

---

# NSMutableSet Class Reference

[Cocoa > Data Management](#)



2006-09-19



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.

iPhone is a trademark of Apple Inc.

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

---

## **NSMutableSet Class Reference 5**

---

- Overview 5
- Tasks 6
  - Creating a Mutable Set 6
  - Adding and Removing Entries 6
  - Combining and Recombining Sets 6
- Class Methods 7
  - setWithCapacity: 7
- Instance Methods 7
  - addObject: 7
  - addObjectsFromArray: 8
  - filterUsingPredicate: 8
  - initWithCapacity: 9
  - intersectSet: 9
  - minusSet: 10
  - removeAllObjects 10
  - removeObject: 11
  - setSet: 11
  - unionSet: 11

---

## **Document Revision History 13**

---

## **Index 15**

---



# NSMutableSet Class Reference

---

<b>Inherits from</b>	NSSet : NSObject
<b>Conforms to</b>	NSCoding (NSSet) NSCopying (NSSet) NSMutableCopying (NSSet) NSFastEnumeration (NSSet) NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Collections Programming Topics for Cocoa
<b>Declared in</b>	NSPredicate.h NSSet.h
<b>Related sample code</b>	Core Data HTML Store CoreRecipes CustomAtomicStoreSubclass MyPhoto QTMetadataEditor

## Overview

The `NSMutableSet` class declares the programmatic interface to an object that manages a mutable set of objects. `NSMutableSet` provides support for the mathematical concept of a set. A set, both in its mathematical sense and in the `NSMutableSet` implementation, is an unordered collection of distinct elements.

The `NSCountedSet` class, which is a concrete subclass of `NSMutableSet`, supports mutable sets that can contain multiple instances of the same element. The `NSSet` class supports creating and managing immutable sets.

You add objects to an `NSMutableSet` object with `addObject:` (page 7), which adds a single object to the set; `addObjectsFromArray:` (page 8), which adds all objects from a specified array to the set; or `unionSet:` (page 11), which adds all the objects from another set. You remove objects from an `NSMutableSet` object using any of the methods `intersectSet:` (page 9), `minusSet:` (page 10), `removeAllObjects` (page 10), or `removeObject:` (page 11).

When an object is added to a set, it receives a `retain` message. When an object is removed from a mutable set, it receives a `release` message. If there are no further references to the object, this means that the object is deallocated. If your program keeps a reference to such an object, the reference will become invalid unless you send the object a `retain` message before it's removed from the array. For example, if `anObject` is not retained before it is removed from the set, the third statement below could result in a runtime error:

```
id anObject = [[aSet anyObject] retain];
[aSet removeObject:anObject];
[anObject someMessage];
```

## Tasks

### Creating a Mutable Set

- + [initWithCapacity:](#) (page 7)  
Creates and returns a mutable set with a given initial capacity.
- [initWithCapacity:](#) (page 9)  
Returns an initialized mutable set with a given initial capacity.

### Adding and Removing Entries

- [addObject:](#) (page 7)  
Adds a given object to the receiver, if it is not already a member.
- [filterUsingPredicate:](#) (page 8)  
Evaluates a given predicate against the receiver's content and removes from the receiver those objects for which the predicate returns false.
- [removeObject:](#) (page 11)  
Removes a given object from the receiver.
- [removeAllObjects](#) (page 10)  
Empties the receiver of all of its members.
- [addObjectsFromArray:](#) (page 8)  
Adds to the receiver each object contained in a given array that is not already a member.

### Combining and Recombining Sets

- [unionSet:](#) (page 11)  
Adds to the receiver each object contained in another given set that is not already a member.
- [minusSet:](#) (page 10)  
Removes from the receiver each object contained in another given set that is present in the receiver.
- [intersectSet:](#) (page 9)  
Removes from the receiver each object that isn't a member of another given set.
- [setSet:](#) (page 11)  
Empties the receiver, then adds to the receiver each object contained in another given set.

## Class Methods

### initWithCapacity:

Creates and returns a mutable set with a given initial capacity.

```
+ (id) initWithCapacity:(NSUInteger)numItems
```

#### Parameters

*numItems*

The initial capacity of the new set.

#### Return Value

A mutable set with initial capacity to hold *numItems* members.

#### Discussion

Mutable sets allocate additional memory as needed, so *numItems* simply establishes the object's initial capacity.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [initWithCapacity:](#) (page 9)

+ set (NSSet)

+ setWithObjects:count: (NSSet)

#### Declared In

NSSet.h

## Instance Methods

### addObject:

Adds a given object to the receiver, if it is not already a member.

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

#### Parameters

*anObject*

The object to add to the receiver.

