

---

# NSComboBoxCell Class Reference

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



2006-05-23



Apple Inc.  
© 2006 Apple Computer, 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

## **NSComboBoxCell Class Reference 5**

---

Overview	5
Tasks	5
Initializing a Combo Box Cell	5
Setting Display Attributes	5
Setting a Data Source	6
Working with an Internal List	6
Manipulating the Displayed List	7
Manipulating the Selection	7
Completing the Text Field	7
Instance Methods	8
addItemWithObjectValues:	8
addItemWithObjectValue:	8
completedString:	8
completes	9
dataSource	9
deselectItemAtIndex:	10
encodeWithCoder:	10
hasVerticalScroller	11
indexOfItemWithObjectValue:	11
indexofSelectedItem	11
initWithCoder:	12
insertItemWithObjectValue:atIndex:	12
intercellSpacing	13
isButtonBordered	13
itemHeight	14
itemObjectValueAtIndex:	14
noteNumberOfItemsChanged	14
numberOfItems	15
numberOfVisibleItems	15
objectValueOfSelectedItem	16
objectValues	16
reloadData	16
removeAllItems	17
removeItemAtIndex:	17
removeItemWithObjectValue:	17
scrollItemAtIndexToTop:	18
scrollItemAtIndexToVisible:	18
selectItemAtIndex:	19
selectItemWithObjectValue:	19
setButtonBordered:	20

setCompletes: 20  
setDataSource: 20  
setHasVerticalScroller: 21  
setIntercellSpacing: 21  
setItemHeight: 22  
setNumberOfVisibleItems: 22  
setUsesDataSource: 23  
usesDataSource 23

---

**Document Revision History 25**

---

**Index 27**

---

# NSComboBoxCell Class Reference

---

<b>Inherits from</b>	NSTextFieldCell : NSActionCell : 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>	Combo Box Programming Topics
<b>Declared in</b>	NSComboBoxCell.h

## Overview

`NSComboBoxCell` is a subclass of `NSTextFieldCell` used to implement the user interface of “combo boxes” (see `NSComboBox` for information on how combo boxes look and work). The `NSComboBox` subclass of `NSTextField` uses a single `NSComboBoxCell`, and essentially all of the `NSComboBox` class’ methods simply invoke the corresponding `NSComboBoxCell` method.

Also see the `NSComboBoxCellDataSource` informal protocol, which declares the methods that an `NSComboBoxCell` object uses to access the contents of its data source object.

## Tasks

### Initializing a Combo Box Cell

- [initWithCoder:](#) (page 12)  
Initializes a newly allocated instance from data in the specified coder.
- [encodeWithCoder:](#) (page 10)  
Encodes the receiver using the specified coder.

### Setting Display Attributes

- [hasVerticalScroller](#) (page 11)  
Returns a Boolean value indicating whether the receiver will display a vertical scroller.

- [isButtonBordered](#) (page 13)  
Returns a Boolean value indicating whether the combo box button is set to display a border.
- [intercellSpacing](#) (page 13)  
Returns the spacing between cells in the receiver's pop-up list.
- [itemHeight](#) (page 14)  
Returns the height of each item in the receiver's pop-up list.
- [numberOfVisibleItems](#) (page 15)  
Returns the maximum number of items visible in the pop-up list.
- [setButtonBordered:](#) (page 20)  
Determines whether the button in the combo box is displayed with a border.
- [setHasVerticalScroller:](#) (page 21)  
Determines whether the receiver displays a vertical scroller.
- [setIntercellSpacing:](#) (page 21)  
Sets the spacing between pop-up list items.
- [setItemHeight:](#) (page 22)  
Sets the height for items.
- [setNumberOfVisibleItems:](#) (page 22)  
Sets the maximum number of items that are visible in the pop-up list.

## Setting a Data Source

- [dataSource](#) (page 9)  
Returns the object that provides the data displayed in the receiver's pop-up list.
- [setDataSource:](#) (page 20)  
Sets the receiver's data source.
- [setUsesDataSource:](#) (page 23)  
Sets whether the receiver uses an external data source to populate its pop-up list.
- [usesDataSource](#) (page 23)  
Returns a Boolean value indicating whether the receiver uses an external data source.

