
NSObject Protocol Reference

[Cocoa > Objective-C Language](#)



2008-12-22



Apple Inc.
© 2008 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, iChat, Mac, Mac OS, Objective-C, and Quartz 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

NSObject Protocol Reference 5

Overview	5
Tasks	5
Identifying Classes	5
Identifying and Comparing Objects	6
Managing Reference Counts	6
Testing Object Inheritance, Behavior, and Conformance	6
Describing Objects	6
Sending Messages	7
Determining Allocation Zones	7
Identifying Proxies	7
Instance Methods	7
autorelease	7
class	8
conformsToProtocol:	8
description	8
hash	9
isEqual:	9
isKindOfClass:	10
isMemberOfClass:	11
isProxy	12
performSelector:	12
performSelector:withObject:	13
performSelector:withObject:withObject:	13
release	14
respondsToSelector:	15
retain	16
retainCount	17
self	17
superclass	18
zone	18

Document Revision History 21

Index 23

NSObject Protocol Reference

Adopted by	NSObject
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSObject.h
Companion guides	Cocoa Fundamentals Guide Memory Management Programming Guide for Cocoa

Overview

The `NSObject` protocol groups methods that are fundamental to all Objective-C objects.

If an object conforms to this protocol, it can be considered a first-class object. Such an object can be asked about its:

- Class, and the place of its class in the inheritance hierarchy
- Conformance to protocols
- Ability to respond to a particular message

In addition, objects that conform to this protocol—with its [retain](#) (page 16), [release](#) (page 14), and [autorelease](#) (page 7) methods—can also integrate with the object management and deallocation scheme defined in Foundation (for more information see, for example, *Memory Management Programming Guide for Cocoa*). Thus, an object that conforms to the `NSObject` protocol can be managed by container objects like those defined by `NSArray` and `NSDictionary`.

The Cocoa root class, `NSObject`, adopts this protocol, so all objects inheriting from `NSObject` have the features described by this protocol.

Tasks

Identifying Classes

- `class` (page 8)
Returns the class object for the receiver's class.

- [superclass](#) (page 18)
Returns the class object for the receiver's superclass.

Identifying and Comparing Objects

- [isEqual:](#) (page 9)
Returns a Boolean value that indicates whether the receiver and a given object are equal.
- [hash](#) (page 9)
Returns an integer that can be used as a table address in a hash table structure.
- [self](#) (page 17)
Returns the receiver.

Managing Reference Counts

- [retain](#) (page 16)
Increments the receiver's reference count.
- [release](#) (page 14)
Decrements the receiver's reference count.
- [autorelease](#) (page 7)
Adds the receiver to the current autorelease pool.
- [retainCount](#) (page 17)
Returns the receiver's reference count.

Testing Object Inheritance, Behavior, and Conformance

- [isKindOfClass:](#) (page 10)
Returns a Boolean value that indicates whether the receiver is an instance of given class or an instance of any class that inherits from that class.
- [isMemberOfClass:](#) (page 11)
Returns a Boolean value that indicates whether the receiver is an instance of a given class.
- [respondsToSelector:](#) (page 15)
Returns a Boolean value that indicates whether the receiver implements or inherits a method that can respond to a specified message.
- [conformsToProtocol:](#) (page 8)
Returns a Boolean value that indicates whether the receiver conforms to a given protocol.

Describing Objects

- [description](#) (page 8)
Returns a string that describes the contents of the receiver.

Sending Messages

- [performSelector:](#) (page 12)
Sends a specified message to the receiver and returns the result of the message.
- [performSelector:withObject:](#) (page 13)
Sends a message to the receiver with an object as the argument.
- [performSelector:withObject:withObject:](#) (page 13)
Sends a message to the receiver with two objects as arguments.

Determining Allocation Zones

- [zone](#) (page 18)
Returns a pointer to the zone from which the receiver was allocated.

Identifying Proxies

- [isProxy](#) (page 12)
Returns a Boolean value that indicates whether the receiver does not descend from NSObject.

Instance Methods

autorelease

Adds the receiver to the current autorelease pool.

```
- (id)autorelease
```

Return Value

self.

