

---

# NSButton Class Reference

User Experience: Controls



2007-04-01



Apple Inc.  
© 2007 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

## **NSButton Class Reference 5**

---

Overview	5
Tasks	6
Configuring Buttons	6
Configuring Button Images	6
Managing Button State	7
Accessing Key Equivalents	7
Handling Keyboard Events	8
Instance Methods	8
allowsMixedState	8
alternateImage	8
alternateTitle	9
attributedAlternateTitle	9
attributedTitle	10
bezelStyle	10
getPeriodicDelay:interval:	10
highlight:	11
image	11
imagePosition	12
isBordered	12
isTransparent	13
keyEquivalent	13
keyEquivalentModifierMask	13
performKeyEquivalent:	14
setAllowsMixedState:	14
setAlternateImage:	15
setAlternateTitle:	15
setAttributedAlternateTitle:	16
setAttributedTitle:	16
setBezelStyle:	17
setBordered:	17
setButtonType:	18
setImage:	18
setImagePosition:	19
setKeyEquivalent:	19
setKeyEquivalentModifierMask:	20
setNextState	20
setPeriodicDelay:interval:	21
setShowsBorderOnlyWhileMouseInside:	21
setSound:	22
setState:	22

setTitle: 23  
setTitleWithMnemonic: 24  
setTransparent: 24  
showsBorderOnlyWhileMouseInside 25  
sound 25  
state 25  
title 26

**Document Revision History 27**

---

**Index 29**

---

# NSButton Class Reference

---

<b>Inherits from</b>	NSControl : NSView : NSResponder : NSObject
<b>Conforms to</b>	NSUserInterfaceValidations NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Button Programming Topics for Cocoa
<b>Declared in</b>	NSButton.h
<b>Related sample code</b>	CocoaSpeechSynthesisExample FunHouse MyPhoto PDF Annotation Editor WhackedTV

## Overview

The `NSButton` class is a subclass of `NSControl` that intercepts mouse-down events and sends an action message to a target object when it's clicked or pressed.

The `NSButton` class uses `NSButtonCell` to implement its user interface.

`NSButton` and `NSMatrix` both provide a control view, which is needed to display an `NSButtonCell` object. However, while `NSMatrix` requires you to access the `NSButtonCell` objects directly, most of the `NSButton` class' methods are "covers" for identically declared methods in `NSButtonCell`. (In other words, the implementation of the `NSButton` method invokes the corresponding `NSButtonCell` method for you, allowing you to be unconcerned with the existence of the `NSButtonCell`.) The only `NSButtonCell` methods that don't have covers relate to the font used to display the key equivalent and to specific methods for highlighting or showing the state of the `NSButton` (these last are usually set together with the `NSButton` [setButtonType:](#) (page 18) method).

## Tasks

### Configuring Buttons

- [setButtonType:](#) (page 18)  
Sets how the receiver button highlights while pressed and how it shows its state.
- [getPeriodicDelay:interval:](#) (page 10)  
Returns by reference the delay and interval periods for a continuous button.
- [setPeriodicDelay:interval:](#) (page 21)  
Sets the message delay and interval periods for a continuous button.
- [alternateTitle](#) (page 9)  
Returns the title that the button displays when it's in its alternate state.
- [setAlternateTitle:](#) (page 15)  
Sets the title that appears on the button when it's in its alternate state.
- [attributedTitle](#) (page 10)  
Returns the title that the button displays in its normal state as an attributed string.
- [setAttributedTitle:](#) (page 16)  
Sets the string that appears on the button when it's in its normal state to the given attributed string and redraws the button.
- [attributedAlternateTitle](#) (page 9)  
Returns the title that the button displays when it's in its alternate state as an attributed string.
- [setAttributedAlternateTitle:](#) (page 16)  
Sets the title that appears on the button when it's in its alternate state to the given attributed string.
- [title](#) (page 26)  
Returns the title displayed on the button when it's in its normal state.
- [setTitle:](#) (page 23)  
Sets the title displayed by the receiver when in its normal state and, if necessary, redraws the button's contents.
- [setTitleWithMnemonic:](#) (page 24)  
Sets the title of a button with a character denoting an access key.
- [setSound:](#) (page 22)  
Sets the sound played when the user presses the button.
- [sound](#) (page 25)  
Returns the sound that's played when the user presses the button.