## Working with an Internal List

- [addItemWithObjectValues:](#) (page 8)  
Adds multiple objects to the internal item list.
- [addItemWithObjectValue:](#) (page 8)  
Adds the specified object to the internal item list.
- [insertItemWithObjectValue:atIndex:](#) (page 12)  
Inserts an object at the specified location in the internal item list.
- [objectValues](#) (page 16)  
Returns the receiver's internal item list.
- [removeAllItems](#) (page 17)  
Removes all items from the receiver's internal item list.

- [removeItemAtIndex:](#) (page 17)  
Removes the object at the specified location from the receiver's internal item list.
- [removeItemWithObjectValue:](#) (page 17)  
Removes all occurrences of the specified object from the receiver's internal item list.
- [numberOfItems](#) (page 15)  
Returns the total number of items in the pop-up list.

## Manipulating the Displayed List

- [indexOfItemWithObjectValue:](#) (page 11)  
Searches the receiver's internal item list for the given object and returns the matching index number.
- [itemObjectValueAtIndex:](#) (page 14)  
Returns the object located at the specified location in the internal item list.
- [noteNumberOfItemsChanged](#) (page 14)  
Informs the receiver that the number of items in its data source has changed.
- [reloadData](#) (page 16)  
Marks the receiver as needing redisplay, so that it will reload the data for visible pop-up items and draw the new values.
- [scrollItemAtIndexToTop:](#) (page 18)  
Scrolls the receiver's pop-up list vertically so that the item at the given index is as close to the top as possible.
- [scrollItemAtIndexToVisible:](#) (page 18)  
Scrolls the receiver's pop-up list vertically so that the item at the given index is visible.

## Manipulating the Selection

- [deselectItemAtIndex:](#) (page 10)  
Deselects the pop-up list item at the given index if it's selected.
- [indexOfSelectedItem](#) (page 11)  
Returns the index of the last item selected from the pop-up list.
- [objectValueOfSelectedItem](#) (page 16)  
Returns the object corresponding to the last item selected from the pop-up list.
- [selectItemAtIndex:](#) (page 19)  
Selects the pop-up list row at the given index.
- [selectItemWithObjectValue:](#) (page 19)  
Selects the first pop-up list item that corresponds to the specified object.

## Completing the Text Field

- [completedString:](#) (page 8)  
Returns a string from the receiver's pop-up list that starts with the given substring.
- [completes](#) (page 9)  
Returns a Boolean value indicating whether the receiver tries to complete text entered by the user.

- [setCompletes:](#) (page 20)  
Sets whether the receiver tries to complete what the user types in the text field.

## Instance Methods

### **addItemWithObjectValues:**

Adds multiple objects to the internal item list.

```
- (void)addItemWithObjectValues:(NSArray *)objects
```

#### **Parameters**

*objects*

The object to add to the end of the receiver's internal item list.

#### **Discussion**

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

#### **Availability**

Available in Mac OS X v10.0 and later.

#### **Declared In**

NSComboBoxCell.h

### **addItemWithObjectValue:**

Adds the specified object to the internal item list.

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

#### **Parameters**

*anObject*

The object to add to the end of the receiver's internal item list.

#### **Discussion**

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

#### **Availability**

Available in Mac OS X v10.0 and later.

#### **Declared In**

NSComboBoxCell.h

### **completedString:**

Returns a string from the receiver's pop-up list that starts with the given substring.

```
- (NSString *)completedString:(NSString *)substring
```



**Parameters***substring*

The substring to search for. This is what the user entered in the combo box's text field.

**Return Value**

The string from the receiver's pop-up list that starts with the specified substring or `nil` if there is no such string.

**Discussion**

The default implementation of this method first checks whether the combo box uses a data source and whether the data source responds to `comboBox:completedString:` or `comboBoxCell:completedString:`. If so, the combo box cell returns that method's return value. Otherwise, this method goes through the combo box's items one by one and returns an item that starts with *substring*.

Override this method only if your subclass completes strings differently. The overriding method does not need to call the superclass's method. Generally, you do not need to call this method directly.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSComboBoxCell.h`

**completes**

Returns a Boolean value indicating whether the receiver tries to complete text entered by the user.

- (BOOL)completes

**Return Value**

YES if the receiver tries to complete what the user types in the text field; NO otherwise.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

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

**Declared In**

`NSComboBoxCell.h`

**dataSource**

Returns the object that provides the data displayed in the receiver's pop-up list.

- (id)dataSource

**Return Value**

The data source for the receiver's pop-up list.

