NSButtonCell Class Reference

Cocoa > User Experience

2009-04-08

Contents

NSButtonCell Class Reference

Inherits from	NSActionCell : NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Button Programming Topics for Cocoa
Declared in	NSButtonCell.h
Related sample code	FunkyOverlayWindow

Overview

The `NSButtonCell` class is a subclass of `NSActionCell` used to implement the user interfaces of push buttons, checkboxes (switches), and radio buttons. It can also be used for any other region of a view that's designed to send a message to a target when clicked. The `NSButton` subclass of `NSControl` uses a single `NSButtonCell`.

The `NSButtonCell` class implements the user interface of `NSButton`.

Setting the integer, float, double, or object value of an `NSButtonCell` object results in a call to `setState:` with the value converted to integer. In the case of `setObjectValue:`, `nil` is equivalent to 0, and a non-`nil` object that doesn't respond to `intValue` sets the state to 1. Otherwise, the state is set to the object's `intValue`. Similarly, querying the integer, float, double, or object value of an `NSButtonCell` returns the current state in the requested representation. In the case of `objectValue`, this is an `NSNumber` containing `YES` for on, `NO` for off, and integer value -1 for the mixed state.

For more information on the behavior of `NSButtonCell`, see the `NSButton` and `NSMatrix` class specifications, and *Button Programming Topics for Cocoa*.

Exceptions

In its implementation of the `compare:` method (declared in `NSCell`), `NSButtonCell` raises an `NSBadComparisonException` if the *`otherCell`* argument is not of the `NSButtonCell` class.

Tasks

Setting Titles

- `alternateMnemonic` (page 10)
  Returns the character in the alternate title that's marked as the "keyboard mnemonic."
- `alternateMnemonicLocation` (page 10)
  Returns an unsigned integer indicating the character in the alternate title that's marked as the "keyboard mnemonic."
- `alternateTitle` (page 11)
  Returns the string displayed by the button when it's in its alternate state.
- `attributedAlternateTitle` (page 11)
  Returns the title displayed by the button when it's in its alternate state, as an attributed string.
- `attributedTitle` (page 12)
  Returns the title displayed by the button when it's in its normal state as an attributed string.
- `setAlternateMnemonicLocation:` (page 21)
  Sets the character in the alternate title that should be the "keyboard mnemonic."
- `setAlternateTitle:` (page 21)
  Sets the title the button displays when it's in its alternate state.
- `setAlternateTitleWithMnemonic:` (page 22)
  Sets the title the button displays when it's in its alternate state to the given string with an embedded mnemonic.
- `setAttributedAlternateTitle:` (page 22)
  Sets the string the button displays when it's in its alternate state to the given attributed string.
- `setAttributedTitle:` (page 23)
  Sets the string the button displays when it's in its normal state to the given attributed string and redraws the button.
- `setFont:` (page 25)
  Sets the font used to display the button's title and alternate title.
- `setTitle:` (page 32)
  Sets the title the button displays when in its normal state and, if necessary, redraws the receiver's contents.
- `setTitleWithMnemonic:` (page 32)
  Sets the title the button displays when it's in its normal state to the given string with an embedded mnemonic.
- `title` (page 34)
  Returns the title displayed on the receiver when it's in its normal state.

Managing Images

- `alternateImage` (page 9)
  Returns the image the button displays in its alternate state.

- `imagePosition` (page 16)
  Returns the position of the receiver’s image relative to its title.
- `setAlternateImage:` (page 20)
  Sets the image the button displays in its alternate state and, if necessary, redraws its contents.
- `setImagePosition:` (page 27)
  Sets the position of the receiver’s image relative to its title.
- `imageScaling` (page 17)
  Returns the scale factor for the receiver’s image.
- `setImageScaling:` (page 27)
  Sets the scale factor for the receiver’s image.

Managing the Repeat Interval

- `getPeriodicDelay:interval:` (page 14)
  Returns by reference the delay and interval periods for a continuous button.
- `setPeriodicDelay:interval:` (page 30)
  Sets the message delay and interval for the receiver.

Managing the Key Equivalent

- `keyEquivalent` (page 18)
  Returns the receiver's key-equivalent character.
- `keyEquivalentFont` (page 18)
  Returns the font used to draw the key equivalent.
- `keyEquivalentModifierMask` (page 19)
  Returns the mask identifying the modifier keys for the button's key equivalent.
- `setKeyEquivalent:` (page 28)
  Sets the key equivalent character of the receiver.
- `setKeyEquivalentModifierMask:` (page 29)
  Sets the mask identifying the modifier keys to use with the button's key equivalent.
- `setKeyEquivalentFont:` (page 28)
  Sets the font used to draw the key equivalent and redisplays the receiver if necessary.
- `setKeyEquivalentFont:size:` (page 29)
  Sets by name and size of the font used to draw the key equivalent.

