

---

# NSMenuItem Protocol Reference

**(Not Recommended)**

[Cocoa > User Experience](#)



2007-02-08



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

## **NSMenuItem Protocol Reference (Not Recommended) 5**

---

Overview	5
Tasks	5
Creating a Menu Item	5
Enabling a Menu Item	6
Setting the Target and Action	6
Setting the Title	6
Setting the Tag	6
Setting the State	6
Setting the Image	7
Managing Submenus	7
Getting a Separator Item	7
Setting the Owning Menu	7
Managing Key Equivalents	8
Managing Mnemonics	8
Managing User Key Equivalents	8
Managing Alternates	8
Managing Indentation Levels	9
Managing Tool Tips	9
Representing an Object	9
Class Methods	9
separatorItem	9
setUsesUserKeyEquivalents:	9
usesUserKeyEquivalents	10
Instance Methods	10
action	10
attributedTitle	10
hasSubmenu	11
image	11
indentationLevel	11
initWithTitle:action:keyEquivalent:	11
isAlternate	12
isEnabled	12
isSeparatorItem	13
keyEquivalent	13
keyEquivalentModifierMask	13
menu	13
mixedStateImage	14
mnemonic	14
mnemonicLocation	14
offStateImage	15

- onStateImage 15
- representedObject 16
- setAction: 16
- setAlternate: 16
- setAttributedTitle: 17
- setEnabled: 18
- setImage: 18
- setIndentationLevel: 18
- setKeyEquivalent: 19
- setKeyEquivalentModifierMask: 19
- setMenu: 20
- setMixedStateImage: 21
- setMnemonicLocation: 21
- setOffStateImage: 21
- setOnStateImage: 22
- setRepresentedObject: 22
- setState: 23
- setSubmenu: 23
- setTag: 24
- setTarget: 24
- setTitle: 25
- setTitleWithMnemonic: 25
- setToolTip: 25
- state 26
- submenu 26
- tag 27
- target 27
- title 27
- toolTip 27
- userKeyEquivalent 28
- userKeyEquivalentModifierMask 28

---

**Document Revision History 29**

---

**Index 31**

---

# NSMenuItem Protocol Reference (Not Recommended)


---

<b>Adopted by</b>	NSMenuItem
<b>Conforms to</b>	NSObject NSCopying NSCoding NSValidatedUserInterfaceItem
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.0 through Mac OS X v10.4.
<b>Companion guide</b>	Application Menu and Pop-up List Programming Topics for Cocoa
<b>Declared in</b>	NSMenuItem.h

## Overview

**Important:** The information in this document is obsolete and should not be used for new development.

Refer to the `NSMenuItem` class description, which replaces this protocol.

 **Warning:** The `NSMenuItem` protocol is being removed from the Application Kit; you must use the `NSMenuItem` class instead. This change does not affect binary compatibility between different versions of projects, but might cause failures in project builds. To adapt your projects to this change, alter all references to the protocol (for example, `id <NSMenuItem>`) to references to the class (`NSMenuItem *`).

## Tasks

### Creating a Menu Item

- `initWithTitle:action:keyEquivalent:` (page 11)  
Returns an initialized instance of an `NSMenuItem`.

## Enabling a Menu Item

- [setEnabled:](#) (page 18)  
Sets whether the receiver is enabled.
- [isEnabled](#) (page 12)  
Returns YES if the receiver is enabled, NO if not.

## Setting the Target and Action

- [setTarget:](#) (page 24)  
Sets the receiver's target.
- [target](#) (page 27)  
Returns the receiver's target.
- [setAction:](#) (page 16)  
Sets the receiver's action method.
- [action](#) (page 10)  
Returns the receiver's action method.

## Setting the Title

- [setTitle:](#) (page 25)  
Sets the receiver's title.
- [title](#) (page 27)  
Returns the receiver's title.
- [setAttributedTitle:](#) (page 17)  
Specifies a custom string for a menu item.
- [attributedTitle](#) (page 10)  
Returns the custom title string for a menu item.

## Setting the Tag

- [setTag:](#) (page 24)  
Sets the receiver's tag.
- [tag](#) (page 27)  
Returns the receiver's tag.

