

---

# NSIndexSet Class Reference

Data Management: Data Types & Collections



2009-08-28



Apple Inc.  
© 2009 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, Bonjour, iPhone, Mac, and Mac OS 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

## **NSIndexSet Class Reference** 5

---

Overview	5
Adopted Protocols	6
Tasks	6
Creating Index Sets	6
Querying Index Sets	6
Comparing Index Sets	7
Getting Indexes	7
Enumerating Indexes	8
Class Methods	8
indexSet	8
indexSetWithIndex:	8
indexSetWithIndexesInRange:	9
Instance Methods	9
containsIndex:	9
containsIndexes:	10
containsIndexesInRange:	10
count	11
countOfIndexesInRange:	11
enumerateIndexesInRange:options:usingBlock:	11
enumerateIndexesUsingBlock:	12
enumerateIndexesWithOptions:usingBlock:	12
firstIndex	13
getIndexes:maxCount:inIndexRange:	13
indexesInRange:options:passingTest:	14
indexesPassingTest:	15
indexesWithOptions:passingTest:	16
indexGreaterThanIndex:	16
indexGreaterThanOrEqualToIndex:	17
indexInRange:options:passingTest:	17
indexLessThanIndex:	18
indexLessThanOrEqualToIndex:	18
indexPassingTest:	19
indexWithOptions:passingTest:	19
init	20
initWithIndex:	20
initWithIndexesInRange:	21
initWithIndexSet:	21
intersectsIndexesInRange:	22
isEqualToIndexSet:	22
lastIndex	23

**Document Revision History 25**

---

# NSIndexSet Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSCopying NSMutableCopying NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in iOS 2.0 and later.
<b>Companion guide</b>	Collections Programming Topics
<b>Declared in</b>	NSIndexSet.h
<b>Related sample code</b>	BonjourWeb

## Overview

The `NSIndexSet` class represents an immutable collection of unique unsigned integers, known as **indexes** because of the way they are used. This collection is referred to as a **index set**.

You use index sets in your code to store indexes into some other data structure. For example, given an `NSArray` object, you could use an index set to identify a subset of objects in that array.

Each index value can appear only once in the index set. This is an important concept to understand and is why you would not use index sets to store an arbitrary collection of integer values. To illustrate how this works, if you created an `NSIndexSet` object with the values 4, 5, 2, and 5, the resulting set would only have the values 4, 5, and 2 in it. Because index values are always maintained in sorted order, the actual order of the values when you created the set would be 2, 4, and then 5.

In most cases, using an index set is more efficient than storing a collection of individual integers. Internally, the `NSIndexSet` class represents indexes using ranges. For maximum performance and efficiency, overlapping ranges in an index set are automatically coalesced—that is, ranges merge rather than overlap. Thus, the more contiguous the indexes in the set, the fewer ranges are required to specify those indexes.

The designated initializers of the `NSIndexSet` class are: [initWithIndexesInRange:](#) (page 21) and [initWithIndexSet:](#) (page 21).

You must not subclass the `NSIndexSet` class.

The mutable subclass of `NSIndexSet` is `NSMutableIndexSet`.

## Adopted Protocols

### NSCoding

- encodeWithCoder:
- initWithCoder:

### NSCopying

- copyWithZone:

### NSMutableCopying

- mutableCopyWithZone:

## Tasks

### Creating Index Sets

- + [indexSet](#) (page 8)  
Creates an empty index set.
- + [indexSetWithIndex:](#) (page 8)  
Creates an index set with an index.
- + [indexSetWithIndexesInRange:](#) (page 9)  
Creates an index set with an index range.
- [init](#) (page 20)  
Initializes an allocated [NSIndexSet](#) (page 5) object.
- [initWithIndex:](#) (page 20)  
Initializes an allocated [NSIndexSet](#) (page 5) object with an index.
- [initWithIndexesInRange:](#) (page 21)  
Initializes an allocated [NSIndexSet](#) (page 5) object with an index range.
- [initWithIndexSet:](#) (page 21)  
Initializes an allocated [NSIndexSet](#) (page 5) object with an index set.

