

---

# NSController Class Reference

[Cocoa > Data Management](#)



2007-04-02



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

---

## **NSController Class Reference 5**

---

Overview	5
Adopted Protocols	5
Tasks	5
Managing Editing	5
Instance Methods	6
commitEditing	6
commitEditingWithDelegate:didCommitSelector:contextInfo:	6
discardEditing	7
isEditing	8
objectDidBeginEditing:	8
objectDidEndEditing:	8

---

## **Document Revision History 9**

---

## **Index 11**

---



# NSController Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/AppKit.framework
<b>Availability</b>	Available in Mac OS X v10.3 and later.
<b>Companion guide</b>	Cocoa Bindings Programming Topics
<b>Declared in</b>	NSController.h

## Overview

NSController is an abstract class that implements the NSEditor and NSEditorRegistration informal protocols required for controller classes.

## Adopted Protocols

- NSCoding
- `encodeWithCoder:`
  - `initWithCoder:`

## Tasks

### Managing Editing

- [objectDidBeginEditing:](#) (page 8)  
Invoked to inform the receiver that *editor* has uncommitted changes that can affect the receiver.
- [objectDidEndEditing:](#) (page 8)  
Invoked to inform the receiver that *editor* has committed or discarded its changes.
- [commitEditing:](#) (page 6)  
Causes the receiver to attempt to commit any pending edits, returning YES if successful or no edits were pending.

- [commitEditingWithDelegate:didCommitSelector:contextInfo:](#) (page 6)  
Attempts to commit any pending changes in known editors of the receiver.
- [discardEditing](#) (page 7)  
Discards any pending changes by registered editors.
- [isEditing](#) (page 8)  
Returns YES if there are any editors currently registered with the receiver, NO otherwise.

## Instance Methods

### commitEditing

Causes the receiver to attempt to commit any pending edits, returning YES if successful or no edits were pending.

- (BOOL)commitEditing

#### Discussion

The receiver invokes `commitEditing` on any current editors, returning their response. A commit is denied if the receiver fails to apply the changes to the model object, perhaps due to a validation error.

#### Availability

Available in Mac OS X v10.3 and later.

#### See Also

- [discardEditing](#) (page 7)

#### Declared In

NSController.h

### commitEditingWithDelegate:didCommitSelector:contextInfo:

Attempts to commit any pending changes in known editors of the receiver.

```
-(void)commitEditingWithDelegate:(id)delegate
    didCommitSelector:(SEL)didCommitSelector contextInfo:(void *)contextInfo
```

#### Parameters

*delegate*

An object that can serve as the receiver's delegate. It should implement the method specified by *didCommitSelector*.

*didCommitSelector*

A selector that is invoked on delegate. The method specified by the selector must have the same signature as the following method:

```
-(void)editor:(id)editor didCommit:(BOOL)didCommit contextInfo:(void *)contextInfo
```

*contextInfo*

Contextual information that is sent as the `contextInfo` argument to delegate when `didCommitSelector` is invoked.

#### Discussion

Provides support for the `NSEditor` informal protocol. This method attempts to commit pending changes in known editors. Known editors are either instances of a subclass of `NSController` or (more rarely) user interface controls that may contain pending edits—such as text fields—that registered with the context using `objectDidBeginEditing:` and have not yet unregistered using a subsequent invocation of `objectDidEndEditing:`.

The receiver iterates through the array of its known editors and invokes `commitEditing` on each. The receiver then sends the message specified by the `didCommitSelector` selector to the specified delegate.

The `didCommit` argument is the value returned by the editor specified by `editor` from the `commitEditing` message. The `contextInfo` argument is the same value specified as the `contextInfo` parameter—you may use this value however you wish.

If an error occurs while attempting to commit, for example if key-value coding validation fails, your implementation of this method should typically send the view in which editing is being performed a `presentError:modalForWindow:delegate:didRecoverSelector:contextInfo: message`, specifying the view's containing window.

You may find this method useful in some situations (typically if you are using Cocoa Bindings) when you want to ensure that pending changes are applied before a change in user interface state. For example, you may need to ensure that changes pending in a text field are applied before a window is closed. See also [commitEditing](#) (page 6) which performs a similar function but which allows you to handle any errors directly, although it provides no information beyond simple success/failure.

#### Availability

Available in Mac OS X v10.4 and later.

#### See Also

- [commitEditing](#) (page 6)
- [discardEditing](#) (page 7)
- [objectDidBeginEditing:](#) (page 8)
- [objectDidEndEditing:](#) (page 8)

#### Declared In

`NSController.h`

## discardEditing

Discards any pending changes by registered editors.

- (void)discardEditing

#### Discussion

The receiver invokes `discardEditing` on any current editors.

#### Availability

Available in Mac OS X v10.3 and later.

**See Also**

- [commitEditing](#) (page 6)

**Declared In**

NSController.h

## isEditing

Returns YES if there are any editors currently registered with the receiver, NO otherwise.

- (BOOL)isEditing

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

NSController.h

## objectDidBeginEditing:

Invoked to inform the receiver that *editor* has uncommitted changes that can affect the receiver.

- (void)objectDidBeginEditing:(id)editor

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [objectDidEndEditing:](#) (page 8)

**Declared In**

NSController.h

## objectDidEndEditing:

Invoked to inform the receiver that *editor* has committed or discarded its changes.

- (void)objectDidEndEditing:(id)editor

**Availability**

Available in Mac OS X v10.3 and later.

**See Also**

- [objectDidBeginEditing:](#) (page 8)

**Declared In**

NSController.h



# Document Revision History

---

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

Date	Notes
2007-04-02	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

**REVISION HISTORY**

Document Revision History

# Index

---

## C

---

`commitEditing` **instance method** [6](#)  
`commitEditingWithDelegate:didCommitSelector:  
contextInfo:` **instance method** [6](#)

## D

---

`discardEditing` **instance method** [7](#)

## I

---

`isEditing` **instance method** [8](#)

## O

---

`objectDidBeginEditing:` **instance method** [8](#)  
`objectDidEndEditing:` **instance method** [8](#)