## Setting the State

- [setState:](#) (page 23)  
Sets the state of the receiver.
- [state](#) (page 26)  
Returns the state of the receiver.

## Setting the Image

- [setImage:](#) (page 18)  
Sets the receiver's image.
- [image](#) (page 11)  
Returns the image displayed by the receiver, or `nil` if it displays no image.
- [setImage:](#) (page 22)  
Sets the image of the receiver that indicates an "on" state.
- [onStateImage](#) (page 15)  
Returns the image used to depict the receiver's "on" state, or `nil` if the image has not been set.
- [setImage:](#) (page 21)  
Sets the image of the receiver that indicates an "off" state.
- [offStateImage](#) (page 15)  
Returns the image used to depict the receiver's "off" state, or `nil` if the image has not been set.
- [setImage:](#) (page 21)  
Sets the image of the receiver that indicates a "mixed" state, that is, a state neither "on" nor "off."
- [mixedStateImage](#) (page 14)  
Returns the image used to depict a "mixed state."

## Managing Submenus

- [setSubmenu:](#) (page 23)  
Sets the submenu of the receiver.
- [submenu](#) (page 26)  
Returns the submenu associated with the receiving menu item, or `nil` if no submenu is associated with it.
- [hasSubmenu](#) (page 11)  
Returns YES if the receiver has a submenu, NO if it doesn't.

## Getting a Separator Item

- + [separatorItem](#) (page 9)  
Returns a menu item that is used to separate logical groups of menu commands.
- [isSeparatorItem](#) (page 13)  
Returns whether the receiver is a separator item (that is, a menu item used to visually segregate related menu items).

## Setting the Owning Menu

- [setMenu:](#) (page 20)  
Sets the receiver's menu.
- [menu](#) (page 13)  
Returns the menu to which the receiver belongs, or `nil` if no menu has been set.

## Managing Key Equivalents

- + [setUsesUserKeyEquivalents:](#) (page 9)  
Sets whether menu items conform to user preferences for key equivalents.
- + [usesUserKeyEquivalents](#) (page 10)  
Returns YES if menu items conform to user preferences for key equivalents; otherwise, returns NO.
- [setKeyEquivalent:](#) (page 19)  
Sets the receiver's unmodified key equivalent.
- [keyEquivalent](#) (page 13)  
Returns the receiver's unmodified keyboard equivalent, or the empty string if one hasn't been defined.
- [setKeyEquivalentModifierMask:](#) (page 19)  
Sets the receiver's keyboard equivalent modifiers (indicating modifiers such as the Shift or Option key) to those in *mask*.
- [keyEquivalentModifierMask](#) (page 13)  
Returns the receiver's keyboard equivalent modifier mask.

## Managing Mnemonics

- [setMnemonicLocation:](#) (page 21)  
Sets the character of the specified menu item that is to be underlined.
- [mnemonicLocation](#) (page 14)  
Returns the position of the underlined character in the menu item title used as a mnemonic.
- [setTitleWithMnemonic:](#) (page 25)  
Sets the title of a menu item with a character underlined to denote an access key.
- [mnemonic](#) (page 14)  
Returns the character in the menu item title that appears underlined for use as a mnemonic.

## Managing User Key Equivalents

- [userKeyEquivalent](#) (page 28)  
Returns the user-assigned key equivalent for the receiver.
- [userKeyEquivalentModifierMask](#) (page 28)  
Returns the modifier mask for the receiver's user-assigned key equivalent.

## Managing Alternates

- [setAlternate:](#) (page 16)  
Marks the receiver as an alternate to the previous menu item.
- [isAlternate](#) (page 12)  
Returns whether the receiver is an alternate to the previous menu item.



## Managing Indentation Levels

- [setIndentationLevel:](#) (page 18)  
Sets the menu item indentation level for the receiver.
- [indentationLevel](#) (page 11)  
Returns the menu item indentation level for the receiver.

## Managing Tool Tips

- [setToolTip:](#) (page 25)  
Sets a help tag for a menu item.
- [toolTip](#) (page 27)  
Returns the help tag for a menu item.

## Representing an Object

- [setRepresentedObject:](#) (page 22)  
Sets the object represented by the receiver.
- [representedObject](#) (page 16)  
Returns the object that the receiving menu item represents.