### Querying Index Sets

- [containsIndex:](#) (page 9)  
Indicates whether the receiver contains a specific index.
- [containsIndexes:](#) (page 10)  
Indicates whether the receiver contains a superset of the indexes in another index set.
- [containsIndexesInRange:](#) (page 10)  
Indicates whether the receiver contains the indexes represented by an index range.
- [intersectsIndexesInRange:](#) (page 22)  
Indicates whether the receiver contains any of the indexes in a range.

- [count](#) (page 11)  
Returns the number of indexes in the receiver.
- [countOfIndexesInRange:](#) (page 11)  
Returns the number of indexes in the receiver that are members of a given range.
- [indexPassingTest:](#) (page 19)  
Returns the index of the first object that passes the predicate Block test.
- [indexesPassingTest:](#) (page 15)  
Returns an `NSIndexSet` containing the receiver's objects that pass the Block test.
- [indexWithOptions:passingTest:](#) (page 19)  
Returns the index of the first object that passes the predicate Block test using the specified enumeration options.
- [indexesWithOptions:passingTest:](#) (page 16)  
Returns an `NSIndexSet` containing the receiver's objects that pass the Block test using the specified enumeration options.
- [indexInRange:options:passingTest:](#) (page 17)  
Returns the index of the first object in the specified range that passes the predicate Block test.
- [indexesInRange:options:passingTest:](#) (page 14)  
Returns an `NSIndexSet` containing the receiver's objects in the specified range that pass the Block test.

## Comparing Index Sets

- [isEqualToIndexSet:](#) (page 22)  
Indicates whether the indexes in the receiver are the same indexes contained in another index set.

## Getting Indexes

- [firstIndex](#) (page 13)  
Returns either the first index in the receiver or the not-found indicator.
- [lastIndex](#) (page 23)  
Returns either the last index in the receiver or the not-found indicator.
- [indexLessThanIndex:](#) (page 18)  
Returns either the closest index in the receiver that is less than a specific index or the not-found indicator.
- [indexLessThanOrEqualToIndex:](#) (page 18)  
Returns either the closest index in the receiver that is less than or equal to a specific index or the not-found indicator.
- [indexGreaterThanOrEqualToIndex:](#) (page 17)  
Returns either the closest index in the receiver that is greater than or equal to a specific index or the not-found indicator.
- [indexGreaterThanIndex:](#) (page 16)  
Returns either the closest index in the receiver that is greater than a specific index or the not-found indicator.

- [getIndexes:maxCount:inIndexRange:](#) (page 13)  
The receiver fills an index buffer with the indexes contained both in the receiver and in an index range, returning the number of indexes copied.

## Enumerating Indexes

- [enumerateIndexesUsingBlock:](#) (page 12)  
Executes a given Block using each object in the receiver.
- [enumerateIndexesWithOptions:usingBlock:](#) (page 12)  
Executes a given Block over the receiver's indexes, using the specified enumeration options.
- [enumerateIndexesInRange:options:usingBlock:](#) (page 11)  
Executes a given Block using the indexes in the specified range, using the specified enumeration options.

## Class Methods

### indexSet

Creates an empty index set.

```
+ (id)indexSet
```

#### Return Value

[NSIndexSet](#) (page 5) object with no members.

#### Availability

Available in iOS 2.0 and later.

#### See Also

- [init](#) (page 20)

#### Declared In

`NSIndexSet.h`

### indexSetWithIndex:

Creates an index set with an index.

```
+ (id)indexSetWithIndex:(NSUInteger) index
```

#### Parameters

*index*

An index.

#### Return Value

[NSIndexSet](#) (page 5) object containing *index*.



**Availability**

Available in iOS 2.0 and later.

