

---

# UIMenuController Class Reference

Data Management: Event Handling



2010-02-25



Apple Inc.  
© 2010 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, iPhone, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

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

## **UIMenuController Class Reference 5**

---

Overview	5
Tasks	6
Getting the Menu Controller Instance	6
Showing and Hiding the Menu	6
Positioning the Menu	6
Updating the Menu	6
Customizing Menu Items	6
Properties	6
arrowDirection	6
menuFrame	7
menuItems	7
menuVisible	8
Class Methods	8
sharedMenuController	8
Instance Methods	8
setMenuVisible:animated:	8
setTargetRect:inView:	9
update	9
Constants	10
UIMenuControllerArrowDirection	10
Notifications	11
UIMenuControllerWillShowMenuNotification	11
UIMenuControllerDidShowMenuNotification	11
UIMenuControllerWillHideMenuNotification	11
UIMenuControllerDidHideMenuNotification	11
UIMenuControllerMenuFrameDidChangeNotification	12

## **Document Revision History 13**

---



# UIMenuController Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/UIKit.framework
<b>Availability</b>	Available in iOS 3.0 and later.
<b>Declared in</b>	UIMenuController.h

## Overview

The singleton `UIMenuController` instance presents the menu interface for the Cut, Copy, Paste, Select, Select All, and Delete commands.

This menu is referred to as the editing menu. When you make this menu visible, `UIMenuController` positions it relative to a target rectangle on the screen; this rectangle usually defines a selection. The menu appears above the target rectangle or, if there is not enough space for it, below it. The menu's pointer is placed at the center of the top or bottom of the target rectangle, as appropriate. Be sure to set the tracking rectangle before you make the menu visible. You are also responsible for detecting, tracking, and displaying selections.

The `UIResponderStandardEditActions` informal protocol declares methods that are invoked when the user taps a menu command. The `canPerformAction:withSender:` method of `UIResponder` is also related to the editing menu. A responder implements this method to enable and disable commands of the editing menu just before the menu is displayed. You can force this updating of menu commands' enabled state by calling the [update](#) (page 9) method.

**Note:** iOS 3.2 introduced three changes to this class:

- You can add your own menu items to the editing menu via the [menuItems](#) (page 7) property.
- You can control the direction the arrow of the editing menu points through the [arrowDirection](#) (page 6) property.
- The Delete menu item was added to the set of system menu items. Tapping it invokes the `UIResponderStandardEditActions` action method `delete:`, also added in iOS 3.2.

## Tasks

### Getting the Menu Controller Instance

- + [sharedMenuController](#) (page 8)  
Returns the menu controller.

### Showing and Hiding the Menu

- [menuVisible](#) (page 8) *property*  
The visibility of the editing menu.
- [setMenuVisible:animated:](#) (page 8)  
Shows or hides the editing menu, optionally animating the action.

### Positioning the Menu

- [setTargetRect:inView:](#) (page 9)  
Sets the area in a view above or below which the editing menu is positioned.
- [menuFrame](#) (page 7) *property*  
Returns the frame of the editing menu. (read-only)
- [arrowDirection](#) (page 6) *property*  
The direction the arrow of the editing menu is pointing.

### Updating the Menu

- [update](#) (page 9)  
Updates the enabled state of menu commands

### Customizing Menu Items

- [menuItems](#) (page 7) *property*  
The custom menu items for the editing menu.

## Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

### **arrowDirection**

The direction the arrow of the editing menu is pointing.

```
@property UIMenuControllerArrowDirection arrowDirection
```

**Discussion**

You can set the direction editing-menu arrow points by assigning a [UIMenuControllerArrowDirection](#) (page 10) enum constant to this property. The default behavior ([UIMenuControllerArrowDefault](#) (page 10)) is to point up or down at the object of focus based on its location on the screen.

**Availability**

Available in iOS 3.2 and later.

**Declared In**

UIMenuController.h

## menuFrame

Returns the frame of the editing menu. (read-only)

```
@property(nonatomic, readonly) CGRect menuFrame
```

**Discussion**

The property value is the bounding rectangle of the menu in screen coordinates. The property holds a value even if the menu is not visible. You can use this property to adjust any user-interface objects away from the menu.

**Availability**

Available in iOS 3.0 and later.

**See Also**

- [setTargetRect:inView:](#) (page 9)

**Declared In**

UIMenuController.h

## menuItems

The custom menu items for the editing menu.

```
@property(copy) NSArray *menuItems
```

**Discussion**

The default value is `nil` (no custom menu items). Each menu item is an instance of the `UIMenuItem` class. You may create your own menu items, each with its own title and action selector, and add them to the editing menu through this property. Custom items appear in the menu after any system menu items.

**Availability**

Available in iOS 3.2 and later.

**Declared In**

UIMenuController.h

## menuVisible

The visibility of the editing menu.

```
@property(n nonatomic, getter=isMenuVisible) BOOL menuVisible
```

### Discussion

Setting this property displays or hides the menu immediately, without animation. For animating the showing or hiding of the menu, use the `setMenuVisible:animated:` (page 8) method. Before showing the menu, be sure to position it relative to the selection.

### Availability

Available in iOS 3.0 and later.

### Declared In

UIMenuController.h

## Class Methods

### sharedMenuController

Returns the menu controller.

```
+ (UIMenuController *)sharedMenuController
```

### Return Value

The shared `NSMenuController` instance.

