

Steppers

On This Page

A stepper lets the user adjust a value by increasing and decreasing it in small steps. Steppers are used in situations where a user needs to adjust a value by a small amount.

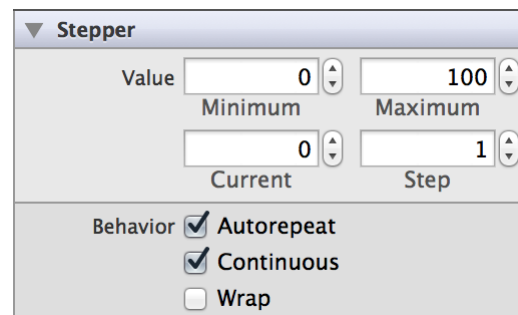


Purpose. Steppers allow users to:

- Make discrete and incremental adjustments to a value
- Have precise control over a value within a range

Implementation. Steppers are implemented in the [UIStepper](#) class. For API reference, see [UIStepper Class Reference](#).

Configuration. Configure steppers in Interface Builder, in the Stepper section of the Attributes Inspector. A few configurations cannot be made through the Attributes Inspector, so you must make them programmatically. You can set other configurations programmatically, too, if you prefer.

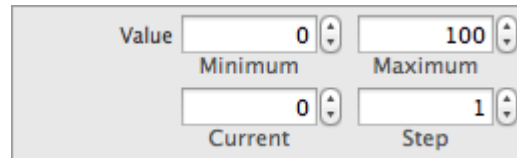
A screenshot of the 'Stepper' configuration panel from the Attributes Inspector in Xcode. The panel has a title bar with a dropdown arrow and the text 'Stepper'. Below the title bar, there are four numeric input fields with up/down arrows. The first two are labeled 'Value' (0) and 'Maximum' (100), with 'Minimum' written below the first. The next two are labeled 'Current' (0) and 'Step' (1). At the bottom, under the 'Behavior' section, there are three checkboxes: 'Autorepeat' (checked), 'Continuous' (checked), and 'Wrap' (unchecked).

Content of Steppers

Configure a minimum, maximum, and current value for the stepper by setting these properties in Interface Builder. By default, a stepper's minimum is set to 0, its maximum is set to 100, and its current value is set to 0. You can change these values by adjusting the Minimum ([minimumValue](#)), Maximum ([maximumValue](#)), and Current ([value](#)) fields.

On This Page

Steppers also allow you to specify step size, the amount by which the current value changes when the increase or decrease buttons are pressed. The default step size is 1. The corresponding Attributes Inspector field is called Step ([stepValue](#)).



Behavior of Steppers

Steppers do not need a [delegate](#) to function properly; their parent view controller can define their behavior without implementing any delegate protocols.

A stepper sends the [UIControlEventValueChanged](#) event when the user interacts it. You can respond to this event by performing some corresponding action in your app, such as adjusting music volume. You register the [target-action](#) methods for a page control as shown below.

```
1 [self.myStepper addTarget:self
2     action:@selector(myAction:)
3     forControlEvents:UIControlEventValueChanged];
```

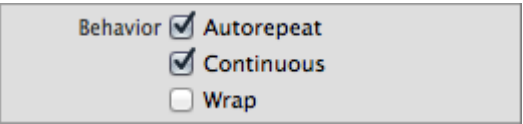
Alternatively, you can Control-drag the stepper's Value Changed event from the Connections Inspector to the action method. For more information, see [Target-Action Mechanism](#).

You can specify when a stepper's [UIControlEventValueChanged](#) events are sent by toggling the “Continuous” ([continuous](#)) checkbox in the Attributes Inspector. In continuous delivery, the stepper sends multiple Value Changed events while the user keeps pressing on the stepper. In noncontinuous delivery, the stepper sends one Value Changed event when the user releases the stepper. Continuous control event delivery is enabled by default.

A stepper defaults to autorepeat, which means that pressing and holding one of its buttons increments or decrements the stepper’s value repeatedly. The rate of change depends on how long the user continues pressing the control. The user can hold the stepper to quickly approach a desired value, and then increment or decrement to the desired value. Uncheck the “Autorepeat” ([autorepeat](#)) box if you want the stepper to be incremented or decremented only once.