**See Also**

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

**Related Sample Code**

BonjourWeb

**Declared In**

NSIndexSet.h

**indexSetWithIndexesInRange:**

Creates an index set with an index range.

```
+ (id)indexSetWithIndexesInRange:(NSRange) indexRange
```

**Parameters**

*indexRange*

An index range.

**Return Value**

[NSIndexSet](#) (page 5) object containing *indexRange*.

**Availability**

Available in iOS 2.0 and later.

**See Also**

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

**Declared In**

NSIndexSet.h

## Instance Methods

**containsIndex:**

Indicates whether the receiver contains a specific index.

```
- (BOOL)containsIndex:(NSUInteger) index
```

**Parameters**

*index*

Index being inquired about.

**Return Value**

YES when the receiver contains *index*, NO otherwise.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [containsIndexes:](#) (page 10)
- [containsIndexesInRange:](#) (page 10)

**Declared In**

NSIndexSet.h

**containsIndexes:**

Indicates whether the receiver contains a superset of the indexes in another index set.

```
(BOOL)containsIndexes:(NSIndexSet *)indexSet
```

**Parameters***indexSet*

Index set being inquired about.

**Return Value**

YES when the receiver contains a superset of the indexes in *indexSet*, NO otherwise.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [containsIndex:](#) (page 9)
- [containsIndexesInRange:](#) (page 10)

**Declared In**

NSIndexSet.h

**containsIndexesInRange:**

Indicates whether the receiver contains the indexes represented by an index range.

```
(BOOL)containsIndexesInRange:(NSRange)indexRange
```

**Parameters***indexRange*

The index range being inquired about.

**Return Value**

YES when the receiver contains the indexes in *indexRange*, NO otherwise.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [containsIndex:](#) (page 9)
- [containsIndexes:](#) (page 10)
- [intersectsIndexesInRange:](#) (page 22)

**Declared In**

NSIndexSet.h

**count**

Returns the number of indexes in the receiver.

- (NSUInteger)count

**Return Value**

Number of indexes in the receiver.

**Availability**

Available in iOS 2.0 and later.

**See Also**

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

**Declared In**

NSIndexSet.h

**countOfIndexesInRange:**

Returns the number of indexes in the receiver that are members of a given range.

- (NSUInteger)countOfIndexesInRange:(NSRange) *indexRange*

**Parameters**

*indexRange*

Index range being inquired about.

**Return Value**

Number of indexes in the receiver that are members of *indexRange*.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [count](#) (page 11)

**Declared In**

NSIndexSet.h

**enumerateIndexesInRange:options:usingBlock:**

Executes a given Block using the indexes in the specified range, using the specified enumeration options.

- (void)enumerateIndexesInRange:(NSRange) *range* options:(NSEnumerationOptions) *opts*  
usingBlock:(void (^)(NSUInteger idx, BOOL \*stop)) *block*

**Parameters**

*range*

Index to enumerate.

*opts*

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order). See `NSEnumerationOptions` for the supported values.

*block*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to `YES` to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to `YES` within the Block.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

`NSIndexSet.h`

**enumerateIndexesUsingBlock:**

Executes a given Block using each object in the receiver.

```
- (void)enumerateIndexesUsingBlock:(void (^)(NSUInteger idx, BOOL *stop))block
```

**Parameters***block*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to `YES` to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to `YES` within the Block.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

`NSIndexSet.h`

**enumerateIndexesWithOptions:usingBlock:**

Executes a given Block over the receiver's indexes, using the specified enumeration options.

```
- (void)enumerateIndexesWithOptions:(NSEnumerationOptions)opts usingBlock:(void (^)(NSUInteger idx, BOOL *stop))block
```

**Parameters***opts*

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order). See `NSEnumerationOptions` for the supported values.

