

---

# NSActionCell Class Reference

[Cocoa](#) > [User Experience](#)



2007-02-28



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

---

## **NSActionCell Class Reference 5**

---

Overview	5
Tasks	5
Configuring an NSActionCell Object	5
Obtaining and Setting Cell Values	6
Managing the Cell's View	6
Assigning the Target and Action	6
Assigning a Tag	7
Instance Methods	7
action	7
controlView	7
doubleValue	8
floatValue	8
integerValue	8
intValue	9
setAction:	9
setAlignment:	9
setBezeled:	10
setBordered:	10
setControlView:	11
setEnabled:	11
setFloatingPointFormat:left:right:	11
setFont:	12
setImage:	13
setObjectValue:	13
setTag:	13
setTarget:	14
stringValue	14
tag	15
target	15

---

## **Document Revision History 17**

---

## **Index 19**

---



# NSActionCell Class Reference

---

<b>Inherits from</b>	NSCell : NSObject
<b>Conforms to</b>	NSCoding (NSCell) NSCopying (NSCell) 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>	Action Messages
<b>Declared in</b>	NSActionCell.h
<b>Related sample code</b>	Clock Control TrackBall

## Overview

An NSActionCell defines an active area inside a control (an instance of NSControl or one of its subclasses).

As an NSControl's active area, an NSActionCell does three things: it usually performs display of text or an icon; it provides the NSControl with a target and an action; and it handles mouse (cursor) tracking by properly highlighting its area and sending action messages to its target based on cursor movement.

## Tasks

### Configuring an NSActionCell Object

- [setAlignment:](#) (page 9)  
Sets the alignment of text in the receiver.
- [setBezeled:](#) (page 10)  
Sets whether the receiver draws itself with a bezeled border.
- [setBordered:](#) (page 10)  
Sets whether the receiver draws itself outlined with a plain border.
- [setEnabled:](#) (page 11)  
Sets whether the receiver is enabled or disabled.

- [setFloatingPointFormat:left:right:](#) (page 11)  
Sets the receiver's floating-point format.
- [setFont:](#) (page 12)  
Sets the font to be used when the receiver displays text.
- [setImage:](#) (page 13)  
Sets the image to be displayed in the receiver.

## Obtaining and Setting Cell Values

- [doubleValue](#) (page 8)  
Returns the receiver's value as a `double` after validating any editing of cell content.
- [floatValue](#) (page 8)  
Returns the receiver's value as a `float` after validating any editing of cell content.
- [intValue](#) (page 9)  
Returns the receiver's value as an `int` after validating any editing of cell content.
- [integerValue](#) (page 8)  
Returns the receiver's value as a 64-bit compatible integer after validating any editing of cell content.
- [stringValue](#) (page 14)  
Returns the receiver's value as a string object as converted by the cell's formatter, if one exists.
- [setObjectValue:](#) (page 13)  
Discards any editing of the receiver's text and sets its object value to *object*.

## Managing the Cell's View

- [controlView](#) (page 7)  
Returns the view in which the receiver was last drawn.
- [setControlView:](#) (page 11)  
Sets the receiver's control view, the view in which it is drawn.

## Assigning the Target and Action

- [setAction:](#) (page 9)  
Sets the selector used for action messages sent by the receiver's control.
- [action](#) (page 7)  
Returns the receiver's action-message selector.
- [setTarget:](#) (page 14)  
Sets the receiver's target object.
- [target](#) (page 15)  
Returns the receiver's target object.

## Assigning a Tag

- [setTag:](#) (page 13)  
Sets the receiver's tag.
- [tag](#) (page 15)  
Returns the receiver's tag.

## Instance Methods

### action

Returns the receiver's action-message selector.

- (SEL)action

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setAction:](#) (page 9)
- [setTarget:](#) (page 14)
- [target](#) (page 15)

#### Related Sample Code

Clock Control

#### Declared In

NSActionCell.h

### controlView

Returns the view in which the receiver was last drawn.

- (NSView \*)controlView

#### Return Value

The returned view is normally an `NSControl` object. The method returns `nil` if the receiver has no control view (usually because it hasn't yet been placed in the view hierarchy).