### Availability

Available in iOS 3.0 and later.

### Declared In

UIMenuController.h

## Instance Methods

### setMenuVisible:animated:

Shows or hides the editing menu, optionally animating the action.

```
- (void)setMenuVisible:(BOOL)menuVisibleanimated:(BOOL)animated
```

### Parameters

*menuVisible*

YES if the menu should be shown, NO if it should be hidden.

*animated*

YES if the showing or hiding of the menu should be animated, otherwise NO.



**Discussion**

Before showing the menu, be sure to position it relative to the selection. See [setTargetRect:inView:](#) (page 9) for details. If you do not set the target rect before displaying the menu, it appears at screen coordinates (0.0, 0.0).

**Availability**

Available in iOS 3.0 and later.

**See Also**

[@property menuVisible](#) (page 8)

**Declared In**

UIMenuController.h

**setTargetRect:inView:**

Sets the area in a view above or below which the editing menu is positioned.

```
- (void)setTargetRect:(CGRect)targetRect inView:(UIView *)targetView
```

**Parameters**

*targetRect*

A rectangle that defines the area that is to be the target of the menu commands.

*targetView*

The view in which *targetRect* appears.

**Discussion**

This target rectangle (*targetRect*) is usually the bounding rectangle of a selection. `UIMenuController` positions the editing menu above this rectangle; if there is not enough space for the menu there, it positions it below the rectangle. The menu's pointer is placed at the center of the top or bottom of the target rectangle as appropriate. Note that if you make the width or height of the target rectangle zero, `UIMenuController` treats the target area as a line or point for positioning (for example, an insertion caret or a single point).

Once it is set, the target rectangle does not track the view; if the view moves (such as would happen in a scroll view), you must update the target rectangle accordingly.

**Availability**

Available in iOS 3.0 and later.

**See Also**

[@property menuFrame](#) (page 7)

**Declared In**

UIMenuController.h

**update**

Updates the enabled state of menu commands

```
- (void)update
```

**Discussion**

By default, `UIMenuController` calls this method just before the editing menu is made visible and when touches occur in the menu. As a result, a responder object in the application enables or disables menu commands depending on the context; for example, if the pasteboard holds no data of a compatible type, the Paste command would be disabled. You can call this method to force an update of the editing menu. You may also override this method to add any custom behavior.

**Availability**

Available in iOS 3.0 and later.

**Declared In**

`UIMenuController.h`

## Constants

**UIMenuControllerArrowDirection**

The direction the arrow of the editing menu is pointing.

```
typedef enum {
    UIMenuControllerArrowDefault,
    UIMenuControllerArrowUp,
    UIMenuControllerArrowDown,
    UIMenuControllerArrowLeft,
    UIMenuControllerArrowRight,
} UIMenuControllerArrowDirection;
```

**Constants**

`UIMenuControllerArrowDefault`

The arrow is pointing up or down at the object of focus based on its location in the screen.

Available in iOS 3.2 and later.

Declared in `UIMenuController.h`.

`UIMenuControllerArrowUp`

The arrow is pointing up at the object of focus.

Available in iOS 3.2 and later.

Declared in `UIMenuController.h`.

`UIMenuControllerArrowDown`

The arrow is pointing down at the object of focus.

Available in iOS 3.2 and later.

Declared in `UIMenuController.h`.

`UIMenuControllerArrowLeft`

The arrow is pointing left at the object of focus.

Available in iOS 3.2 and later.

Declared in `UIMenuController.h`.

`UIMenuControllerArrowRight`

The arrow is pointing right at the object of focus.

Available in iOS 3.2 and later.

Declared in `UIMenuController.h`.

**Availability**

Available in iOS 3.2 and later.

**Declared In**

`UIMenuController.h`

## Notifications

### **UIMenuControllerWillShowMenuNotification**

Posted by the menu controller just before it shows the menu.

There is no `userInfo` dictionary.

**Availability**

Available in iOS 3.0 and later.

**Declared In**

`UIMenuController.h`

### **UIMenuControllerDidShowMenuNotification**

Posted by the menu controller just after it shows the menu.

There is no `userInfo` dictionary.

**Availability**

Available in iOS 3.0 and later.

**Declared In**

`UIMenuController.h`

### **UIMenuControllerWillHideMenuNotification**

Posted by the menu controller just before it hides the menu.

There is no `userInfo` dictionary.

**Availability**

Available in iOS 3.0 and later.

**Declared In**

`UIMenuController.h`

### **UIMenuControllerDidHideMenuNotification**

Posted by the menu controller just after it hides the menu.

There is no `userInfo` dictionary.

**Availability**

Available in iOS 3.0 and later.

**Declared In**

`UIMenuController.h`

**UIMenuControllerMenuFrameDidChangeNotification**

Posted when the frame of a visible menu changes.

There is no `userInfo` dictionary.

**Availability**

Available in iOS 3.0 and later.

**Declared In**

`UIMenuController.h`

# Document Revision History

---

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

Date	Notes
2010-02-25	Describes the <code>menuItems</code> property and <code>UIMenuControllerArrowDirection</code> constants added in iOS 3.2.
2009-05-19	Made minor corrections.
2009-03-08	First version of the document that describes the class used to present the editing menu interface.

**REVISION HISTORY**

Document Revision History