*block*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

`NSIndexSet.h`

**firstIndex**

Returns either the first index in the receiver or the not-found indicator.

```
- (NSUInteger)firstIndex
```

**Return Value**

First index in the receiver or `NSNotFound` when the receiver is empty.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [lastIndex](#) (page 23)

**Declared In**

`NSIndexSet.h`

**getIndexes:maxCount:inIndexRange:**

The receiver fills an index buffer with the indexes contained both in the receiver and in an index range, returning the number of indexes copied.

```
- (NSUInteger)getIndexes:(NSUInteger *)indexBuffer maxCount:(NSUInteger)bufferSize
inIndexRange:(NSRangePointer)indexRangePointer
```

**Parameters***indexBuffer*

Index buffer to fill.

*bufferSize*Maximum size of *indexBuffer*.*indexRange*Index range to compare with indexes in the receiver; `nil` represents all the indexes in the receiver. Indexes in the index range and in the receiver are copied to *indexBuffer*. On output, the range of indexes not copied to *indexBuffer*.**Return Value**Number of indexes placed in *indexBuffer*.**Discussion**You are responsible for allocating the memory required for *indexBuffer* and for releasing it later.

Suppose you have an index set with contiguous indexes from 1 to 100. If you use this method to request a range of (1, 100)—which represents the set of indexes 1 through 100—and specify a buffer size of 20, this method returns 20 indexes—1 through 20—in *indexBuffer* and sets *indexRange* to (21, 80)—which represents the indexes 21 through 100.

Use this method to retrieve entries quickly and efficiently from an index set. You can call this method repeatedly to retrieve blocks of index values and then process them. When doing so, use the return value and *indexRange* to determine when you have finished processing the desired indexes. When the return value is less than *bufferSize*, you have reached the end of the range.

**Availability**

Available in iOS 2.0 and later.

**Declared In**

NSIndexSet.h

**indexesInRange:options:passingTest:**

Returns an NSIndexSet containing the receiver's objects in the specified range that pass the Block test.

```
- (NSIndexSet *)indexesInRange:(NSRange)range options:(NSEnumerationOptions)opts
    passingTest:(BOOL (^)(NSUInteger idx, BOOL *stop))predicate
```

**Parameters***range*

The range of indexes to test.

*opts*A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order). See `NSEnumerationOptions` for the supported values.

*predicate*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The *stop* argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

**Return Value**

An `NSIndexSet` containing the indexes of the receiver that passed the predicate Block test.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

`NSIndexSet.h`

**indexesPassingTest:**

Returns an `NSIndexSet` containing the receiver's objects that pass the Block test.

- (`NSIndexSet *`)`indexesPassingTest:(BOOL (^)(NSUInteger idx, BOOL *stop))predicate`

**Parameters**

*predicate*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The *stop* argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

**Return Value**

An `NSIndexSet` containing the indexes of the receiver that passed the predicate Block test.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

`NSIndexSet.h`

## indexesWithOptions:passingTest:

Returns an `NSIndexSet` containing the receiver's objects that pass the Block test using the specified enumeration options.

```
- (NSIndexSet *)indexesWithOptions:(NSEnumerationOptions)opts passingTest:(BOOL (^)(NSUInteger idx, BOOL *stop))predicate
```

### Parameters

*opts*

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order). See `NSEnumerationOptions` for the supported values.

*predicate*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to `YES` to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to `YES` within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

### Return Value

An `NSIndexSet` containing the indexes of the receiver that passed the predicate Block test.

### Availability

Available in iOS 4.0 and later.

### Declared In

`NSIndexSet.h`

## indexGreaterThanIndex:

Returns either the closest index in the receiver that is greater than a specific index or the not-found indicator.

```
- (NSUInteger)indexGreaterThanIndex:(NSUInteger)index
```

### Parameters

*index*

Index being inquired about.

### Return Value

Closest index in the receiver greater than *index*; `NSNotFound` when the receiver contains no qualifying index.

### Availability

Available in iOS 2.0 and later.