### Configuring Button Images

- [image](#) (page 11)  
Returns the image that appears on the receiver when it's in its normal state.
- [setImage:](#) (page 18)  
Sets the receiver's image and redraws the button.
- [alternateImage](#) (page 8)  
Returns the image that appears on the button when it's in its alternate state.

- [setImage:](#) (page 15)  
Sets the image displayed by the button when it's in its alternate state and, if necessary, redraws the contents of the button.
- [imagePosition](#) (page 12)  
Returns the position of the receiver's image relative to its title.
- [setImagePosition:](#) (page 19)  
Sets the position of the button's image relative to its title.
- [isBordered](#) (page 12)  
Returns a Boolean value indicating whether the button has a border.
- [setBordered:](#) (page 17)  
Sets whether the receiver has a beveled border.
- [isTransparent](#) (page 13)  
Returns a Boolean value indicating whether the button is transparent.
- [setTransparent:](#) (page 24)  
Sets whether the receiver is transparent and redraws the receiver if necessary.
- [bezelStyle](#) (page 10)  
Returns the appearance of the receiver's border.
- [setBezelStyle:](#) (page 17)  
Sets the appearance of the border, if the receiver has one.
- [showsBorderOnlyWhileMouseInside](#) (page 25)  
Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.
- [setShowsBorderOnlyWhileMouseInside:](#) (page 21)  
Sets whether the receiver's border is displayed only when the cursor is over the button.

## Managing Button State

- [allowsMixedState](#) (page 8)  
Returns a Boolean value indicating whether the button allows a mixed state.
- [setAllowsMixedState:](#) (page 14)  
Sets whether the button allows a mixed state.
- [state](#) (page 25)  
Returns the receiver's state.
- [setState:](#) (page 22)  
Sets the cell's state to the specified value.
- [setNextState](#) (page 20)  
Sets the receiver to its next state.
- [highlight:](#) (page 11)  
Highlights (or unhighlights) the receiver.

## Accessing Key Equivalents

- [keyEquivalent](#) (page 13)  
Returns the key-equivalent character of the receiver.

- [setKeyEquivalent:](#) (page 19)  
Sets the key equivalent character of the receiver to the given character.
- [keyEquivalentModifierMask](#) (page 13)  
Returns the mask specifying the modifier keys for the receiver's key equivalent.
- [setKeyEquivalentModifierMask:](#) (page 20)  
Sets the mask indicating the modifier keys used by the receiver's key equivalent.

## Handling Keyboard Events

- [performKeyEquivalent:](#) (page 14)  
Checks the button's key equivalent against the specified event and, if they match, simulates the button being clicked.

## Instance Methods

### allowsMixedState

Returns a Boolean value indicating whether the button allows a mixed state.

- (BOOL)allowsMixedState

#### Return Value

YES if the receiver has three states: on, off, and mixed. NO if the receiver has two states: on and off. The default is NO.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setAllowsMixedState:](#) (page 14)
- [setNextState](#) (page 20)

#### Declared In

NSButton.h

### alternateImage

Returns the image that appears on the button when it's in its alternate state.

- (NSImage \*)alternateImage

#### Return Value

The image displayed by the button when it's in its alternate state, or `nil` if there is no alternate image. Note that some button types don't display an alternate image. Buttons don't display images by default.

#### Availability

Available in Mac OS X v10.0 and later.

**See Also**

- [setImage:](#) (page 15)
- [image](#) (page 11)
- [imagePosition](#) (page 12)
- [keyEquivalent](#) (page 13)
- [setButtonType:](#) (page 18)

**Declared In**

NSButton.h

**alternateTitle**

Returns the title that the button displays when it's in its alternate state.

```
- (NSString *)alternateTitle
```

**Return Value**

The string that appears on the receiver when it's in its alternate state, or the empty string if the receiver doesn't display an alternate title. By default, a button's alternate title is "Button."

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setAlternateTitle:](#) (page 15)
- [attributedAlternateTitle](#) (page 9)
- [setButtonType:](#) (page 18)
- [title](#) (page 26)

**Declared In**

NSButton.h

**attributedAlternateTitle**

Returns the title that the button displays when it's in its alternate state as an attributed string.

```
- (NSAttributedString *)attributedAlternateTitle
```

**Return Value**

The string that appears on the receiver when it's in its alternate state, as an `NSAttributedString`, or the empty string if the receiver doesn't display an alternate title. By default, a button's alternate title is "Button."