**Discussion**

This method logs a warning if [usesDataSource](#) (page 23) returns NO. See the class description and the `NSComboBoxCellDataSource` informal protocol specification for more information on combo box cell data source objects.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSComboBoxCell.h

**deselectItemAtIndex:**

Deselects the pop-up list item at the given index if it's selected.

```
- (void)deselectItemAtIndex:(NSInteger) index
```

**Parameters**

*index*

The index of the item to deselect.

**Discussion**

If the selection does in fact change, this method posts an `NSComboBoxSelectionDidChangeNotification` to the default notification center.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [indexOfSelectedItem](#) (page 11)
- [numberOfItems](#) (page 15)
- [selectItemAtIndex:](#) (page 19)

**Declared In**

NSComboBoxCell.h

**encodeWithCoder:**

Encodes the receiver using the specified coder.

```
- (void)encodeWithCoder:(NSCoder *) encoder
```

**Parameters**

*encoder*

The coder to use when encoding the receiver. If the receiver uses a data source, the data source is conditionally encoded as well.

**Availability**

Available in Mac OS X v10.0 through Mac OS X v10.2.

**See Also**

- [initWithCoder:](#) (page 12)

**Declared In**

NSComboBoxCell.h

## hasVerticalScroller

Returns a Boolean value indicating whether the receiver will display a vertical scroller.

- (BOOL)hasVerticalScroller

### Return Value

YES if the receiver displays a vertical scroller; otherwise NO.

### Discussion

Note that the scroller will be displayed even if the pop-up list contains fewer items than will fit in the area specified for display.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [numberOfItems](#) (page 15)
- [numberOfVisibleItems](#) (page 15)

### Declared In

NSComboBoxCell.h

## indexOfItemWithObjectValue:

Searches the receiver's internal item list for the given object and returns the matching index number.

- (NSInteger)indexOfItemWithObjectValue:(id)anObject

### Parameters

*anObject*

The object for which to return the index.

### Return Value

The lowest index whose corresponding value is equal to *anObject*. Objects are considered equal if they have the same `id` or if `isEqual:` returns YES. If none of the objects in the receiver's internal item list is equal to *anObject*, `indexOfItemWithObjectValue:` returns `NSNotFound`.

### Discussion

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

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

### Declared In

NSComboBoxCell.h

## indexOfSelectedItem

Returns the index of the last item selected from the pop-up list.

- (NSInteger)indexOfSelectedItem

#### Return Value

The index of the last item selected from the receiver's pop-up list or -1 if no item is selected.

#### Discussion

Note that nothing is initially selected in a newly initialized combo box cell.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [objectValueOfSelectedItem](#) (page 16)

#### Declared In

NSComboBoxCell.h

### initWithCoder:

Initializes a newly allocated instance from data in the specified coder.

- (id)initWithCoder:(NSCoder \*)*decoder*

#### Parameters

*decoder*

The coder from which to draw the data to initialize the combo box cell.

#### Return Value

The initialized NSComboBoxCell object.

#### Discussion

If the decoded instance uses a data source, `initWithCoder:` decodes the data source as well.

#### Availability

Available in Mac OS X v10.0 through Mac OS X v10.2.

#### See Also

- [encodeWithCoder:](#) (page 10)

#### Declared In

NSComboBoxCell.h

### insertItemWithObjectValue:atIndex:

Inserts an object at the specified location in the internal item list.

- (void)insertItemWithObjectValue:(id)*anObject* atIndex:(NSInteger)*index*

#### Parameters

*anObject*

The object to add to the receiver's internal item list.

*index*

The index at which to add the specified object. The previous item at *index*—along with all following items—is shifted down one slot to make room.

#### Discussion

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [addItemWithObjectValue:](#) (page 8)
- [numberOfItems](#) (page 15)

#### Declared In

NSComboBoxCell.h

## intercellSpacing

Returns the spacing between cells in the receiver's pop-up list.

- (NSSize)intercellSpacing

#### Return Value

The horizontal and vertical spacing between cells in the receiver's pop-up list. The default spacing is (3.0, 2.0).

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [itemHeight](#) (page 14)
- [numberOfVisibleItems](#) (page 15)

#### Declared In

NSComboBoxCell.h

## isButtonBordered

Returns a Boolean value indicating whether the combo box button is set to display a border.

- (BOOL)isButtonBordered

#### Return Value

YES if the button has a border; otherwise NO.

