down and the incoming symbol scales up, communicating a change in state.
- Up-up, where both the outgoing and incoming symbols scale up. This configuration communicates a change in state that includes a sense of forward progression.
- Off-up, where the outgoing symbol hides immediately and the incoming symbol scales up. This configuration communicates a state change that emphasizes the next available state or action.

Play ⊙

From left to right: down-up, up-up, off-up

Magic Replace — In SF Symbols 6 and later, performs a smart transition between two symbols with related shapes. For example, slashes can draw on and off, and badges can appear or disappear, or you can replace them independently of the base symbol. Magic Replace is the new default replace animation, but doesn't occur between unrelated symbols; the default down-up animation occurs instead. You can choose a custom direction for the fallback animation in these situations if you prefer one other than the default.

Play

Additionally, SF Symbols 6 and later support the following animations.

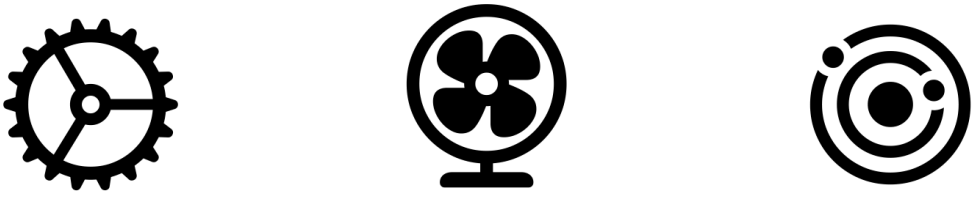
Wiggle — Moves the symbol back and forth along a directional axis. You might use the wiggle animation to highlight a change or a call to action that a person might overlook. Wiggle can also add a dynamic emphasis to an interaction or reinforce what the symbol is representing, such as when an arrow points in a specific direction.

Play

Breathe — Smoothly increases and decreases the presence of a symbol, giving it a living quality. You might use the breathe animation to convey status changes, or signal that an activity is taking place, like an ongoing recording session. Breathe is similar to pulse; however pulse animates by changing opacity alone, while breathe changes both opacity and size to convey ongoing activity.

Play

Rotate — Rotates the symbol to act as a visual indicator or imitate an object's behavior in the real world. For example, when a task is in progress, rotation confirms that it's working as expected. The rotate animation causes some symbols to rotate entirely, while in others only certain parts of the symbol rotate. Symbols like the desk fan, for example, use the By Layer rotation option to spin only the fan blades.



Play

Apply symbol animations judiciously. While there's no limit to how many animations you can add to a view, too many animations can overwhelm an interface and distract people.

Make sure that animations serve a clear purpose in communicating a symbol's intent. Each type of animation has a discrete movement that communicates a certain type of action or elicits a certain response. Consider how people might interpret an animated symbol and whether the animation, or combination of animations, might be confusing.

Use symbol animations to communicate information more efficiently. Animations provide visual feedback, reinforcing that something happened in your interface. You can use animations to present complex information in a simple way and without taking up a lot of visual space.

Consider your app's tone when adding animations. When animating a symbol, think about what the animation can convey and how that might align with your brand identity and your app's overall style and tone. For guidance, see Branding.

Custom symbols

If you need a symbol that SF Symbols doesn't provide, you can create your own. To create a custom symbol, first export the template for a symbol that's similar to the design you want, then use a vector-editing tool to modify it. For developer guidance, see Creating custom symbol images for your app.

> **Important**
>
> SF Symbols includes copyrighted symbols that depict Apple products and features. You can display these symbols in your app, but you can't customize them. To help you identify a noncustomizable symbol, the SF Symbols app badges it with an Info icon; to help you use the symbol correctly, the inspector pane describes its usage restrictions.

Using a process called *annotating*, you can assign a specific color — or a specific hierarchical level, such as primary, secondary, or tertiary — to each layer in a custom symbol. Depending on the rendering modes you support, you can use a different mode in each instance of the symbol in your app.

Use the template as a guide. Create a custom symbol that's consistent with the ones the system provides in level of detail, optical weight, alignment, position, and perspective. Strive to design a symbol that is:

- Simple
- Recognizable
- Inclusive
- Directly related to the action or content it represents

For guidance, see Icons.

Assign negative side margins to your custom symbol if necessary. SF Symbols supports negative side margins to aid optical horizontal alignment when a symbol contains a badge or other elements that increase its width. For example, negative side margins can help you horizontally align a stack of folder symbols, some of which include a badge. The name of each

margin includes the relevant configuration — such as “left-margin-Regular-M” — so be sure to use this naming pattern if you add margins to your custom symbols.

Optimize layers to use animations with custom symbols. If you want to animate your symbol by layer, make sure to annotate the layers in the SF Symbols app. The Z-order determines the order that you want to apply colors to the layers of a variable color symbol, and you can choose whether to animate those changes from front-to-back, or back-to-front. SF Symbols 5 introduces layer groups, which allows related layers to move together when animated.

Test animations for custom symbols. It’s important to test your custom symbols with all of the animation presets because the shapes and paths might not appear how you expect when the layers are in motion. To get the most out of this feature, consider drawing your custom symbols with whole shapes. For example, a custom symbol similar to the `person.2.fill` symbol doesn’t need to create a cutout for the shape representing the person on the left. Instead, you can draw the full shape of the person, and in addition to that, draw an offset path of the person on the right to help represent the gap between them. You can later annotate this offset path as an erase layer to render the symbol as you want. This method of drawing helps preserve additional layer information that allows for animations to perform as you expect.

Avoid making custom symbols that include common variants, such as enclosures or badges. The SF Symbols app offers a component library for creating variants of your custom symbol. Using the component library allows you to create commonly used variants of your custom symbol while maintaining design consistency with the included SF Symbols.

Provide alternative text labels for custom symbols. Alternative text labels — or accessibility descriptions — let VoiceOver describe visible UI and content, making navigation easier for people with visual disabilities. For guidance, see VoiceOver.

Don’t design replicas of Apple products. Apple products are copyrighted and you can’t reproduce them in your custom symbols. Also, you can’t customize a symbol that SF Symbols identifies as representing an Apple feature or product.

Platform considerations

No additional considerations for iOS, iPadOS, macOS, tvOS, visionOS, or watchOS.

Resources

Related

SF Symbols

Typography

Developer documentation

Configuring and displaying symbol images in your UI — UIKit

Creating custom symbol images for your app — UIKit

Symbols

Videos

What’s new in SF Symbols 6

What’s new in SF Symbols 5

Create animated symbols

Change log

Date	Changes
June 10, 2024	Updated with guidance for new animations and features of SF Symbols 6.
June 5, 2023	Added a new section on animations. Included animation guidance for custom symbols.
September 14, 2022	Added a new section on variable color. Removed instructions on creating custom symbol paths, exporting templates, and layering paths, deferring to developer articles that cover these topics.