## Class Methods

### separatorItem

Returns a menu item that is used to separate logical groups of menu commands.

```
+ (id <NSMenuItem>)separatorItem
```

#### Discussion

This menu item is disabled. The default separator item is blank space.

### setUsesUserKeyEquivalents:

Sets whether menu items conform to user preferences for key equivalents.

```
+ (void)setUsesUserKeyEquivalents:(BOOL)flag
```

#### Parameters

*flag*

If YES, menu items conform to user preferences for key equivalents; if NO, the key equivalents originally assigned to the menu items are used.

#### See Also

+ [usesUserKeyEquivalents](#) (page 10)

- [userKeyEquivalent](#) (page 28)

## usesUserKeyEquivalents

Returns YES if menu items conform to user preferences for key equivalents; otherwise, returns NO.

+ (BOOL)usesUserKeyEquivalents

### See Also

- + [setUsesUserKeyEquivalents:](#) (page 9)
- [userKeyEquivalent](#) (page 28)

## Instance Methods

### action

Returns the receiver's action method.

- (SEL)action

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [target](#) (page 27)
- [setAction:](#) (page 16)

### Declared In

NSMenuItem.h

### attributedString

Returns the custom title string for a menu item.

- (NSAttributedString \*)attributedString

### Availability

Available in Mac OS X v10.3 and later.

### See Also

- [setAttributedString:](#) (page 17)
- [title](#) (page 27)

### Declared In

NSMenuItem.h

## hasSubmenu

Returns YES if the receiver has a submenu, NO if it doesn't.

- (BOOL)hasSubmenu

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [setSubmenu:forItem:](#) (NSMenu)

### Declared In

NSMenuItem.h

## image

Returns the image displayed by the receiver, or nil if it displays no image.

- (NSImage \*)image

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

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

### Declared In

NSMenuItem.h

## indentationLevel

Returns the menu item indentation level for the receiver.

- (NSInteger)indentationLevel

### Discussion

The return value will be from 0 to 15. The default indentation level is 0.

### Availability

Available in Mac OS X v10.3 and later.

### See Also

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

### Declared In

NSMenuItem.h

## initWithTitle:action:keyEquivalent:

Returns an initialized instance of an NSMenuItem.

```
- (id)initWithTitle:(NSString *)itemName action:(SEL)anAction keyEquivalent:(NSString *)charCode
```

### Parameters

*itemName*

The title of the menu item. It must not be `nil` (if there is no title, specify an empty `NSString`).

*anAction*

The action selector to be associated with the menu item. It must be a valid selector or `NULL`.

*charCode*

A string representing a keyboard key to be used as the key equivalent. It must not be `nil` (if there is no key equivalent, specify an empty `NSString`).

### Return Value

Returns an instance of `NSMenuItem` or `nil` if the object couldn't be created.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### Declared In

`NSMenuItem.h`

## isAlternate

Returns whether the receiver is an alternate to the previous menu item.

```
- (BOOL)isAlternate
```

### Availability

Available in Mac OS X v10.3 and later.

### See Also

- [setAlternate:](#) (page 16)

### Declared In

`NSMenuItem.h`

## isEnabled

Returns YES if the receiver is enabled, NO if not.

```
- (BOOL)isEnabled
```

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

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

### Declared In

`NSMenuItem.h`

## isSeparatorItem

Returns whether the receiver is a separator item (that is, a menu item used to visually segregate related menu items).

- (BOOL)isSeparatorItem

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### Declared In

NSMenuItem.h

## keyEquivalent

Returns the receiver's unmodified keyboard equivalent, or the empty string if one hasn't been defined.

- (NSString \*)keyEquivalent

### Discussion

Use [keyEquivalentModifierMask](#) (page 13) to determine the modifier mask for the key equivalent.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [userKeyEquivalent](#) (page 28)

- [mnemonic](#) (page 14)

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

### Declared In

NSMenuItem.h

## keyEquivalentModifierMask

Returns the receiver's keyboard equivalent modifier mask.

- (NSUInteger)keyEquivalentModifierMask

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [setKeyEquivalentModifierMask:](#) (page 19)

### Declared In

NSMenuItem.h

## menu

Returns the menu to which the receiver belongs, or `nil` if no menu has been set.