### See Also

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



- [indexGreaterThanOrEqualToIndex:](#) (page 17)
- [indexLessThanOrEqualToIndex:](#) (page 18)

**Declared In**

NSIndexSet.h

**indexGreaterThanOrEqualToIndex:**

Returns either the closest index in the receiver that is greater than or equal to a specific index or the not-found indicator.

```
(NSUInteger)indexGreaterThanOrEqualToIndex:(NSUInteger) index
```

**Parameters***index*

Index being inquired about.

**Return Value**

Closest index in the receiver greater than or equal to *index*; `NSNotFound` when the receiver contains no qualifying index.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [indexGreaterThanIndex:](#) (page 16)
- [indexLessThanIndex:](#) (page 18)
- [indexLessThanOrEqualToIndex:](#) (page 18)

**Declared In**

NSIndexSet.h

**indexInRange:options:passingTest:**

Returns the index of the first object in the specified range that passes the predicate Block test.

```
(NSUInteger)indexInRange:(NSRange) range options:(NSEnumerationOptions) opts  
passingTest:(BOOL (^)(NSUInteger idx, BOOL *stop)) predicate
```

**Parameters***range*

The range of indexes to test.

*opts*

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order). See `NSEnumerationOptions` for the supported values.

*predicate*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The *stop* argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

**Return Value**

The index of the first object that passes the predicate test.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

NSIndexSet.h

**indexLessThanIndex:**

Returns either the closest index in the receiver that is less than a specific index or the not-found indicator.

- (NSUInteger)indexLessThanIndex:(NSUInteger) *index*

**Parameters***index*

Index being inquired about.

**Return Value**

Closest index in the receiver less than *index*; NSNotFound when the receiver contains no qualifying index.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [indexGreaterThanIndex:](#) (page 16)
- [indexGreaterThanOrEqualToIndex:](#) (page 17)
- [indexLessThanOrEqualToIndex:](#) (page 18)

**Declared In**

NSIndexSet.h

**indexLessThanOrEqualToIndex:**

Returns either the closest index in the receiver that is less than or equal to a specific index or the not-found indicator.

- (NSUInteger)indexLessThanOrEqualToIndex:(NSUInteger) *index*

**Parameters***index*

Index being inquired about.

**Return Value**Closest index in the receiver less than or equal to *index*; `NSNotFound` when the receiver contains no qualifying index.**Availability**

Available in iOS 2.0 and later.

**See Also**

- [indexGreaterThanIndex:](#) (page 16)
- [indexLessThanIndex:](#) (page 18)
- [indexGreaterThanOrEqualToIndex:](#) (page 17)

**Declared In**

NSIndexSet.h

**indexPassingTest:**

Returns the index of the first object that passes the predicate Block test.

```
- (NSUInteger)indexPassingTest:(BOOL (^)(NSUInteger idx, BOOL *stop))predicate
```

**Parameters***predicate*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*A reference to a Boolean value. The block can set the value to `YES` to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to `YES` within the Block.The Block returns a Boolean value that indicates whether *obj* passed the test.**Return Value**

The index of the first object that passes the predicate test.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

NSIndexSet.h

**indexWithOptions:passingTest:**

Returns the index of the first object that passes the predicate Block test using the specified enumeration options.

```
- (NSUInteger)indexWithOptions:(NSEnumerationOptions)opts passingTest:(BOOL
 (^)(NSUInteger idx, BOOL *stop))predicate
```

**Parameters***opts*

A bitmask that specifies the options for the enumeration (whether it should be performed concurrently and whether it should be performed in reverse order). See `NSEnumerationOptions` for the supported values.

*predicate*

The Block to apply to elements in the set.

The Block takes two arguments:

*idx*

The index of the object.

*stop*

A reference to a Boolean value. The block can set the value to YES to stop further processing of the set. The `stop` argument is an out-only argument. You should only ever set this Boolean to YES within the Block.

The Block returns a Boolean value that indicates whether *obj* passed the test.

**Return Value**

The index of the first object that passes the predicate test.