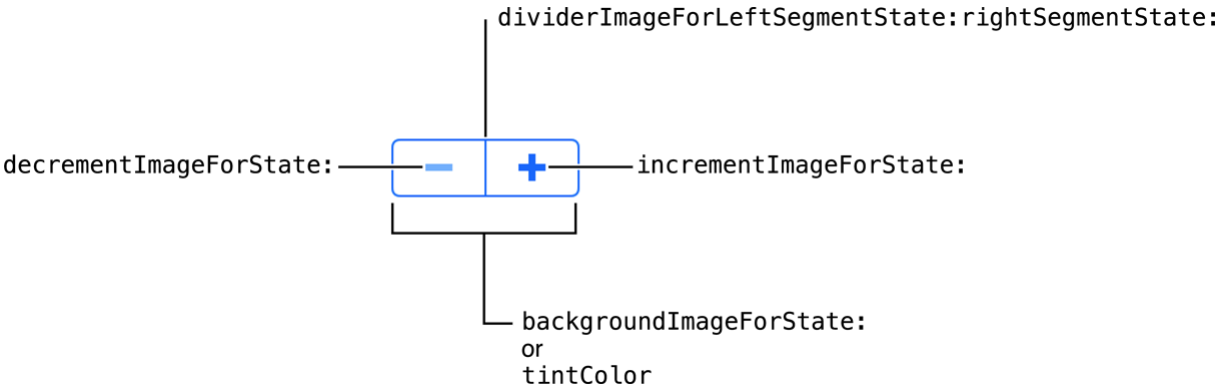
On This Page

You can set a stepper to wrap around to the minimum value when you try to increment it past its maximum—and vice versa. This functionality is disabled by default; to enable it, check the “Wrap” ([wraps](#)) box.



Appearance of Steppers

You can customize the appearance of a stepper by setting the properties depicted below.



To customize the appearance of all steppers in your app, use the appearance proxy (for example, `[UIStepper appearance]`). For more information about appearance proxies, see [Appearance Proxies](#).

Tint Color (Programmatic)

You can specify a custom stepper tint by setting the `tintColor` property programmatically. This property sets the color of the icons, divider, and border of the stepper. A translucent version of this color is also used to tint a stepper button when it is pressed down, as shown on the increment button below.

On This Page



If you do not explicitly set a tint color, the stepper will inherit its superview's tint color. For more information, see [Tint Color](#).

Icons

The increment and decrement images are the icons that sit on top of each stepper button. They appear on top of the background image. Use the `setDecrementImage:forState:` and `setIncrementImage:forState:` methods to specify custom increment and decrement images for each control state. Note that an increment or decrement image will be automatically rendered as a template image within a toolbar, unless you explicitly set its rendering mode to `UIImageRenderingModeAlwaysOriginal`. For more information, see [Template Images](#).

Background and Divider Images

You can set custom images for each `UIControlState` of a stepper. For more information about control states, see [Control States](#).

A stepper can have a background image that covers the entirety of the control except for the divider, filling the entire frame of the stepper. Use the `setBackgroundImageForState:` method to set a background image for each control state of the stepper.

To strengthen the visual effect of a stepper button being pressed, you can set custom divider images for different combinations of button states: increment button pressed, decrement button pressed, and neither pressed. Note that it is impossible to press both buttons at the same time. Use the `setDividerImage:forLeftSegmentState:rightSegmentState:` method to specify custom divider images. Don't forget to set an image for every state.

Using Auto Layout with Steppers

You can create Auto Layout constraints between a stepper and other user interface elements. You can create any type of constraint for a stepper besides a baseline constraint.

For general information about using Auto Layout with iOS controls, see [Using Auto Layout with Controls](#).

On This Page

Making Steppers Accessible

Steppers are accessible by default. The default accessibility traits for a stepper are User Interaction Enabled and Adjustable.

For general information about making iOS controls accessible, see [Making Controls Accessible](#).

Internationalizing Steppers

Steppers have no special properties related to internationalization. However, if you use a stepper with a label, make sure you provide localized strings for the label.

For more information, see [Internationalization and Localization Guide](#).

Elements Similar to a Stepper

The following elements provide similar functionality to a stepper:

- **Slider.** Use sliders to adjust a value continuously, rather than in discrete steps. Sliders are more appropriate than steppers for setting a value that has a large range. For more information, see [Sliders](#).
- **Picker View.** Use pickers to let the user select one of a list of options, rather than stepping through the range of a value. A picker is more appropriate when selecting from a fixed set of options—for example, choosing a month. For more information, see [Picker Views](#).