Managing Graphics Attributes

- `backgroundColor` (page 12)
  Returns the background color of the receiver.
- `setBackgroundColor:` (page 23)
  Sets the background color of the receiver.
- `bezelStyle` (page 12)
  Returns the appearance of the receiver’s border.

- `setBezelStyle:` (page 24)
  Sets the appearance of the border, if the receiver has one.
- `gradientType` (page 15)
  Returns the gradient of the receiver's border.
- `setGradientType:` (page 26)
  Sets the type of gradient to use for the receiver.
- `imageDimsWhenDisabled` (page 16)
  Returns a Boolean value that indicates whether the receiver's image and text appear "dim" when the receiver is disabled.
- `setImageDimsWhenDisabled:` (page 27)
  Sets whether the receiver's image appears "dim" when the button cell is disabled.
- `isOpaque` (page 17)
  Returns a Boolean value that indicates whether the receiver is opaque.
- `isTransparent` (page 17)
  Returns a Boolean value that indicates whether the receiver is transparent.
- `setTransparent:` (page 33)
  Sets whether the receiver is transparent.
- `showsBorderOnlyWhileMouseInside` (page 33)
  Returns a Boolean value indicating whether the button displays its border only when the cursor is over it.
- `setShowsBorderOnlyWhileMouseInside:` (page 30)
  Sets whether the receiver's border is displayed only when the cursor is over the button.

Displaying the Cell

- `highlightsBy` (page 15)
  Returns flags indicating how the button highlights when it receives a mouse-down event.
- `setHighlightsBy:` (page 26)
  Sets the way the receiver highlights itself while pressed.
- `setShowsStateBy:` (page 31)
  Sets the way the receiver indicates its alternate state.
- `setButtonType:` (page 24)
  Sets how the receiver highlights while pressed and how it shows its state.
- `showsStateBy` (page 33)
  Returns the flags indicating how the button cell shows its alternate state.

Managing the Sound

- `sound` (page 34)
  Returns the sound that's played when the user presses the receiver.
- `setSound:` (page 31)
  Sets the sound that's played when the user presses the receiver.

Handling Events and Action Messages

- `mouseEntered:` (page 19)
  Draws the receiver's border.
- `mouseExited:` (page 20)
  Erases the receiver's border.
- `performClick:` (page 20)
  Simulates the user clicking the receiver with the cursor.

Drawing the Button Content

- `drawBezelWithFrame:inView:` (page 13)
  Draws the border of the button using the current bezel style.
- `drawImage:withFrame:inView:` (page 13)
  Draws the image associated with the button's current state.
- `drawTitle:withFrame:inView:` (page 14)
  Draws the button's title centered vertically in a specified rectangle.

Instance Methods

alternateImage

Returns the image the button displays 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.

Discussion
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
- `setAlternateImage:` (page 20)
- `imagePosition` (page 16)
- `keyEquivalent` (page 18)
- `setButtonType:` (page 24)
- `image` (`NSCell`)

Declared In
`NSButtonCell.h`

alternateMnemonic

Returns the character in the alternate title that’s marked as the “keyboard mnemonic.”

```
- (NSString *)alternateMnemonic
```

Return Value
The character in the alternate title (the title displayed on the receiver when it's in its alternate state) marked as the "keyboard mnemonic."

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alternateMnemonicLocation` (page 10)
- `setAlternateTitleWithMnemonic:` (page 22)
- `mnemonic` (`NSCell`)

Declared In
`NSButtonCell.h`

alternateMnemonicLocation

Returns an unsigned integer indicating the character in the alternate title that’s marked as the “keyboard mnemonic.”

```
- (NSUInteger)alternateMnemonicLocation
```

Return Value
An unsigned integer indicating the character in the alternate title (the title displayed on the receiver when it’s in its alternate state) that’s marked as the “keyboard mnemonic.” If the alternate title doesn’t have a keyboard mnemonic, returns `NSNotFound`.

Discussion
Mnemonics are not supported in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setAlternateMnemonicLocation:` (page 21)
- `alternateMnemonic` (page 10)
- `setAlternateTitleWithMnemonic:` (page 22)
- `mnemonicLocation` (`NSCell`)

Declared In
`NSButtonCell.h`

alternateTitle

Returns the string displayed by the button when it’s in its alternate state.

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

Return Value
The string that appears on the button when it's in its alternate state, or the empty string if the receiver doesn’t display an alternate title.

Discussion
Note that some button types don’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 21)
`- alternateMnemonic` (page 10)
`- attributedAlternateTitle` (page 11)
`- setButtonType:` (page 24)
`- title` (page 34)

Declared In
`NSButtonCell.h`

attributedAlternateTitle

Returns the title displayed by the button when it’s in its alternate state, as an attributed string.

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

Return Value
The attributed string that appears on the button when it's in its alternate state, or the empty string if the receiver doesn’t display an alternate title.