**Availability**

Available in iOS 4.0 and later.

**Declared In**

`NSIndexSet.h`

**init**

Initializes an allocated [NSIndexSet](#) (page 5) object.

```
- (id)init
```

**Return Value**

Initialized, empty [NSIndexSet](#) (page 5) object.

**Availability**

Available in iOS 2.0 and later.

**See Also**

+ [indexSet](#) (page 8)

**Declared In**

`NSIndexSet.h`

**initWithIndex:**

Initializes an allocated [NSIndexSet](#) (page 5) object with an index.

- (id)initWithIndex:(NSUInteger) *index*

**Parameters**

*index*

An index.

**Return Value**

Initialized [NSIndexSet](#) (page 5) object with *index*.

**Availability**

Available in iOS 2.0 and later.

**See Also**

+ [indexSetWithIndex:](#) (page 8)

**Declared In**

NSIndexSet.h

**initWithIndexesInRange:**

Initializes an allocated [NSIndexSet](#) (page 5) object with an index range.

- (id)initWithIndexesInRange:(NSRange) *indexRange*

**Parameters**

*indexRange*

An index range. Must include only indexes representable as unsigned integers.

**Return Value**

Initialized [NSIndexSet](#) (page 5) object with *indexRange*.

**Discussion**

This method raises an `NSRangeException` when *indexRange* would add an index that exceeds the maximum allowed value for unsigned integers.

This method is a designated initializer for [NSIndexSet](#) (page 5).

**Availability**

Available in iOS 2.0 and later.

**See Also**

+ [indexSetWithIndexesInRange:](#) (page 9)

**Declared In**

NSIndexSet.h

**initWithIndexSet:**

Initializes an allocated [NSIndexSet](#) (page 5) object with an index set.

- (id)initWithIndexSet:(NSIndexSet \*) *indexSet*

**Parameters***indexSet*

An index set.

**Return Value**Initialized [NSIndexSet](#) (page 5) object with *indexSet*.**Discussion**This method is a designated initializer for [NSIndexSet](#) (page 5).**Availability**

Available in iOS 2.0 and later.

**Declared In**

NSIndexSet.h

**intersectsIndexesInRange:**

Indicates whether the receiver contains any of the indexes in a range.

- (BOOL)intersectsIndexesInRange:(NSRange) *indexRange***Parameters***indexRange*

Index range being inquired about.

**Return Value**YES when the receiver contains one or more of the indexes in *indexRange*, NO otherwise.**Availability**

Available in iOS 2.0 and later.

**See Also**- [containsIndexesInRange:](#) (page 10)**Declared In**

NSIndexSet.h

**isEqualToIndexSet:**

Indicates whether the indexes in the receiver are the same indexes contained in another index set.

- (BOOL)isEqualToIndexSet:(NSIndexSet \*) *indexSet***Parameters***indexSet*

Index set being inquired about.

**Return Value**YES when the indexes in the receiver are the same indexes *indexSet* contains, NO otherwise.**Availability**

Available in iOS 2.0 and later.

**Declared In**

NSIndexSet.h

**lastIndex**

Returns either the last index in the receiver or the not-found indicator.

- (NSUInteger)lastIndex

**Return Value**

Last index in the receiver or `NSNotFound` when the receiver is empty.

**Availability**

Available in iOS 2.0 and later.

**See Also**

- [firstIndex](#) (page 13)

**Declared In**

NSIndexSet.h





# Document Revision History

---

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

Date	Notes
2009-08-28	Updated for Mac OS X v 10.6. Added new methods supporting Blocks.
2007-03-24	Updated for Mac OS X v10.5.
	Added <a href="#">countOfIndexesInRange:</a> (page 11).
	Added details to <a href="#">getIndexes:maxCount:inIndexRange:</a> (page 13) to correct and clarify the given example.
	Corrected a code example.
2006-05-23	First publication of this content as a separate document.

**REVISION HISTORY**

Document Revision History