#### Availability

Available in Mac OS X v10.0 and later.

#### Related Sample Code

AnimatedSlider

#### Declared In

NSActionCell.h

## doubleValue

Returns the receiver's value as a `double` after validating any editing of cell content.

- (double)doubleValue

### Discussion

If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- `validateEditing` (NSControl)

### Declared In

NSActionCell.h

## floatValue

Returns the receiver's value as a `float` after validating any editing of cell content.

- (float)floatValue

### Discussion

If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- `validateEditing` (NSControl)

### Declared In

NSActionCell.h

## integerValue

Returns the receiver's value as a 64-bit compatible integer after validating any editing of cell content.

- (NSInteger)integerValue

### Return Value

A 64-bit compatible integer value, as defined by the `NSInteger` type.

### Discussion

If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

### Availability

Available in Mac OS X v10.5 and later.

### Declared In

NSActionCell.h



## intValue

Returns the receiver's value as an `int` after validating any editing of cell content.

- (int)intValue

### Discussion

If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [validateEditing \(NSControl\)](#)

### Declared In

NSActionCell.h

## setAction:

Sets the selector used for action messages sent by the receiver's control.

- (void)setAction:(SEL)aSelector

### Parameters

*aSelector*

The selector that identifies the action method to invoke.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [action](#) (page 7)
- [setTarget:](#) (page 14)
- [target](#) (page 15)

### Related Sample Code

Quartz Composer WWDC 2005 TextEdit  
TextEditPlus

### Declared In

NSActionCell.h

## setAlignment:

Sets the alignment of text in the receiver.

- (void)setAlignment:(NSTextAlignment)mode

**Parameters***mode*

One of five constants that specifies alignment within the cell: `NSLeftTextAlignment`, `NSRightTextAlignment`, `NSCenterTextAlignment`, `NSJustifiedTextAlignment`, and `NSNaturalTextAlignment` (the default alignment for the text).

**Discussion**

The method marks the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSActionCell.h`

**setBezeled:**

Sets whether the receiver draws itself with a bezeled border.

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

**Parameters***flag*

YES if the cell is to be drawn with a bezeled border, NO otherwise.

**Discussion**

After setting the attribute the method marks the receiver as needing redisplay. The `setBezeled:` and `setBordered:` (page 10) methods are mutually exclusive—that is, a border can be only plain or bezeled.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

`UIKitMovieShuffler`

**Declared In**

`NSActionCell.h`

**setBordered:**

Sets whether the receiver draws itself outlined with a plain border.

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

**Parameters***flag*

YES if the cell is to be drawn with a plain border, NO otherwise.

**Discussion**

After setting the attribute the method marks the receiver as needing redisplay. The `setBezeled:` (page 10) and `setBordered:` methods are mutually exclusive—that is, a border can be only plain or bezeled.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSActionCell.h

**setControlView:**

Sets the receiver's control view, the view in which it is drawn.

```
- (void)setControlView:(NSView *)view
```

**Parameters**

*view*

The view object, which is normally an NSControl *view*. Pass in *nil* if the receiver has no control view (usually because it hasn't yet been placed in the view hierarchy).

**Discussion**

The control view is typically set in the receiver's implementation of `drawWithFrame:inView:` (NSCell).

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

NSActionCell.h

**setEnabled:**

Sets whether the receiver is enabled or disabled.

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

**Parameters**

*flag*

YES if the cell is to be enabled, NO otherwise

**Discussion**

The text of disabled cells is changed to gray. If a cell is disabled, it cannot be highlighted, cannot be edited, and does not support mouse tracking (and thus cannot participate in target-action behavior). The method marks the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSActionCell.h

**setFloatingPointFormat:left:right:**

Sets the receiver's floating-point format.

```
- (void)setFloatingPointFormat:(BOOL)autoRange left:(NSUInteger)leftDigits
right:(NSUInteger)rightDigits
```

**Parameters***autoRange*

NO if you want the receiver to place digits to the right and left of the decimal point as specified (in *leftDigits* and *rightDigits*); YES if you want it to place the digits flexibly.

*leftDigits*

The maximum number of digits to the left of the decimal point. The receiver might interpret this value flexibly if *autoRange* is YES.