Discussion
Note that some button types don’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 22)
`- alternateMnemonic` (page 10)
`- attributedTitle` (page 12)
`- setButtonType:` (page 24)

Declared In
`NSButtonCell.h`

attributedTitle

Returns the title displayed by the button when it’s in its normal state as an attributed string.

```
- (NSAttributedString *)attributedTitle
```

Return Value
The attributes string that appears on the button when it's in its normal state, or an empty attributed string if the receiver doesn’t display a title.

Discussion
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
`– setAttributedTitle:` (page 23)
`– attributedAlternateTitle` (page 11)
`– setButtonType:` (page 24)
`– mnemonic` (NSCell)

Declared In
`NSButtonCell.h`

backgroundColor

Returns the background color of the receiver.

```
- (NSColor *)backgroundColor
```

Return Value
The receiver’s background color.

Discussion
The background color is used only when drawing borderless buttons.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– setBackgroundColor:` (page 23)

Declared In
`NSButtonCell.h`

bezelStyle

Returns the appearance of the receiver’s border.

```
- (NSBezelStyle)bezelStyle
```

Return Value
A constant specifying the bezel style used by the button. See "Bezel Styles" (page 35) for a list of possible values.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setBezelStyle:` (page 24)

Declared In
`NSButtonCell.h`

drawBezelWithFrame:inView:

Draws the border of the button using the current bezel style.

```
- (void)drawBezelWithFrame:(NSRect)frame inView:(NSView *)controlView
```

Parameters

`frame`
: The bounding rectangle of the button.

`controlView`
: The control being drawn.

Discussion
This method is called automatically when the button is redrawn; you should not call it directly.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setBezelStyle:` (page 24)

Declared In
`NSButtonCell.h`

drawImage:withFrame:inView:

Draws the image associated with the button's current state.

```
- (void)drawImage:(NSImage *)image withFrame:(NSRect)frame inView:(NSView
    *)controlView
```

Parameters

`image`
: The image associated with the button's current state.

`frame`
: The bounding rectangle of the button.

`controlView`
: The control being drawn.

Discussion
This method is called automatically when the button is redrawn; you should not call it directly.

You specify the primary and alternate images for the button using Interface Builder.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAlternateImage:` (page 20)

Declared In
`NSButtonCell.h`

drawTitle:withFrame:inView:

Draws the button's title centered vertically in a specified rectangle.

```
- (NSRect)drawTitle:(NSAttributedString *)title withFrame:(NSRect)frame
    inView:(NSView *)controlView
```

Parameters

`title`
: The title of the button.

`frame`
: The rectangle in which to draw the title.

`controlView`
: The control being drawn.

Return Value
The bounding rectangle for the text of the title.

Discussion
This method is called automatically when the button is redrawn; you should not call it directly.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAlternateTitle:` (page 21)
`- setAttributedTitle:` (page 23)

Declared In
`NSButtonCell.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) that the button will pause before starting to periodically send action messages to the target object. Default values are taken from the user's defaults (60 seconds maximum); if the user hasn't specified a default value, this defaults to 0.4 seconds.

interval

On return, the amount of time (in seconds) between each action message. Default values are taken from the user's defaults (60 seconds maximum); if the user hasn't specified a default value, this defaults to 0.075 seconds.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- isContinuous`

`- isContinuous` (`NSCell`)

Declared In

`NSButtonCell.h`

gradientType

Returns the gradient of the receiver’s border.

```
- (NSGradientType)gradientType
```

Return Value

A constant specifying the gradient used for the button's border. See “Gradient Types” (page 40) for a list of possible values.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setGradientType:` (page 26)

Declared In

`NSButtonCell.h`

highlightsBy

Returns flags indicating how the button highlights when it receives a mouse-down event.

```
- (NSInteger)highlightsBy
```

Return Value

The logical OR of flags that indicate the way the receiver highlights when it receivers a mouse-down event. See the “Constants” section of NSCell for the list of flags.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setHighlightsBy:` (page 26)

`- showsStateBy` (page 33)

Declared In

`NSButtonCell.h`

imageDimsWhenDisabled

Returns a Boolean value that indicates whether the receiver’s image and text appear “dim” when the receiver is disabled.

```
- (BOOL)imageDimsWhenDisabled
```

Return Value

`YES` if the button's image and text are dimmed when the button is disabled, otherwise `NO`.

Discussion

By default, all button types except `NSSwitchButton` and `NSRadioButton` do dim when disabled. When buttons of type `NSSwitchButton` and `NSRadioButton` are disabled, only the associated text dims.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setButtonType:` (page 24)

`- setImageDimsWhenDisabled:` (page 27)

Declared In

`NSButtonCell.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 27)

`- setButtonType:` (page 24)

`- setTitle:` (page 32)

`- setImage: (NSCell)`

Declared In
`NSButtonCell.h`

imageScaling

Returns the scale factor for the receiver's image.

```
- (NSImageScaling)imageScaling
```

Return Value
The scale factor for the receiver's image.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSButtonCell.h`

isOpaque

Returns a Boolean value that indicates whether the receiver is opaque.

```
- (BOOL)isOpaque
```

Return Value
`YES` if the receiver draws over every pixel in its frame, otherwise `NO`.