- (NSMenu \*)menu

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

#### See Also

- [setMenu:](#) (page 20)

#### Declared In

NSMenuItem.h

## mixedStateImage

Returns the image used to depict a “mixed state.”

- (NSImage \*)mixedStateImage

#### Discussion

A mixed state is useful for indicating “off” and “on” attribute values in a group of selected objects, such as a selection of text containing bold and plain (nonbolded) words.

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

#### See Also

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

#### Declared In

NSMenuItem.h

## mnemonic

Returns the character in the menu item title that appears underlined for use as a mnemonic.

- (NSString \*)mnemonic

#### Discussion

If there is no mnemonic character, returns an empty string. Mnemonics are not supported in Mac OS X.

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

#### See Also

- [setTitleWithMnemonic:](#) (page 25)

#### Declared In

NSMenuItem.h

## mnemonicLocation

Returns the position of the underlined character in the menu item title used as a mnemonic.

- (NSUInteger)mnemonicLocation

#### Discussion

The position is the zero-based index of that character in the title string. If the receiver has no mnemonic character, returns `NSNotFound`. Mnemonics are not supported in Mac OS X.

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

#### See Also

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

#### Declared In

NSMenuItem.h

## offStateImage

Returns the image used to depict the receiver's "off" state, or `nil` if the image has not been set.

- (NSImage \*)offStateImage

#### Discussion

By default, there is no off state image.

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

#### See Also

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

#### Declared In

NSMenuItem.h

## onStateImage

Returns the image used to depict the receiver's "on" state, or `nil` if the image has not been set.

- (NSImage \*)onStateImage

#### Discussion

By default, the on state image is a checkmark.

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

#### See Also

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

#### Declared In

NSMenuItem.h

## representedObject

Returns the object that the receiving menu item represents.

- (id)representedObject

### Discussion

For example, you might have a menu list the names of views that are swapped into the same panel. The represented objects would be the appropriate `NSView` objects. The user would then be able to switch back and forth between the different views that are displayed by selecting the various menu items.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [tag](#) (page 27)
- [setRepresentedObject:](#) (page 22)

### Declared In

NSMenuItem.h

## setAction:

Sets the receiver's action method.

- (void)setAction:(SEL)aSelector

### Parameters

*aSelector*

A selector identifying the action method.

### Discussion

See *Action Messages* for additional information on action messages.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [setTarget:](#) (page 24)
- [action](#) (page 10)

### Declared In

NSMenuItem.h

## setAlternate:

Marks the receiver as an alternate to the previous menu item.

- (void)setAlternate:(BOOL)isAlternate

### Parameters

*isAlternate*

YES if the receiver is an alternate to the previous menu item, NO otherwise.



**Discussion**

If the receiver has the same key equivalent as the previous item, but has different key equivalent modifiers, the items are folded into a single visible item and the appropriate item shows while tracking the menu. The menu items may also have no key equivalent as long as the key equivalent modifiers are different.

If there are two or more items with no key equivalent but different modifiers, then the only way to get access to the alternate items is with the mouse. If you mark items as alternates but their key equivalents don't match, they might be displayed as separate items. Marking the first item as an alternate has no effect.

The *isAlternate* value is archived.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [isAlternate](#) (page 12)

**Declared In**

NSMenuItem.h

**setAttributedTitle:**

Specifies a custom string for a menu item.

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

**Parameters**

*string*

An attributed string to use as the receiver's title.

**Discussion**

You can use this method to add styled text and embedded images to menu item strings. If you do not set a text color for the attributed string, it is black when not selected, white when selected, and gray when disabled. Colored text remains unchanged when selected.

When you call this method to set the menu title to an attributed string, the [setTitle:](#) (page 25) method is also called to set the menu title with a plain string. If you clear the attributed title, the plain title remains unchanged.

The attributed string is not archived in the old nib format.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [attributedString](#) (page 10)

- [setTitle:](#) (page 25)

**Declared In**

NSMenuItem.h

**setEnabled:**

Sets whether the receiver is enabled.

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

**Parameters**

*flag*

YES if the receiver is to be enabled, NO otherwise.