*rightDigits*

The maximum number of digits to the right of the decimal point. The receiver might interpret this value flexibly if *autoRange* is YES.

**Discussion**

The implementation of this method is based on the `NSCell` method

`setFloatingPointFormat:left:right:`. See the description of that method for details.

The `NSActionCell` implementation of the method supplements the `NSCell` implementation by marking the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

**Note:** This method is being deprecated in favor of `NSFormatter` objects. For more information, see `NSFormatter`. This documentation is provided only for developers who need to modify older applications.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSActionCell.h`

**setFont:**

Sets the font to be used when the receiver displays text.

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

**Parameters***fontObj*

The font object encapsulating information about the new font. If *fontObj* is `nil` and the receiver is a text-type cell, the font object currently held by the receiver is autoreleased.

**Discussion**

If the receiver is not a text-type cell, the method converts it to that type. `NSActionCell` supplements the `NSCell` implementation of this method by marking the updated cell as needing redisplay. If the receiver was converted to a text-type cell and is selected, it also updates the field editor with *fontObj*.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSActionCell.h`

**setImage:**

Sets the image to be displayed in the receiver.

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

**Parameters**

*image*

The image for the receiver to display. If *image* is *nil*, the image currently displayed by the receiver is removed.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

FunkyOverlayWindow

**Declared In**

NSActionCell.h

**setObjectValue:**

Discards any editing of the receiver's text and sets its object value to *object*.

```
- (void)setObjectValue:(id < NSCopying >)object
```

**Parameters**

*object*

The object value to assign to the receiver.

**Discussion**

If the object value is afterward different from what it was before the method was invoked, the method marks the receiver as needing redisplay.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSActionCell.h

**setTag:**

Sets the receiver's tag.

```
- (void)setTag:(NSInteger)anInt
```

**Parameters**

*anInt*

An integer tag to be associated with the receiver.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [tag](#) (page 15)

**Declared In**

NSActionCell.h

**setTarget:**

Sets the receiver's target object.

- (void)setTarget:(id)*anObject***Parameters***anObject*

The object that is the target of action messages sent by the receiver's control.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [action](#) (page 7)
- [setAction:](#) (page 9)
- [target](#) (page 15)

**Related Sample Code**

Quartz Composer WWDC 2005 TextEdit  
 TextEditPlus

**Declared In**

NSActionCell.h

**stringValue**

Returns the receiver's value as a string object as converted by the cell's formatter, if one exists.

- (NSString \*)stringValue

**Discussion**

If no formatter exists and the value is an `NSString`, returns the value as a plain, attributed, or localized formatted string. If the value is not an `NSString` or cannot be converted to one, returns an empty string. The method supplements the `NSCell` implementation by validating and retaining any editing changes being made to cell text.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [validateEditing](#) (NSControl)

**Declared In**

NSActionCell.h

## tag

Returns the receiver's tag.

- (NSInteger)tag

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setTag:](#) (page 13)

### Declared In

NSActionCell.h

## target

Returns the receiver's target object.

- (id)target

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [action](#) (page 7)  
- [setAction:](#) (page 9)  
- [setTarget:](#) (page 14)

### Related Sample Code

Clock Control

### Declared In

NSActionCell.h





# Document Revision History

---

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

Date	Notes
2007-02-28	Updated for Mac OS X version 10.5.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

action [instance method 7](#)

## C

---

convertView [instance method 7](#)

## D

---

doubleValue [instance method 8](#)

## F

---

floatValue [instance method 8](#)

## I

---

integerValue [instance method 8](#)

intValue [instance method 9](#)

## S

---

setAction: [instance method 9](#)

setAlignment: [instance method 9](#)

setBezeled: [instance method 10](#)

setBordered: [instance method 10](#)

setconvertView: [instance method 11](#)

setEnabled: [instance method 11](#)

setFloatingPointFormat:left:right: [instance method 11](#)

setFont: [instance method 12](#)

setImage: [instance method 13](#)

setObjectValue: [instance method 13](#)

setTag: [instance method 13](#)

setTarget: [instance method 14](#)

stringValue [instance method 14](#)

## T

---

tag [instance method 15](#)

target [instance method 15](#)