Discussion
A button cell is opaque only if it isn't transparent and if it has a border.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- isTransparent` (page 17)
`- setTransparent:` (page 33)

Declared In
`NSButtonCell.h`

isTransparent

Returns a Boolean value that indicates whether the receiver is transparent.

```
- (BOOL)isTransparent
```

Return Value
`YES` if the receiver is transparent, `NO` otherwise.

Discussion
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 33)
`– isOpaque` (page 17)

Declared In
`NSButtonCell.h`

keyEquivalent

Returns the receiver's key-equivalent character.

```
- (NSString *)keyEquivalent
```

Return Value
The string containing the key equivalent character of the button, or the empty string if one hasn't been defined.

Discussion
Buttons don't have a default key equivalent.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– setKeyEquivalent:` (page 28)
`– keyEquivalentFont` (page 18)

Declared In
`NSButtonCell.h`

keyEquivalentFont

Returns the font used to draw the key equivalent.

```
- (NSFont *)keyEquivalentFont
```

Return Value
The font object describing the font used to draw the button's key equivalent, or `nil` if the receiver doesn’t have a key equivalent.

Discussion
The default font is the same as that used to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also

- `setKeyEquivalentFont:` (page 28)
- `setKeyEquivalentFont:size:` (page 29)
- `setFont:` (page 25)

Declared In

`NSButtonCell.h`

keyEquivalentModifierMask

Returns the mask identifying the modifier keys for the button's key equivalent.

```
- (NSUInteger)keyEquivalentModifierMask
```

Return Value

A mask indicating the modifier keys that are applied to the receiver'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` bits.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setKeyEquivalentModifierMask:` (page 29)
- `keyEquivalent` (page 18)

Declared In

`NSButtonCell.h`

mouseEntered:

Draws the receiver's border.

```
- (void)mouseEntered:(NSEvent *)event
```

Parameters

event

The event object generated by the mouse movement.

Discussion

This method is called only when the cursor moves onto the receiver and `showsBorderOnlyWhileMouseInside` (page 33) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSButtonCell.h`

mouseExited:

Erases the receiver's border.

```
- (void)mouseExited:(NSEvent *)event
```

Parameters

event

The event object generated by the mouse movement.

Discussion

This method is called only when the cursor moves off the receiver and `showsBorderOnlyWhileMouseInside` (page 33) returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSButtonCell.h`

performClick:

Simulates the user clicking the receiver with the cursor.

```
- (void)performClick:(id)sender
```

Parameters

sender

The sender of the message.

Discussion

This method essentially highlights the button, sends the button's action message to the target object, and then unhighlights the button. If an exception is raised while the target object is processing the action message, the button is unhighlighted before the exception is propagated out of `performClick:`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSButtonCell.h`

setAlternateImage:

Sets the image the button displays in its alternate state and, if necessary, redraws its contents.

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

Parameters

image

The image displayed by the button when it's in its alternate state.

Discussion

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 9)
- `setButtonType:` (page 24)
- `setImage:` (`NSCell`)

Declared In
`NSButtonCell.h`

setAlternateMnemonicLocation:

Sets the character in the alternate title that should be the "keyboard mnemonic."

```
- (void)setAlternateMnemonicLocation:(NSUInteger)location
```

Parameters

location
An unsigned integer indicating the character in the alternate title that should be marked as the "keyboard mnemonic." If you don't want the alternate title to have a keyboard mnemonic, specify a location of `NSNotFound`.

Discussion
Mnemonics are not supported in Mac OS X.

The `setAlternateMnemonicLocation:` method doesn't cause the button cell to be redisplayed.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `alternateMnemonicLocation` (page 10)
- `setAlternateTitleWithMnemonic:` (page 22)

Declared In
`NSButtonCell.h`

setAlternateTitle:

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

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

Parameters

aString
The string to set as the button's title when it's in its alternate state.

Discussion
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 11)
`- setAlternateMnemonicLocation:` (page 21)
`- setAlternateTitleWithMnemonic:` (page 22)
`- setTitle:` (page 32)
`- setButtonType:` (page 24)
`- setFont:` (page 25)

Declared In

`NSButtonCell.h`

setAlternateTitleWithMnemonic:

Sets the title the button displays when it's in its alternate state to the given string with an embedded mnemonic.

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

Parameters

aString
The string to set as the button's alternate title, taking into account the fact that an embedded "&" character is not a literal but instead marks the alternate state's "keyboard mnemonic."

Discussion

Mnemonics are not supported in Mac OS X.