**Discussion**

If a menu item is disabled, its keyboard equivalent is also disabled. See the NSMenuValidation informal protocol specification for cautions regarding this method.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [isEnabled](#) (page 12)

**Declared In**

NSMenuItem.h

**setImage:**

Sets the receiver's image.

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

**Parameters**

*menuItemImage*

An NSImage object representing an image to be displayed in the menu item. If *menuItemImage* is nil, the current image (if any) is removed.

**Discussion**

The menu item's image is not affected by changes in its state.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [image](#) (page 11)

**Declared In**

NSMenuItem.h

**setIndentationLevel:**

Sets the menu item indentation level for the receiver.

```
- (void)setIndentationLevel:(NSInteger)indentationLevel
```

**Parameters***indentationLevel*

The value for *indentationLevel* may be from 0 to 15. If *indentationLevel* is greater than 15, the value is pinned to the maximum. If *indentationLevel* is less than 0, an exception is raised. The default indentation level is 0.

**Discussion**

The *indentationLevel* value is archived.

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [indentationLevel](#) (page 11)

**Declared In**

NSMenuItem.h

**setKeyEquivalent:**

Sets the receiver's unmodified key equivalent.

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

**Parameters***aKeyEquivalent*

A string containing a character code representing a keyboard key. If you want to remove the key equivalent from a menu item, pass an empty string (@" ") for *aString* (never pass nil).

**Discussion**

Use [setKeyEquivalentModifierMask:](#) (page 19) to set the appropriate mask for the modifier keys for the key equivalent.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

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

- [keyEquivalent](#) (page 13)

**Declared In**

NSMenuItem.h

**setKeyEquivalentModifierMask:**

Sets the receiver's keyboard equivalent modifiers (indicating modifiers such as the Shift or Option key) to those in *mask*.

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

**Parameters***mask*

The key masks indicate modifiers such as the Shift or Option keys. *mask* is an integer bit field containing any of these modifier key masks, combined using the C bitwise OR operator:

NSShiftKeyMask

NSAlternateKeyMask

NSCommandKeyMask

NSControlKeyMask

You should always set `NSCommandKeyMask` in *mask*.

`NSShiftKeyMask` is relevant only for function keys—that is, for key events whose modifier flags include `NSFunctionKeyMask`. For all other key events `NSShiftKeyMask` is ignored, and characters typed while the Shift key is pressed are interpreted as the shifted versions of those characters; for example, Command-Shift-c is interpreted as Command-C.

**Discussion**

See the `NSEvent` class specification for more information about modifier mask values.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [keyEquivalentModifierMask](#) (page 13)

**Declared In**

`NSMenuItem.h`

**setMenu:**

Sets the receiver's menu.

```
- (void)setMenu:(NSMenu *)aMenu
```

**Parameters***aMenu*

The menu object that "owns" the receiver.

**Discussion**

This method is invoked by the owning `NSMenu` object when the receiver is added or removed. You shouldn't have to invoke this method in your own code, although it can be overridden to provide specialized behavior.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [menu](#) (page 13)

**Declared In**

`NSMenuItem.h`

## setMixedStateImage:

Sets the image of the receiver that indicates a “mixed” state, that is, a state neither “on” nor “off.”

```
- (void)setMixedStateImage:(NSImage *)itemImage
```

### Parameters

*itemImage*

The `NSImage` object to use for the “mixed” state of the menu item. If *itemImage* is `nil`, any current mixed-state image is removed.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [mixedStateImage](#) (page 14)
- [setOffStateImage:](#) (page 21)
- [setOnStateImage:](#) (page 22)
- [setState:](#) (page 23)

### Declared In

`NSMenuItem.h`

## setMnemonicLocation:

Sets the character of the specified menu item that is to be underlined.

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

### Parameters

*location*

An integer index into the character array of the title.