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setAttributedAlternateTitle:](#) (page 16)
- [attributedTitle](#) (page 10)
- [setButtonType:](#) (page 18)

**Declared In**  
NSButton.h

## attributedString

Returns the title that the button displays in its normal state as an attributed string.

```
- (NSAttributedString *)attributedString
```

### Return Value

The string that appears on the receiver when it's in its normal state as an `NSAttributedString`, or an empty attributed string if the receiver doesn't display a title.

A button's title is always displayed if the button doesn't use its alternate contents for highlighting or displaying the alternate state. By default, a button's title is "Button."

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setAttributeString:](#) (page 16)
- [attributedStringAlternateString](#) (page 9)
- [setButtonType:](#) (page 18)

**Declared In**  
NSButton.h

## bezelStyle

Returns the appearance of the receiver's border.

```
- (NSBezelStyle)bezelStyle
```

### Return Value

The bezel style of the button. See the "Constants" section of `NSButtonCell` for the list of possible values.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setBezelStyle:](#) (page 17)

**Declared In**  
NSButton.h

## getPeriodicDelay:interval:

Returns by reference the delay and interval periods for a continuous button.

```
- (void)getPeriodicDelay:(float *)delay interval:(float *)interval
```

**Parameters***delay*

On return, the amount of time (in seconds) the button will pause before starting to periodically send action messages to the target object. The default delay is taken from a user's default (60 seconds maximum). If the user hasn't specified a default value, *delay* defaults to 0.4 seconds,

*interval*

On return, the amount of time (in seconds) between each action message that is sent. The default interval is taken from a user's default (60 seconds maximum). If the user hasn't specified a default value, *interval* defaults to 0.075 seconds.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [isContinuous](#) (NSControl)

**Declared In**

NSButton.h

**highlight:**

Highlights (or unhighlights) the receiver.

- (void)highlight:(BOOL)flag

**Parameters***flag*

YES to highlight the button; NO to unhighlight the button. If the current state of the button matches *flag*, no action is taken.

**Discussion**

Highlighting may involve the button appearing “pushed in” to the screen, displaying its alternate title or image, or causing the button to appear to be “lit.”

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setButtonType:](#) (page 18)

**Declared In**

NSButton.h

**image**

Returns the image that appears on the receiver when it's in its normal state.

- (NSImage \*)image

**Return Value**

The image displayed by the button when it's in its normal state, or *nil* if there is no such image. This image is always displayed on a button that doesn't change its contents when highlighting or showing its alternate state. Buttons don't display images by default.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setImage:](#) (page 18)
- [alternateImage](#) (page 8)
- [setButtonType:](#) (page 18)

**Declared In**

NSButton.h

**imagePosition**

Returns the position of the receiver's image relative to its title.

- (NSCellImagePosition)imagePosition

**Return Value**

The position of the button's image. This is one of the image positions described in the “Constants” section of NSCell.

**Discussion**

If the title is above, below, or overlapping the image, or if there is no image, the text is horizontally centered within the button.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setImagePosition:](#) (page 19)
- [setButtonType:](#) (page 18)
- [setImage:](#) (page 18)
- [setTitle:](#) (page 23)

**Declared In**

NSButton.h

**isBordered**

Returns a Boolean value indicating whether the button has a border.

- (BOOL)isBordered

**Return Value**

YES if the receiver has a border, NO otherwise. A button's border isn't the single line of most other controls' borders—instead, it's a raised bezel. By default, buttons are bordered.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setBordered:](#) (page 17)

**Declared In**

NSButton.h

**isTransparent**

Returns a Boolean value indicating whether the button is transparent.

- (BOOL)isTransparent

**Return Value**

YES if the receiver is transparent, NO otherwise. A transparent button never draws itself, but it receives mouse-down events and tracks the mouse properly.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setTransparent:](#) (page 24)

**Declared In**

NSButton.h

**keyEquivalent**

Returns the key-equivalent character of the receiver.

- (NSString \*)keyEquivalent

**Return Value**

The button's key equivalent, or the empty string if one hasn't been defined. Buttons don't have a default key equivalent.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setKeyEquivalent:](#) (page 19)
- [performKeyEquivalent:](#) (page 14)
- `keyEquivalentFont` (NSButtonCell)

**Declared In**

NSButton.h

**keyEquivalentModifierMask**

Returns the mask specifying the modifier keys for the receiver's key equivalent.