If necessary, `setAlternateTitleWithMnemonic:` redraws the button cell. Note that some button types don't display an alternate title.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setAlternateMnemonicLocation:` (page 21)
`- setTitleWithMnemonic:` (page 32)

Declared In

`NSButtonCell.h`

setAttributedAlternateTitle:

Sets the string the button displays 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.

Discussion

Note that some button types don't display an alternate title.

Graphics attributes that are set on the cell (`backgroundColor`, `alignment`, `font`, etc.) are overriden when corresponding properties are set for the attributed string.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attributedAlternateTitle` (page 11)
- `setAlternateMnemonicLocation:` (page 21)
- `setAlternateTitleWithMnemonic:` (page 22)
- `setAttributedTitle:` (page 23)
- `setButtonType:` (page 24)
- `setFont:` (page 25)

Declared In
`NSButtonCell.h`

setAttributedTitle:

Sets the string the button displays 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.

Discussion
The title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Graphics attributes configured for the cell (`backgroundColor`, `alignment`, `font`, etc.) are overriden when corresponding properties are set for the attributed string.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `attributedTitle` (page 12)
- `setAttributedAlternateTitle:` (page 22)
- `setButtonType:` (page 24)
- `setFont:` (page 25)
- `setMnemonicLocation:` (NSCell)

Declared In
`NSButtonCell.h`

setBackgroundColor:

Sets the background color of the receiver.

```
- (void)setBackgroundColor:(NSColor *)color
```

Parameters

color

The color to use for the receiver's background.

Discussion

The background color is used only when drawing borderless buttons.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- backgroundColor` (page 12)

Declared In

`NSButtonCell.h`

setBezelStyle:

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

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

Parameters

bezelStyle

A constant specifying the bezel style to use for the button. This must be one of the values specified in "Bezel Styles" (page 35).

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

Discussion

A button uses shading to look like it's sticking out or pushed in. You can set the shading with `setGradientType:` (page 26).

Availability

Available in Mac OS X v10.0 and later.

See Also

`- bezelStyle` (page 12)

Declared In

`NSButtonCell.h`

setButtonType:

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

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

Parameters

aType

A constant specifying the type of button. This can be one of the constants defined in "Button Types" (page 37).

Discussion

`setButtonType:` redisplays the receiver 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 `setHighlightsBy:` (page 26) and `setShowsStateBy:` (page 31) methods.

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 20)
- `setImage:` (NSCell)

Declared In

`NSButtonCell.h`

setFont:

Sets the font used to display the button's title and alternate title.

```
- (void)setFont:(NSFont *)fontObj
```

Parameters

fontObj

The font object specifying the font to use.

Discussion

This method does nothing if the receiver has no title or alternate title.

If the button cell has a key equivalent, its font is not changed, but the key equivalent’s font size is changed to match the new title font.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `setKeyEquivalentFont:` (page 28)
- `setKeyEquivalentFont:size:` (page 29)
- `font` (NSCell)

Related Sample Code

QTAudioExtractionPanel

Quartz Composer WWDC 2005 TextEdit

TextEditPlus

Declared In

`NSButtonCell.h`

setGradientType:

Sets the type of gradient to use for the receiver.

```
- (void)setGradientType:(NSGradientType)gradientType
```

Parameters

gradientType

A constant specifying the gradient to use for the button's border. This can be one of the constants defined in "Gradient Types" (page 40).

Discussion

If the receiver has no border, this method has no effect on its appearance. A concave gradient is darkest in the top-left corner; a convex gradient is darkest in the bottom-right corner. Weak versus strong is how much contrast exists between the colors used in opposite corners.

Note: This method is currently unused by the Application Kit and has no effect.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `gradientType` (page 15)

Declared In

`NSButtonCell.h`

setHighlightsBy:

Sets the way the receiver highlights itself while pressed.

```
- (void)setHighlightsBy:(NSInteger)aType
```

Parameters

aType

The logical OR of one or more of the cell masks described in the "Constants" section of NSCell.

Discussion

If both `NSChangeGrayCellMask` and `NSChangeBackgroundCellMask` are specified, both are recorded, but which behavior is used depends on the button cell's image. If the button has no image, or if the image has no alpha (transparency) data, `NSChangeGrayCellMask` is used. If the image does have alpha data, `NSChangeBackgroundCellMask` is used; this arrangement allows the color swap of the background to show through the image's transparent pixels.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `highlightsBy` (page 15)
- `setShowsStateBy:` (page 31)

Declared In

`NSButtonCell.h`

setImageDimsWhenDisabled:

Sets whether the receiver's image appears "dim" when the button cell is disabled.

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

Parameters

flag

`YES` to indicate that the button's image should dim when the button is disabled.