#### Availability

Available in Mac OS X v10.3 and later.

#### See Also

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

**Declared In**

NSComboBoxCell.h

**itemHeight**

Returns the height of each item in the receiver's pop-up list.

- (CGFloat)itemHeight

**Return Value**

The height of each item in the pop-up list. The default item height is 16.0.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [intercellSpacing](#) (page 13)
- [numberOfVisibleItems](#) (page 15)

**Declared In**

NSComboBoxCell.h

**itemObjectValueAtIndex:**

Returns the object located at the specified location in the internal item list.

- (id)itemObjectValueAtIndex:(NSInteger) *index***Parameters***index*The index of the object to return. If *index* is beyond the end of the list, an `NSRangeException` is raised.**Return Value**

The object at the given location in the receiver's internal item list.

**Discussion**This method logs a warning if [usesDataSource](#) (page 23) returns YES.**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [objectValueOfSelectedItem](#) (page 16)

**Declared In**

NSComboBoxCell.h

**noteNumberOfItemsChanged**

Informs the receiver that the number of items in its data source has changed.

- (void)noteNumberOfItemsChanged

#### Discussion

This method allows the receiver to update the scrollers in its displayed pop-up list without actually reloading data into the receiver. It is particularly useful for a data source that continually receives data in the background over a period of time, in which case the `NSComboBoxCell` can remain responsive to the user while the data is received.

See the `NSComboBoxCellDataSource` informal protocol specification for information on the messages an `NSComboBoxCell` sends to its data source.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [reloadData](#) (page 16)

#### Declared In

`NSComboBoxCell.h`

## numberOfItems

Returns the total number of items in the pop-up list.

- (NSInteger)numberOfItems

#### Return Value

The number of items in the pop-up list.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [numberOfVisibleItems](#) (page 15)

- `numberOfItemsInComboBoxCell:` (`NSComboBoxCellDataSource` protocol)

#### Declared In

`NSComboBoxCell.h`

## numberOfVisibleItems

Returns the maximum number of items visible in the pop-up list.

- (NSInteger)numberOfVisibleItems

#### Return Value

The maximum number of items that are visible in the receiver's pop-up list at any one time.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [numberOfItems](#) (page 15)

**Declared In**

NSComboBoxCell.h

**objectValueOfSelectedItem**

Returns the object corresponding to the last item selected from the pop-up list.

```
- (id)objectValueOfSelectedItem
```

**Return Value**

The object from the receiver's internal item list corresponding to the last item selected from the pop-up list, or `nil` if no item is selected.

**Discussion**

Note that nothing is initially selected in a newly initialized combo box cell. This method logs a warning if [usesDataSource](#) (page 23) returns YES.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [indexOfSelectedItem](#) (page 11)
- `comboBoxCell:objectValueForItemAtIndex:` (NSComboBoxCellDataSource protocol)

**Declared In**

NSComboBoxCell.h

**objectValues**

Returns the receiver's internal item list.

```
- (NSArray *)objectValues
```

**Return Value**

An array containing the objects in the receiver's internal item list.

**Discussion**

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSComboBoxCell.h

**reloadData**

Marks the receiver as needing redisplay, so that it will reload the data for visible pop-up items and draw the new values.

```
- (void)reloadData
```



**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [noteNumberOfItemsChanged](#) (page 14)

**Declared In**

NSComboBoxCell.h

**removeAllItems**

Removes all items from the receiver's internal item list.

- (void)removeAllItems

**Discussion**

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [objectValues](#) (page 16)

**Declared In**

NSComboBoxCell.h

**removeItemAtIndex:**

Removes the object at the specified location from the receiver's internal item list.

- (void)removeItemAtIndex:(NSInteger)*index*

**Parameters**

*index*

The index of the object to remove from the receiver's internal item list. All items beyond *index* are moved up one slot to fill the gap.

**Discussion**

The removed object receives a `release` message. This method raises an `NSRangeException` if *index* is beyond the end of the list and logs a warning if [usesDataSource](#) (page 23) returns YES.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSComboBoxCell.h

**removeItemWithObjectValue:**

Removes all occurrences of the specified object from the receiver's internal item list.

- (void)removeItemWithObjectValue:(id)anObject

#### Parameters

*anObject*

The object to remove from the receiver's internal item list. Objects are considered equal if they have the same `id` or if `isEqual:` returns YES.