### Discussion

This character identifies the access key by which users can access the menu item. Mnemonics are not supported in Mac OS X.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [mnemonicLocation](#) (page 14)

### Declared In

`NSMenuItem.h`

## setOffStateImage:

Sets the image of the receiver that indicates an “off” state.

```
- (void)setOffStateImage:(NSImage *)itemImage
```

**Parameters***itemImage*

The `NSImage` object to use for the "off" state of the menu item. If *itemImage* is `nil`, any current off-state image is removed.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [offStateImage](#) (page 15)
- [setMixedStateImage:](#) (page 21)
- [setOffStateImage:](#) (page 21)
- [setState:](#) (page 23)

**Declared In**

`NSMenuItem.h`

**setOnStateImage:**

Sets the image of the receiver that indicates an "on" state.

```
- (void)setOnStateImage:(NSImage *)itemImage
```

**Parameters***itemImage*

The `NSImage` object to use for the "on" state of the menu item. If *itemImage* is `nil`, any current on-state image is removed.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [onStateImage](#) (page 15)
- [setMixedStateImage:](#) (page 21)
- [setOffStateImage:](#) (page 21)
- [setState:](#) (page 23)

**Declared In**

`NSMenuItem.h`

**setRepresentedObject:**

Sets the object represented by the receiver.

```
- (void)setRepresentedObject:(id)anObject
```

**Parameters***anObject*

The object to be represented by the receiver.

**Discussion**

By setting a represented object for a menu item, you make an association between the menu item and that object. The represented object functions as a more specific form of tag that allows you to associate any object, not just an arbitrary integer, with the items in a menu.

For example, an `NSView` object might be associated with a menu item—when the user chooses the menu item, the represented object is fetched and displayed in a panel. Several menu items might control the display of multiple views in the same panel.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [setTag:](#) (page 24)
- [representedObject](#) (page 16)

**Declared In**

`NSMenuItem.h`

**setState:**

Sets the state of the receiver.

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

**Parameters**

*itemState*

An integer constant representing a state; it should be one of `NSOffState`, `NSOnState`, or `NSMixedState`.

**Discussion**

The image associated with the new state is displayed to the left of the menu item.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [state](#) (page 26)
- [setMixedStateImage:](#) (page 21)
- [setOffStateImage:](#) (page 21)
- [setOnStateImage:](#) (page 22)

**Declared In**

`NSMenuItem.h`

**setSubmenu:**

Sets the submenu of the receiver.

```
- (void)setSubmenu:(NSMenu *) aSubmenu
```

### Parameters

*aSubmenu*

The menu object to set as submenu.

### Discussion

The default implementation raises an exception if *aSubmenu* already has a supermenu.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [submenu](#) (page 26)
- [hasSubmenu](#) (page 11)

### Declared In

NSMenuItem.h

## setTag:

Sets the receiver's tag.

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

### Parameters

*anInt*

An integer tag to associate with the receiver.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [setRepresentedObject:](#) (page 22)
- [tag](#) (page 27)

### Declared In

NSMenuItem.h

## setTarget:

Sets the receiver's target.

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

### Parameters

*anObject*

An object to be the target of action messages sent by the receiver.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [setAction:](#) (page 16)
- [target](#) (page 27)



**Declared In**

NSMenuItem.h

**setTitle:**

Sets the receiver's title.

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

**Parameters**

*aString*

The new title of the menu item. If you do not want a title, use an empty string, not nil.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [title](#) (page 27)

**Declared In**

NSMenuItem.h

**setTitleWithMnemonic:**

Sets the title of a menu item with a character underlined to denote an access key.

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

**Discussion**

Mnemonics are not supported in Mac OS X.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**See Also**

- [mnemonic](#) (page 14)

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

**Declared In**

NSMenuItem.h

**setToolTip:**

Sets a help tag for a menu item.

- (void)setToolTip:(NSString \*)*toolTip*

**Parameters**

*toolTip*

A short string that describes the menu item.

### Discussion

You can invoke this method for any menu item, including items in the main menu bar. This string is not archived in the old nib format.

### Availability

Available in Mac OS X v10.3 and later.

### See Also

- [toolTip](#) (page 27)

### Declared In

NSMenuItem.h

## state

Returns the state of the receiver.

- (NSInteger)state

### Discussion

A menu-item state can be one of `NSOffState` (the default), `NSOnState`, or `NSMixedState`.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

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

### Declared In

NSMenuItem.h

## submenu

Returns the submenu associated with the receiving menu item, or `nil` if no submenu is associated with it.