Discussion

By default, all button types except `NSSwitchButton` and `NSRadioButton` do dim when disabled. When `NSSwitchButton`s and `NSRadioButton`s are disabled, only the associated text dims. The default setting for this condition is reasserted whenever you invoke `setButtonType:` (page 24), so be sure to specify the button cell's type before you invoke `setImageDimsWhenDisabled:`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- imageDimsWhenDisabled` (page 16)

Declared In

`NSButtonCell.h`

setImagePosition:

Sets the position of the receiver'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 16)

Related Sample Code

FunkyOverlayWindow

Declared In

`NSButtonCell.h`

setImageScaling:

Sets the scale factor for the receiver's image.

```
- (void)setImageScaling:(NSImageScaling)scaling
```

Parameters

`scaling`
The scale factor for the receiver's image.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSButtonCell.h`

setKeyEquivalent:

Sets the key equivalent character of the receiver.

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

Parameters

`aKeyEquivalent`
The key equivalent character.

Discussion

This method redraws the receiver's inside 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 18)
- `– setAlternateImage:` (page 20)
- `– setImagePosition:` (page 27)
- `– setKeyEquivalentFont:` (page 28)
- `– setImage:` (NSCell)

Declared In

`NSButtonCell.h`

setKeyEquivalentFont:

Sets the font used to draw the key equivalent and redisplays the receiver if necessary.

```
- (void)setKeyEquivalentFont:(NSFont *)fontObj
```

Parameters

`fontObj`
The font object specifying the font to use for the receiver's key equivalent.

Discussion

This method does nothing if the receiver doesn't have a key equivalent associated with it.

The default font is the same as that used to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalentFont` (page 18)
- `setFont:` (page 25)

Declared In
`NSButtonCell.h`

setKeyEquivalentFont:size:

Sets by name and size of the font used to draw the key equivalent.

```
- (void)setKeyEquivalentFont:(NSString *)fontName size:(CGFloat)fontSize
```

Parameters

fontName
    The name of the font to use to draw the key equivalent.

fontSize
    The font size to use to draw the key equivalent.

Discussion
This method redisplays the receiver if necessary. It does nothing if the receiver doesn’t have a key equivalent associated with it. The default font is the same as that used to draw the title.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `keyEquivalentFont` (page 18)
- `setFont:` (page 25)

Declared In
`NSButtonCell.h`

setKeyEquivalentModifierMask:

Sets the mask identifying the modifier keys to use with the button's key equivalent.

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

Parameters

mask
    The mask indicating the modifier keys to be applied to the receiver'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 19)
- `setKeyEquivalent:` (page 28)

Declared In

`NSButtonCell.h`

setPeriodicDelay:interval:

Sets the message delay and interval for the receiver.

```
- (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 value is 60.0 seconds; if a larger value is supplied, it's 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's ignored, and 60.0 seconds is used.

Discussion

These values are used if the receiver 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:` (NSCell)

Declared In

`NSButtonCell.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 button's border only when the cursor is within the receiver's border and the button is active. `NO` to continue to display the border when the cursor is outside button's bounds.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `showsBorderOnlyWhileMouseInside` (page 33)

Declared In
`NSButtonCell.h`

setShowsStateBy:

Sets the way the receiver indicates its alternate state.

```
- (void)setShowsStateBy:(NSInteger)aType
```

Parameters

aType
The logical `OR` of one or more of the cell masks described in the "Constants" section of NSCell.

Discussion
If both `NSChangeGrayCellMask` and `NSChangeBackgroundCellMask` are specified, both are recorded, but the actual behavior depends on the button cell's image. If the button has no image, or if the image has no alpha (transparency) data, `NSChangeGrayCellMask` is used. If the image exists and has alpha data, `NSChangeBackgroundCellMask` is used; this arrangement allows the color swap of the background to show through the image's transparent pixels.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– setHighlightsBy:` (page 26)
`– showsStateBy` (page 33)

Declared In
`NSButtonCell.h`

setSound:

Sets the sound that's played when the user presses the receiver.

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

Parameters

aSound
The sound to play when the button is pressed.

Discussion
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 34)

Declared In
`NSButtonCell.h`

setTitle:

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

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

Parameters

aString

The string to set as the button's title.

Discussion

The 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 34)
- `- setAlternateTitle:` (page 21)
- `- setButtonType:` (page 24)
- `- setFont:` (page 25)
- `- setTitleWithMnemonic:` (page 32)

Declared In

`NSButtonCell.h`

setTitleWithMnemonic:

Sets the title the button displays when it's in its normal state to the given string with an embedded mnemonic.

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

Parameters

aString

The string to set as the button's title, taking into account the fact that an embedded "&" character is not a literal but instead marks the alternate state's "keyboard mnemonic." This title is always shown on buttons that don't use their alternate contents when highlighting or displaying their alternate state.

Discussion

If necessary, `setTitleWithMnemonic:` redraws the button cell. Mnemonics are not supported in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `- setAlternateTitleWithMnemonic:` (page 22)
- `- setTitleWithMnemonic:` (NSCell)
- `- setMnemonicLocation:` (NSCell)

Declared In

`NSButtonCell.h`

setTransparent:

Sets whether the receiver is transparent.

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

Parameters

flag

YES to make the button cell transparent.

Discussion

This method redraws the receiver if necessary. 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 17)
`- isOpaque` (page 17)

Declared In

`NSButtonCell.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.

Discussion

By default, this method returns `NO`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setShowsBorderOnlyWhileMouseInside:` (page 30)

Declared In

`NSButtonCell.h`

showsStateBy

Returns the flags indicating how the button cell shows its alternate state.