- (NSUInteger)keyEquivalentModifierMask

**Return Value**

The mask specifying the modifier keys that are applied to the button's key equivalent. Mask bits are defined in `NSEvent.h`. The only mask bits relevant in button key-equivalent modifier masks are `NSControlKeyMask`, `NSAlternateKeyMask`, and `NSCommandKeyMask`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setKeyEquivalentModifierMask:](#) (page 20)
- [keyEquivalent](#) (page 13)

**Declared In**

`NSButton.h`

**performKeyEquivalent:**

Checks the button's key equivalent against the specified event and, if they match, simulates the button being clicked.

```
- (BOOL)performKeyEquivalent:(NSEvent *)anEvent
```

**Parameters**

*anEvent*

The event containing the key equivalent.

**Return Value**

YES if the key equivalent in *anEvent* matches the button's key equivalent; NO if it does not. This method also returns NO if the receiver is blocked by a modal panel or the button is disabled.

**Discussion**

If the character in *anEvent* matches the receiver's key equivalent, and the modifier flags in *anEvent* match the key-equivalent modifier mask, `performKeyEquivalent:` simulates the user clicking the button and returning YES. Otherwise, `performKeyEquivalent:` does nothing and returns NO.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [keyEquivalent](#) (page 13)
- [keyEquivalentModifierMask](#) (page 13)

**Declared In**

`NSButton.h`

**setAllowsMixedState:**

Sets whether the button allows a mixed state.

```
- (void)setAllowsMixedState:(BOOL)flag
```

**Parameters***flag*

YES to indicate that the receiver has three states: on, off, and mixed. If *flag* is NO, the receiver has two states: on and off.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [allowsMixedState](#) (page 8)
- [setNextState](#) (page 20)

**Related Sample Code**

EnhancedAudioBurn

Quartz Composer WWDC 2005 TextEdit

**Declared In**

NSButton.h

**setAlternateImage:**

Sets the image displayed by the button when it's in its alternate state and, if necessary, redraws the contents of the button.

```
- (void)setAlternateImage:(NSImage *)image
```

**Parameters***image*

The image that appears on the receiver when it's in its alternate state. Note that some button types don't display an alternate image.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [alternateImage](#) (page 8)
- [setButtonType:](#) (page 18)
- [setImage:](#) (page 18)

**Declared In**

NSButton.h

**setAlternateTitle:**

Sets the title that appears on the button when it's in its alternate state.

```
- (void)setAlternateTitle:(NSString *)aString
```

**Parameters***aString*

The string to set as the button's alternate title. Note that some button types don't display an alternate title.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [alternateTitle](#) (page 9)
- [setTitle:](#) (page 23)
- [setTitleWithMnemonic:](#) (page 24)
- [setButtonType:](#) (page 18)
- [setFont:](#) (NSButtonCell)

**Declared In**

NSButton.h

**setAttributedAlternateTitle:**

Sets the title that appears on the button when it's in its alternate state to the given attributed string.

```
- (void)setAttributedAlternateTitle:(NSAttributedString *)aString
```

**Parameters**

*aString*

The attributed string to set as the button's alternate title. Note that some button types don't display an alternate title.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [attributedAlternateTitle](#) (page 9)
- [setAttributedTitle:](#) (page 16)
- [setButtonType:](#) (page 18)
- [setFont:](#) (NSButtonCell)

**Declared In**

NSButton.h

**setAttributedTitle:**

Sets the string that appears on the button when it's in its normal state to the given attributed string and redraws the button.

```
- (void)setAttributedTitle:(NSAttributedString *)aString
```

**Parameters**

*aString*

The attributed string to set as the button's title. The title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

**Discussion****Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [attributedString](#) (page 10)
- [setAttributedStringAlternateTitle:](#) (page 16)
- [setButtonType:](#) (page 18)
- [setFont:](#) (NSButtonCell)

**Declared In**

NSButton.h

**setBezelStyle:**

Sets the appearance of the border, if the receiver has one.

```
- (void)setBezelStyle:(NSBezelStyle)bezelStyle
```

**Parameters***bezelStyle*

The bezel style of the button. This must be one of the bezel styles described in the “Constants” section of NSButtonCell.

If the button is not bordered, the bezel style is ignored.