- (NSMenu \*)submenu

### Discussion

If the receiver responds YES to [hasSubmenu](#) (page 11), the submenu is returned.

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [hasSubmenu](#) (page 11)

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

### Declared In

NSMenuItem.h

## tag

Returns the receiver's tag.

- (NSInteger)tag

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [representedObject](#) (page 16)
- [setTag:](#) (page 24)

### Declared In

NSMenuItem.h

## target

Returns the receiver's target.

- (id)target

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [action](#) (page 10)
- [setTarget:](#) (page 24)

### Declared In

NSMenuItem.h

## title

Returns the receiver's title.

- (NSString \*)title

### Availability

Available in Mac OS X v10.0 through Mac OS X v10.4.

### See Also

- [setTitle:](#) (page 25)

### Declared In

NSMenuItem.h

## toolTip

Returns the help tag for a menu item.

- (NSString \*)toolTip

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [setToolTip:](#) (page 25)

**Declared In**

NSMenuItem.h

## **userKeyEquivalent**

Returns the user-assigned key equivalent for the receiver.

- (NSString \*)userKeyEquivalent

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**Declared In**

NSMenuItem.h

## **userKeyEquivalentModifierMask**

Returns the modifier mask for the receiver's user-assigned key equivalent.

- (NSUInteger)userKeyEquivalentModifierMask

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.4.

**Declared In**

NSMenuItem.h

# Document Revision History

---

This table describes the changes to *NSMenuItem Protocol Reference*.

Date	Notes
2007-02-08	Moving to Legacy because all content is deprecated.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

`action` protocol instance method [10](#)  
`attributedString` protocol instance method [10](#)

## H

---

`hasSubMenu` protocol instance method [11](#)

## I

---

`image` protocol instance method [11](#)  
`indentationLevel` protocol instance method [11](#)  
`initWithTitle:action:keyEquivalent:` protocol instance method [11](#)  
`isAlternate` protocol instance method [12](#)  
`isEnabled` protocol instance method [12](#)  
`isSeparatorItem` protocol instance method [13](#)

## K

---

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

## M

---

`menu` protocol instance method [13](#)  
`mixedStateImage` protocol instance method [14](#)  
`mnemonic` protocol instance method [14](#)  
`mnemonicLocation` protocol instance method [14](#)

## O

---

`offStateImage` protocol instance method [15](#)  
`onStateImage` protocol instance method [15](#)

## R

---

`representedObject` protocol instance method [16](#)

## S

---

`separatorItem` protocol class method [9](#)  
`setAction:` protocol instance method [16](#)  
`setAlternate:` protocol instance method [16](#)  
`setAttributedString:` protocol instance method [17](#)  
`setEnabled:` protocol instance method [18](#)  
`setImage:` protocol instance method [18](#)  
`setIndentationLevel:` protocol instance method [18](#)  
`setKeyEquivalent:` protocol instance method [19](#)  
`setKeyEquivalentModifierMask:` protocol instance method [19](#)  
`setMenu:` protocol instance method [20](#)  
`setMixedStateImage:` protocol instance method [21](#)  
`setMnemonicLocation:` protocol instance method [21](#)  
`setOffStateImage:` protocol instance method [21](#)  
`setOnStateImage:` protocol instance method [22](#)  
`setRepresentedObject:` protocol instance method [22](#)  
`setState:` protocol instance method [23](#)  
`setSubMenu:` protocol instance method [23](#)  
`setTag:` protocol instance method [24](#)  
`setTarget:` protocol instance method [24](#)  
`setTitle:` protocol instance method [25](#)  
`setTitleWithMnemonic:` protocol instance method [25](#)  
`setToolTip:` protocol instance method [25](#)  
`setUsesUserKeyEquivalents:` protocol class method [9](#)  
`state` protocol instance method [26](#)  
`submenu` protocol instance method [26](#)

## T

---

tag [protocol instance method 27](#)  
target [protocol instance method 27](#)  
title [protocol instance method 27](#)  
toolTip [protocol instance method 27](#)

## U

---

userKeyEquivalent [protocol instance method 28](#)  
userKeyEquivalentModifierMask [protocol instance method 28](#)  
usesUserKeyEquivalents [protocol class method 10](#)