```
- (NSInteger)showsStateBy
```

Return Value
The logical OR of flags that indicate the way the receiver shows its alternate state. See the "Constants" section of NSCell for the list of flags.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- highlightsBy` (page 15)
`- setShowsStateBy:` (page 31)

Declared In
`NSButtonCell.h`

sound

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

```
- (NSSound *)sound
```

Return Value
The sound played when the receiver is pressed.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setSound:` (page 31)

Declared In
`NSButtonCell.h`

title

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

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

Return Value
The title displayed by the button in its normal state, or the empty string if the button doesn't display a title.

Discussion
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
`- setTitle:` (page 32)
`- alternateTitle` (page 11)
`- setButtonType:` (page 24)
`- mnemonic` (`NSCell`)

```
- mnemonicLocation (NSCell)
```

Declared In
`NSButtonCell.h`

Constants

NSBezelStyle

Type to define bezel styles.

```
typedef NSUInteger NSBezelStyle;
```

Discussion
For possible values, see "Bezel Styles" (page 35).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSButtonCell.h`

Bezel Styles

Define the bezel styles used by `bezelStyle` (page 12) and `setBezelStyle:` (page 24).

```
enum {
    NSRoundedBezelStyle           = 1,
    NSRegularSquareBezelStyle     = 2,
    NSThickSquareBezelStyle       = 3,
    NSThickerSquareBezelStyle     = 4,
    NSDisclosureBezelStyle        = 5,
    NSShadowlessSquareBezelStyle  = 6,
    NSCircularBezelStyle          = 7,
    NSTexturedSquareBezelStyle    = 8,
    NSHelpButtonBezelStyle        = 9,
    NSSmallSquareBezelStyle       = 10,
    NSTexturedRoundedBezelStyle   = 11,
    NSRoundRectBezelStyle         = 12,
    NSRecessedBezelStyle          = 13,
    NSRoundedDisclosureBezelStyle = 14,
}
```

Constants
`NSRoundedBezelStyle`

A rounded rectangle button, designed for text.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSRegularSquareBezelStyle`
A rectangular button with a 2 point border, designed for icons.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSThickSquareBezelStyle`
A rectangular button with a 3 point border, designed for icons.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSThickerSquareBezelStyle`
A rectangular button with a 4 point border, designed for icons.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSDisclosureBezelStyle`
A bezel style for use with a disclosure triangle.

To create the disclosure triangle, set the button bezel style to `NSDisclosureBezelStyle` and the button type to `NSOnOffButton`.

Available in Mac OS X v10.3 and later.

Declared in `NSButtonCell.h`.

`NSShadowlessSquareBezelStyle`
Similar to `NSRegularSquareBezelStyle`, but has no shadow so you can abut the cells without overlapping shadows.

This style would be used in a tool palette, for example.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSCircularBezelStyle`
A round button with room for a small icon or a single character.

This style has both regular and small variants, but the large variant is available only in gray at this time.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSTexturedSquareBezelStyle`
A bezel style appropriate for use with textured (metal) windows.

Available in Mac OS X v10.3 and later.

Declared in `NSButtonCell.h`.

`NSHelpButtonBezelStyle`
A round button with a question mark providing the standard help button look.

Available in Mac OS X v10.3 and later.

Declared in `NSButtonCell.h`.

`NSSmallSquareBezelStyle`
A simple square bezel style. Buttons using this style can be scaled to any size.

Available in Mac OS X v10.4 and later.

Declared in `NSButtonCell.h`.

`NSTexturedRoundedBezelStyle`
: A textured (metal) bezel style similar in appearance to the Finder's action (gear) button.

  The height of this button is fixed.

  Available in Mac OS X v10.4 and later.

  Declared in `NSButtonCell.h`.

`NSRoundRectBezelStyle`
: A bezel style that matches the search buttons in Finder and Mail.

  Available in Mac OS X v10.4 and later.

  Declared in `NSButtonCell.h`.

`NSRecessedBezelStyle`
: A bezel style that matches the recessed buttons in Mail, Finder and Safari.

  Available in Mac OS X v10.4 and later.

  Declared in `NSButtonCell.h`.

`NSRoundedDisclosureBezelStyle`
: A bezel style that matches the disclosure style used in the standard Save panel.

  Available in Mac OS X v10.4 and later.

  Declared in `NSButtonCell.h`.

Discussion
For examples of how these styles are displayed, see *Button Programming Topics for Cocoa*.

Declared In
`NSButtonCell.h`

NSButtonType

Type to define button types.