#### Discussion

This method logs a warning if [usesDataSource](#) (page 23) returns YES.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [indexOfItemWithObjectValue:](#) (page 11)

#### Declared In

NSComboBoxCell.h

## scrollItemAtIndexToTop:

Scrolls the receiver's pop-up list vertically so that the item at the given index is as close to the top as possible.

- (void)scrollItemAtIndexToTop:(NSInteger)index

#### Parameters

*index*

The index of the item to scroll to the top.

#### Discussion

The pop-up list need not be displayed at the time this method is invoked.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

NSComboBoxCell.h

## scrollItemAtIndexToVisible:

Scrolls the receiver's pop-up list vertically so that the item at the given index is visible.

- (void)scrollItemAtIndexToVisible:(NSInteger)index

#### Parameters

*index*

The index of the item to make visible.

#### Discussion

The pop-up list need not be displayed at the time this method is invoked.

#### Availability

Available in Mac OS X v10.0 and later.

**Declared In**

NSComboBoxCell.h

**selectItemAtIndex:**

Selects the pop-up list row at the given index.

```
- (void)selectItemAtIndex:(NSInteger) index
```

**Parameters***index*

The index of the row to select.

**Discussion**

Posts an `NSComboBoxSelectionDidChangeNotification` to the default notification center if the selection does in fact change. Note that this method does not alter the contents of the combo box cell's text field—see “Setting the Combo Box's Value” for more information.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `setObjectValue:` (NSControl)

**Declared In**

NSComboBoxCell.h

**selectItemWithObjectValue:**

Selects the first pop-up list item that corresponds to the specified object.

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

**Parameters***anObject*

The object for which to select the corresponding pop-up list item. Objects are considered equal if they have the same `id` or if `isEqual:` returns YES.

**Discussion**

This method logs a warning if `usesDataSource` (page 23) returns YES. Posts an `NSComboBoxSelectionDidChangeNotification` to the default notification center if the selection does in fact change. Note that this method doesn't alter the contents of the combo box cell's text field—see “Setting the Combo Box's Value” for more information.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `setObjectValue:` (NSControl)

**Declared In**

NSComboBoxCell.h

## setButtonBordered:

Determines whether the button in the combo box is displayed with a border.

- (void)setButtonBordered:(BOOL)*flag*

### Parameters

*flag*

YES to display a border. For example, it is often useful when using a combo box in an `NSTableView` to display the button without the border.

### Availability

Available in Mac OS X v10.3 and later.

### See Also

- [isButtonBordered](#) (page 13)

### Declared In

`NSComboBoxCell.h`

## setCompletes:

Sets whether the receiver tries to complete what the user types in the text field.

- (void)setCompletes:(BOOL)*completes*

### Parameters

*completes*

YES to indicate that the receiver should try to complete text typed by the user. If *completes* is YES, every time the user adds characters to the end of the text field, the combo box calls the `NSComboBoxCell` method [completedString:](#) (page 8).

### Discussion

If [completedString:](#) (page 8) returns a string that's longer than the existing string, the combo box replaces the existing string with the returned string and selects the additional characters. If the user is deleting characters or adds characters somewhere besides the end of the string, the combo box does not try to complete it.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [completes](#) (page 9)

### Declared In

`NSComboBoxCell.h`

## setDataSource:

Sets the receiver's data source.

- (void)setDataSource:(id)*aSource*

**Parameters***aSource*

The data source for the receiver. *aSource* should implement the appropriate methods of the `NSComboBoxCellDataSource` informal protocol.

This method logs a warning if *aSource* doesn't respond to either `numberOfItemsInComboBoxCell:` or `comboBoxCell:objectValueForItemAtIndex:`.

**Discussion**

This method doesn't automatically set `usesDataSource` (page 23) to `NO` and in fact logs a warning if `usesDataSource` returns `NO`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

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

**Declared In**

`NSComboBoxCell.h`

**setHasVerticalScroller:**

Determines whether the receiver displays a vertical scroller.

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

**Parameters***flag*

YES to have the receiver display a vertical scroller. By default, *flag* is YES.

**Discussion**

If *flag* is `NO` and the combo box cell has more list items (either in its internal item list or from its data source) than are allowed by `numberOfVisibleItems` (page 15), only a subset will be displayed. `NSComboBoxCell's scroll...` methods can be used to position this subset within the pop-up list.