**Discussion**

The button uses shading to look like it’s sticking out or pushed in. You can set the shading with the NSButtonCell method `setGradientType:`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [bezelStyle](#) (page 10)

**Related Sample Code**

FunHouse

**Declared In**

NSButton.h

**setBordered:**

Sets whether the receiver has a beveled border.

```
- (void)setBordered:(BOOL)flag
```

**Parameters***flag*

YES if the receiver should display a border; NO if it should not. A button’s border is not the single line of most other controls’ borders—instead, it’s a raised bezel.

**Discussion**

This method redraws the button if `setBordered:` causes the bordered state to change.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [isBordered](#) (page 12)

**Related Sample Code**

FunHouse

**Declared In**

NSButton.h

**setButtonType:**

Sets how the receiver button highlights while pressed and how it shows its state.

```
- (void)setButtonType:(NSButtonType)aType
```

**Parameters**

*aType*

A constant specifying the type of the button—one of the constants described in the Constants section of `NSButtonCell`.

**Discussion**

`setButtonType:` redisplay the button before returning.

The types available are for the most common button types, which are also accessible in Interface Builder. You can configure different behavior with the `NSButtonCell` methods `setHighlightsBy:` and `setShowsStateBy:`.

Note that there is no `-buttonType` method. The `set` method sets various button properties that together establish the behavior of the type.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setAlternateImage:](#) (page 15)
- [setImage:](#) (page 18)
- `setButtonType: (NSButtonCell)`

**Related Sample Code**

FunHouse

**Declared In**

NSButton.h

**setImage:**

Sets the receiver's image and redraws the button.

```
- (void)setImage:(NSImage *)anImage
```

**Parameters***anImage*

The button's image. A button's image is displayed when the button is in its normal state, or all the time for a button that doesn't change its contents when highlighting or displaying its alternate state.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [image](#) (page 11)
- [setImagePosition:](#) (page 19)
- [setAlternateImage:](#) (page 15)
- [setButtonType:](#) (page 18)

**Related Sample Code**

FunHouse

**Declared In**

NSButton.h

**setImagePosition:**

Sets the position of the button's image relative to its title.

```
- (void)setImagePosition:(NSCellImagePosition)aPosition
```

**Parameters***aPosition*

A constant specifying the position of the button's image. See the "Constants" section of NSCell for a listing of possible values.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [imagePosition](#) (page 12)

**Related Sample Code**

ButtonMadness

FunHouse

**Declared In**

NSButton.h

**setKeyEquivalent:**

Sets the key equivalent character of the receiver to the given character.

```
- (void)setKeyEquivalent:(NSString *)charCode
```

**Parameters***charCode*

The character to set as the button's key equivalent.

**Discussion**

This method redraws the button's interior if it displays a key equivalent instead of an image. The key equivalent isn't displayed if the image position is set to `NSNoImage`, `NSImageOnly`, or `NSImageOverlaps`; that is, the button must display both its title and its "image" (the key equivalent in this case), and they must not overlap.

To display a key equivalent on a button, set the image and alternate image to `nil`, then set the key equivalent, then set the image position.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [keyEquivalent](#) (page 13)
- [performKeyEquivalent:](#) (page 14)
- [setAlternateImage:](#) (page 15)
- [setImage:](#) (page 18)
- [setImagePosition:](#) (page 19)
- [setKeyEquivalentFont:](#) (NSButtonCell)

**Declared In**

NSButton.h

**setKeyEquivalentModifierMask:**

Sets the mask indicating the modifier keys used by the receiver's key equivalent.

```
- (void)setKeyEquivalentModifierMask:(NSUInteger)mask
```

**Parameters***mask*

The mask identifying the modifier keys to be applied to the button's key equivalent.

Mask bits are defined in `NSEvent.h`. The only mask bits relevant in button key-equivalent modifier masks are `NSControlKeyMask`, `NSAlternateKeyMask`, and `NSCommandKeyMask`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [keyEquivalentModifierMask](#) (page 13)
- [setKeyEquivalent:](#) (page 19)

**Declared In**

NSButton.h

**setNextState**

Sets the receiver to its next state.

- (void)setNextState

#### Discussion

If the button has three states, it cycles through them in this order: on, off, mixed, on, and so forth. If the button has two states, it toggles between them.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [allowsMixedState](#) (page 8)
- [setAllowsMixedState:](#) (page 14)

#### Declared In

NSButton.h

## setPeriodicDelay:interval:

Sets the message delay and interval periods for a continuous button.