```
typedef NSUInteger NSButtonType;
```

Discussion
For possible values, see "Button Types" (page 37).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSButtonCell.h`

Button Types

Represent the button types that can be specified using `setButtonType:` (page 24).

```
enum {
    NSMomentaryLightButton   = 0,
    NSPushOnPushOffButton    = 1,
    NSToggleButton           = 2,
    NSSwitchButton           = 3,
    NSRadioButton            = 4,
    NSMomentaryChangeButton  = 5,
    NSOnOffButton            = 6,
    NSMomentaryPushInButton  = 7,
    NSMomentaryPushButton    = 0,
    NSMomentaryLight         = 7
};
```

Constants

`NSMomentaryLightButton`

While the button is held down it's shown as "lit," and also "pushed in" to the screen if the button is bordered.

This type of button is best for simply triggering actions, as it doesn't show its state; it always displays its normal image or title. This option is called "Momentary Light" in Interface Builder's Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSPushOnPushOffButton`

The first click both highlights and causes the button to be "pushed in" if the button is bordered; a second click returns it to its normal state.

This option is called "Push On Push Off" in Interface Builder's Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSToggleButton`

After the first click, the button displays its alternate image or title; a second click returns the button to its normal state.

This option is called "Toggle" in Interface Builder's Button Inspector.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSSwitchButton`

This style is a variant of `NSToggleButton` that has no border and is used to represent a checkbox.

This type of button is available as a separate Library item in Interface Builder.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSRadioButton`

This style is similar to `NSSwitchButton`, but it used to constrain a selection to a single element from several.

You typically use this type of button in a group formed by an instance of `NSMatrix`. In Interface Builder, a matrix of this type of button is available as a separate Library item.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSMomentaryChangeButton`
: While the button is held down, the alternate image and alternate title are displayed.

  The normal image and title are displayed when the button isn't pressed. This option is called "Momentary Change" in Interface Builder's Button Inspector.

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

`NSOnOffButton`
: The first click highlights the button; a second click returns it to the normal (unhighlighted) state.

  This option is called "On Off" in Interface Builder's Button Inspector.

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

`NSMomentaryPushInButton`
: While the button is held down it's shown as "lit."

  This type of button is best for simply triggering actions, as it doesn't show its state; it always displays its normal image or title. This option is called "Momentary Push In" in Interface Builder's Button Inspector.

  This button type is the default.

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

`NSMomentaryPushButton`
: While the button is held down it's shown as "lit," and also "pushed in" to the screen if the button is bordered. (**Deprecated.** Use `NSMomentaryLightButton` instead.)

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

`NSMomentaryLight`
: While the button is held down it's shown as "lit." (**Deprecated.** Use `NSMomentaryPushInButton` instead.)

  Available in Mac OS X v10.0 and later.

  Declared in `NSButtonCell.h`.

Discussion
For examples of how these types behave, see *Button Programming Topics for Cocoa*.

Declared In
`NSButtonCell.h`

NSGradientType

Type to define gradient types.

```
typedef NSUInteger NSGradientType;
```

Discussion
For possible values, see "Gradient Types" (page 40).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSButtonCell.h`

Gradient Types

Specify the gradients used by `gradientType` (page 15) and `setGradientType:` (page 26).

```
typedef enum _NSGradientType {
    NSGradientNone          = 0,
    NSGradientConcaveWeak   = 1,
    NSGradientConcaveStrong = 2,
    NSGradientConvexWeak    = 3,
    NSGradientConvexStrong  = 4
} NSGradientType;
```

Constants

`NSGradientNone`

There is no gradient, so the button looks flat.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSGradientConcaveWeak`

The top-left corner is light gray, and the bottom-right corner is dark gray, so the button appears to be pushed in.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSGradientConcaveStrong`

As with `NSGradientConcaveWeak`, the top-left corner is light gray, and the bottom-right corner is dark gray, but the difference between the grays is greater, so the appearance of being pushed in is stronger.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSGradientConvexWeak`

The top-left corner is dark gray, and the bottom-right corner is light gray, so the button appears to be sticking out.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

`NSGradientConvexStrong`

As with `NSGradientConvexWeak`, the top-left corner is dark gray, and the bottom-right corner is light gray, but the difference between the grays is greater, so the appearance of sticking out is stronger.

Available in Mac OS X v10.0 and later.

Declared in `NSButtonCell.h`.

Declared In
`NSButtonCell.h`

Document Revision History

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

Date	Notes
2009-04-08	Added style override details to setAttributedTitle:.
2008-10-15	Noted that setGradientType: has no effect.
2007-12-11	Noted that the API uses "switch" to describe a checkbox.
2007-01-05	Updated to include API introduced in Mac OS X v10.5.
2006-06-28	Corrected the declaration of drawTitle:withFrame:inView:.
2006-05-23	Corrected descriptions of bezel constant borders to specify points instead of pixels.

Index