Note that if *flag* is YES, a scroller will be displayed even if the combo box cell has fewer list items than are allowed by `numberOfVisibleItems`.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [numberOfItems](#) (page 15)
- [scrollItemAtIndexToTop:](#) (page 18)
- [scrollItemAtIndexToVisible:](#) (page 18)

**Declared In**

`NSComboBoxCell.h`

**setIntercellSpacing:**

Sets the spacing between pop-up list items.

- (void)setInterCellSpacing:(NSSize)*aSize*

#### Parameters

*aSize*

The width and height between pop-up list items. The default intercell spacing is (3.0, 2.0).

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setItemHeight:](#) (page 22)
- [setNumberOfVisibleItems:](#) (page 22)

#### Declared In

NSComboBoxCell.h

### setItemHeight:

Sets the height for items.

- (void)setItemHeight:(CGFloat)*itemHeight*

#### Parameters

*itemHeight*

The height of pop-up list items.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [setInterCellSpacing:](#) (page 21)
- [setNumberOfVisibleItems:](#) (page 22)

#### Declared In

NSComboBoxCell.h

### setNumberOfVisibleItems:

Sets the maximum number of items that are visible in the pop-up list.

- (void)setNumberOfVisibleItems:(NSInteger)*visibleItems*

#### Parameters

*visibleItems*

The maximum number of items that should be visible at one time in the receiver's pop-up list.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [numberOfItems](#) (page 15)
- [numberOfVisibleItems](#) (page 15)
- [setInterCellSpacing:](#) (page 21)

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

**Declared In**

NSComboBoxCell.h

**setUsesDataSource:**

Sets whether the receiver uses an external data source to populate its pop-up list.

- (void)setUsesDataSource:(BOOL)flag

**Parameters**

*flag*

YES to indicate that the receiver uses an external data source (specified by [setDataSource:](#) (page 20)) to populate the receiver's pop-up list.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSComboBoxCell.h

**usesDataSource**

Returns a Boolean value indicating whether the receiver uses an external data source.

- (BOOL)usesDataSource

**Return Value**

YES if the receiver uses an external data source to populate the receiver's pop-up list, NO if it uses an internal item list.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [dataSource](#) (page 9)

**Declared In**

NSComboBoxCell.h





# Document Revision History

---

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

Date	Notes
2006-05-23	First publication of this content as a separate document.

**REVISION HISTORY**

Document Revision History

# Index

---

## A

---

addItemWithObjectValues: [instance method 8](#)  
addItemWithObjectValue: [instance method 8](#)

## C

---

completedString: [instance method 8](#)  
completes [instance method 9](#)

## D

---

dataSource [instance method 9](#)  
deselectItemAtIndex: [instance method 10](#)

## E

---

encodeWithCoder: [instance method 10](#)

## H

---

hasVerticalScroller [instance method 11](#)

## I

---

indexOfItemWithObjectValue: [instance method 11](#)  
indexOfSelectedItem [instance method 11](#)  
initWithCoder: [instance method 12](#)  
insertItemWithObjectValue:atIndex: [instance method 12](#)  
intercellSpacing [instance method 13](#)  
isButtonBordered [instance method 13](#)  
itemHeight [instance method 14](#)

itemObjectValueAtIndex: [instance method 14](#)

## N

---

noteNumberOfItemsChanged [instance method 14](#)  
numberOfItems [instance method 15](#)  
numberOfVisibleItems [instance method 15](#)

## O

---

objectValueOfSelectedItem [instance method 16](#)  
objectValues [instance method 16](#)

## R

---

reloadData [instance method 16](#)  
removeAllItems [instance method 17](#)  
removeItemAtIndex: [instance method 17](#)  
removeItemWithObjectValue: [instance method 17](#)

## S

---

scrollItemAtIndexToTop: [instance method 18](#)  
scrollItemAtIndexToVisible: [instance method 18](#)  
selectItemAtIndex: [instance method 19](#)  
selectItemWithObjectValue: [instance method 19](#)  
setButtonBordered: [instance method 20](#)  
setCompletes: [instance method 20](#)  
setDataSource: [instance method 20](#)  
setHasVerticalScroller: [instance method 21](#)  
setIntercellSpacing: [instance method 21](#)  
setItemHeight: [instance method 22](#)  
setNumberOfVisibleItems: [instance method 22](#)  
setUsesDataSource: [instance method 23](#)

## U

---

usesDataSource instance method [23](#)