- (void)setPeriodicDelay:(float)delay interval:(float)interval

#### Parameters

*delay*

The amount of time (in seconds) that a continuous button will pause before starting to periodically send action messages to the target object. The maximum allowed value is 60.0 seconds; if a larger value is supplied, it is ignored, and 60.0 seconds is used.

*interval*

The amount of time (in seconds) between each action message. The maximum value is 60.0 seconds; if a larger value is supplied, it is ignored, and 60.0 seconds is used.

#### Discussion

The delay and interval values are used if the button is configured (by a `setContinuous:` message) to continuously send the action message to the target object while tracking the mouse.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- `setContinuous:(NSControl)`

#### Declared In

NSButton.h

## setShowsBorderOnlyWhileMouseInside:

Sets whether the receiver's border is displayed only when the cursor is over the button.

- (void)setShowsBorderOnlyWhileMouseInside:(BOOL)show

**Parameters***show*

YES to display the border only when the cursor is within the button's border and the button is active.  
 NO, to continue to display the button's border when the cursor is outside the button's bounds.

**Discussion**

If [isBordered](#) (page 12) returns NO, the border is never displayed, regardless of what this method returns.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [showsBorderOnlyWhileMouseInside](#) (page 25)

**Declared In**

NSButton.h

**setSound:**

Sets the sound played when the user presses the button.

```
- (void)setSound:(NSSound *)aSound
```

**Parameters***aSound*

The sound that should be played when the user presses the button. The sound is played during a mouse-down event, such as `NSLeftMouseDown`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [sound](#) (page 25)

**Declared In**

NSButton.h

**setState:**

Sets the cell's state to the specified value.

```
- (void)setState:(NSInteger)value
```

**Parameters***value*

The state of the button. This can be `NSOnState`, `NSOffState`, `NSMixedState`. See the discussion for a more detailed explanation.

**Discussion**

If necessary, this method also redraws the receiver.

The cell can have two or three states. If it has two, *value* can be `NSOffState` (the normal or unpressed state) and `NSOnState` (the alternate or pressed state). If it has three, *value* can be `NSOnState` (the feature is in effect everywhere), `NSOffState` (the feature is in effect nowhere), or `NSMixedState` (the feature is in effect somewhere). Note that if the cell has only two states and *value* is `NSMixedState`, this method sets the cell's state to `NSOnState`.

Although using the enumerated constants is preferred, *value* can also be an integer. If the cell has two states, 0 is treated as `NSOffState`, and a nonzero value is treated as `NSOnState`. If the cell has three states, 0 is treated as `NSOffState`; a negative value, as `NSMixedState`; and a positive value, as `NSOnState`.

To check whether the button uses the mixed state, use the method [allowsMixedState](#) (page 8).

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [state](#) (page 25)

#### Related Sample Code

EnhancedAudioBurn

FunHouse

QTAudioContextInsert

Sketch-112

WhackedTV

#### Declared In

NSButton.h

## setTitle:

Sets the title displayed by the receiver when in its normal state and, if necessary, redraws the button's contents.

```
- (void)setTitle:(NSString *)aString
```

#### Parameters

*aString*

The string to set as the button's title. This title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [title](#) (page 26)

- [setAlternateTitle:](#) (page 15)

- [setButtonType:](#) (page 18)

- [setTitleWithMnemonic:](#) (page 24)

- [setFont:](#) (NSButtonCell)

#### Related Sample Code

ExtractMovieAudioToAIFF

FunHouse

QTAudioContextInsert  
 SharedMemory  
 WhackedTV

**Declared In**  
 NSButton.h

### setTitleWithMnemonic:

Sets the title of a button with a character denoting an access key.

```
- (void)setTitleWithMnemonic:(NSString *)aString
```

#### Discussion

Mnemonics are not supported in Mac OS X.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [title](#) (page 26)
- [setAlternateTitle:](#) (page 15)
- [setButtonType:](#) (page 18)
- [setTitle:](#) (page 23)
- [setFont:](#) (NSButtonCell)

**Declared In**  
 NSButton.h

### setTransparent:

Sets whether the receiver is transparent and redraws the receiver if necessary.

```
- (void)setTransparent:(BOOL)flag
```

#### Parameters

*flag*

YES if the button is transparent; otherwise NO.