#### Discussion

If *anObject* is already present in the set, this method has no effect on either the set or *anObject*.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

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

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

#### Related Sample Code

Core Data HTML Store  
 CoreRecipes  
 CustomAtomicStoreSubclass  
 QTMetadataEditor  
 Sketch-112

#### Declared In

NSSet.h

### addObjectsFromArray:

Adds to the receiver each object contained in a given array that is not already a member.

```
- (void)addObjectsFromArray:(NSArray *)anArray
```

#### Parameters

*anArray*

An array of objects to add to the receiver.

#### Discussion

If a given element of the array is already present in the set, this method has no effect on either the set or the array element.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [addObject:](#) (page 7)  
 - [unionSet:](#) (page 11)

#### Related Sample Code

Core Data HTML Store  
 CoreRecipes

#### Declared In

NSSet.h

### filterUsingPredicate:

Evaluates a given predicate against the receiver's content and removes from the receiver those objects for which the predicate returns false.

```
- (void)filterUsingPredicate:(NSPredicate *)predicate
```

#### Parameters

*predicate*

A predicate.



**Discussion**

The following example illustrates the use of this method.

```
NSMutableSet *mutableSet =
    [NSMutableSet setWithObjects:@"One", @"Two", @"Three", @"Four", nil];
NSPredicate *predicate =
    [NSPredicate predicateWithFormat:@"SELF beginswith 'T'"];
[mutableSet filterUsingPredicate:predicate];
// mutableSet contains (Two, Three)
```

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

NSPredicate.h

**initWithCapacity:**

Returns an initialized mutable set with a given initial capacity.

```
- (id)initWithCapacity:(NSUInteger)numItems
```

**Parameters**

*numItems*

The initial capacity of the set.

**Return Value**

An initialized mutable set with initial capacity to hold *numItems* members. The returned object might be different than the original receiver.

**Discussion**

Mutable sets allocate additional memory as needed, so *numItems* simply establishes the object's initial capacity.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [initWithCapacity:](#) (page 7)

**Declared In**

NSSet.h

**intersectSet:**

Removes from the receiver each object that isn't a member of another given set.

```
- (void)intersectSet:(NSSet *)otherSet
```

**Parameters**

*otherSet*

The set with which to perform the intersection.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [removeObject:](#) (page 11)
- [removeAllObjects](#) (page 10)
- [minusSet:](#) (page 10)

**Declared In**

NSSet.h

**minusSet:**

Removes from the receiver each object contained in another given set that is present in the receiver.

```
- (void)minusSet:(NSSet *)otherSet
```

**Parameters**

*otherSet*

The set of objects to remove from the receiver.

**Discussion**

If any member of *otherSet* isn't present in the receiving set, this method has no effect on either the receiver or the *otherSet* member.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [removeObject:](#) (page 11)
- [removeAllObjects](#) (page 10)
- [intersectSet:](#) (page 9)

**Related Sample Code**

CoreRecipes

MyPhoto

**Declared In**

NSSet.h

**removeAllObjects**

Empties the receiver of all of its members.

```
- (void)removeAllObjects
```

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [removeObject:](#) (page 11)
- [minusSet:](#) (page 10)

- [intersectSet:](#) (page 9)

#### Related Sample Code

CoreRecipes

#### Declared In

NSSet.h

### removeObject:

Removes a given object from the receiver.

- (void)removeObject:(id)anObject

#### Parameters

*anObject*

The object to remove from the receiver.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [removeAllObjects](#) (page 10)

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

- [intersectSet:](#) (page 9)

#### Related Sample Code

CoreRecipes

#### Declared In

NSSet.h

### setSet:

Empties the receiver, then adds to the receiver each object contained in another given set.

- (void)setSet:(NSSet \*)otherSet

#### Parameters

*otherSet*

The set whose members replace the receiver's content.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

NSSet.h

### unionSet:

Adds to the receiver each object contained in another given set that is not already a member.

- (void)unionSet:(NSSet \*)*otherSet*

**Parameters**

*otherSet*

The set of objects to add to the receiver.

**Discussion**

If any member of *otherSet* is already present in the receiver, this method has no effect on either the receiver or the *otherSet* member.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [addObject:](#) (page 7)
- [addObjectsFromArray:](#) (page 8)

**Declared In**

NSSet.h

# Document Revision History

---

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

Date	Notes
2006-09-19	Included API introduced in Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

addObject: [instance method 7](#)  
addObjectsFromArray: [instance method 8](#)

## F

---

filterUsingPredicate: [instance method 8](#)

## I

---

initWithCapacity: [instance method 9](#)  
intersectSet: [instance method 9](#)

## M

---

minusSet: [instance method 10](#)

## R

---

removeAllObjects [instance method 10](#)  
removeObject: [instance method 11](#)

## S

---

setSet: [instance method 11](#)  
setWithCapacity: [class method 7](#)

## U

---

unionSet: [instance method 11](#)