#### Discussion

A transparent button tracks the mouse and sends its action, but doesn't draw. A transparent button is useful for sensitizing an area on the screen so that an action gets sent to a target when the area receives a mouse click.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [isTransparent](#) (page 13)

**Declared In**  
 NSButton.h

## showsBorderOnlyWhileMouseInside

Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.

- (BOOL)showsBorderOnlyWhileMouseInside

### Return Value

YES if the receiver's border is displayed only when the cursor is over the button and the button is active; NO if the border is displayed all the time.

By default, this method returns NO.

### Discussion

If [isBordered](#) (page 12) returns NO, the border is never displayed, regardless of what this method returns.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setShowsBorderOnlyWhileMouseInside:](#) (page 21)

### Declared In

NSButton.h

## sound

Returns the sound that's played when the user presses the button.

- (NSSound \*)sound

### Return Value

The sound played when the user presses the button.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setSound:](#) (page 22)

### Declared In

NSButton.h

## state

Returns the receiver's state.

- (NSInteger)state

### Return Value

The button's state. A button can have two or three states. If it has two, this value is either `NSOffState` (the normal or unpressed state) or `NSOnState` (the alternate or pressed state). If it has three, this value can be `NSOnState` (the feature is in effect everywhere), `NSOffState` (the feature is in effect nowhere), or `NSMixedState` (the feature is in effect somewhere).

**Discussion**

To check whether the button uses the mixed state, use the method [allowsMixedState](#) (page 8).

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setState:](#) (page 22)

**Related Sample Code**

DatePicker

DragNDropOutlineView

FinalCutPro\_AppleEvents

GLUT

WhackedTV

**Declared In**

NSButton.h

**title**

Returns the title displayed on the button when it's in its normal state.

```
- (NSString *)title
```

**Return Value**

The title displayed on the receiver when it's in its normal state or the empty string if the button doesn't display a title. This title is always displayed if the button doesn't use its alternate contents for highlighting or displaying the alternate state. By default, a button's title is "Button."

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [alternateTitle](#) (page 9)

- [setButtonType:](#) (page 18)

- [setTitle:](#) (page 23)

- [setTitleWithMnemonic:](#) (page 24)

**Related Sample Code**

SpeedometerView

**Declared In**

NSButton.h

# Document Revision History

---

This table describes the changes to *NSButton Class Reference*.

Date	Notes
2007-04-01	Made editorial improvements.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

allowsMixedState [instance method 8](#)  
alternateImage [instance method 8](#)  
alternateTitle [instance method 9](#)  
attributedAlternateTitle [instance method 9](#)  
attributedTitle [instance method 10](#)

## B

---

bezelStyle [instance method 10](#)

## G

---

getPeriodicDelay:interval: [instance method 10](#)

## H

---

highlight: [instance method 11](#)

## I

---

image [instance method 11](#)  
imagePosition [instance method 12](#)  
isBordered [instance method 12](#)  
isTransparent [instance method 13](#)

## K

---

keyEquivalent [instance method 13](#)  
keyEquivalentModifierMask [instance method 13](#)

## P

---

performKeyEquivalent: [instance method 14](#)

## S

---

setAllowsMixedState: [instance method 14](#)  
setAlternateImage: [instance method 15](#)  
setAlternateTitle: [instance method 15](#)  
setAttributedAlternateTitle: [instance method 16](#)  
setAttributedTitle: [instance method 16](#)  
setBezelStyle: [instance method 17](#)  
setBordered: [instance method 17](#)  
setButtonType: [instance method 18](#)  
setImage: [instance method 18](#)  
setImagePosition: [instance method 19](#)  
setKeyEquivalent: [instance method 19](#)  
setKeyEquivalentModifierMask: [instance method 20](#)  
setNextState [instance method 20](#)  
setPeriodicDelay:interval: [instance method 21](#)  
setShowsBorderOnlyWhileMouseInside: [instance method 21](#)  
setSound: [instance method 22](#)  
setState: [instance method 22](#)  
setTitle: [instance method 23](#)  
setTitleWithMnemonic: [instance method 24](#)  
setTransparent: [instance method 24](#)  
showsBorderOnlyWhileMouseInside [instance method 25](#)  
sound [instance method 25](#)  
state [instance method 25](#)

## T

---

title [instance method 26](#)