Discussion

You add an object to an autorelease pool so it will receive a `release` message—and thus might be deallocated—when the pool is destroyed. For more information on the autorelease mechanism, see *Memory Management Programming Guide for Cocoa*.

Special Considerations

If garbage collection is enabled, this method is a no-op.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [retain](#) (page 16)

Related Sample Code

CoreRecipes

GridCalendar

iSpend
Mountains
PDFKitLinker2

Declared In
NSObject.h

class

Returns the class object for the receiver's class.

```
- (Class)class
```

Return Value
The class object for the receiver's class.

Availability
Available in Mac OS X v10.0 and later.

See Also
+ class (NSObject class)

Declared In
NSObject.h

conformsToProtocol:

Returns a Boolean value that indicates whether the receiver conforms to a given protocol.

```
- (BOOL)conformsToProtocol:(Protocol *)aProtocol
```

Parameters

aProtocol

A protocol object that represents a particular protocol.

Return Value
YES if the receiver conforms to *aProtocol*, otherwise NO.

Discussion

This method works identically to the `conformsToProtocol:` class method declared in `NSObject`. It's provided as a convenience so that you don't need to get the class object to find out whether an instance can respond to a given set of messages.

Availability
Available in Mac OS X v10.0 and later.

Declared In
NSObject.h

description

Returns a string that describes the contents of the receiver.

- (NSString *)description

Return Value

A string that describes the contents of the receiver.

Discussion

The debugger's print-object command indirectly invokes this method to produce a textual description of an object.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSObject.h

hash

Returns an integer that can be used as a table address in a hash table structure.

- (NSInteger)hash

Return Value

An integer that can be used as a table address in a hash table structure.

Discussion

If two objects are equal (as determined by the [isEqual:](#) (page 9) method), they must have the same hash value. This last point is particularly important if you define `hash` in a subclass and intend to put instances of that subclass into a collection.

If a mutable object is added to a collection that uses hash values to determine the object's position in the collection, the value returned by the `hash` method of the object must not change while the object is in the collection. Therefore, either the `hash` method must not rely on any of the object's internal state information or you must make sure the object's internal state information does not change while the object is in the collection. Thus, for example, a mutable dictionary can be put in a hash table but you must not change it while it is in there. (Note that it can be difficult to know whether or not a given object is in a collection.)

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Related Sample Code

ImageMapExample

TrackBall

Declared In

NSObject.h

isEqual:

Returns a Boolean value that indicates whether the receiver and a given object are equal.

- (BOOL)isEqual:(id)anObject

Parameters

anObject

The object to be compared to the receiver.

Return Value

YES if the receiver and *anObject* are equal, otherwise NO.

Discussion

This method defines what it means for instances to be equal. For example, a container object might define two containers as equal if their corresponding objects all respond YES to an `isEqual:` request. See the NSData, NSDictionary, NSArray, and NSString class specifications for examples of the use of this method.

If two objects are equal, they must have the same hash value. This last point is particularly important if you define `isEqual:` in a subclass and intend to put instances of that subclass into a collection. Make sure you also define `hash` (page 9) in your subclass.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [hash](#) (page 9)

Related Sample Code

IdentitySample

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

Declared In

NSObject.h

isKindOfClass:

Returns a Boolean value that indicates whether the receiver is an instance of given class or an instance of any class that inherits from that class.

- (BOOL)isKindOfClass:(Class)aClass

Parameters

aClass

A class object representing the Objective-C class to be tested.

Return Value

YES if the receiver is an instance of *aClass* or an instance of any class that inherits from *aClass*, otherwise NO.

Discussion

For example, in this code, `isKindOfClass:` would return YES because, in Foundation, the `NSArchiver` class inherits from `NSCoder`:

```
NSMutableData *myData = [NSMutableData dataWithCapacity:30];
id anArchiver = [[NSArchiver alloc] initWithWritingWithMutableData:myData];
if ( [anArchiver isKindOfClass:[NSCoder class]] )
```

```
...
```

Be careful when using this method on objects represented by a class cluster. Because of the nature of class clusters, the object you get back may not always be the type you expected. If you call a method that returns a class cluster, the exact type returned by the method is the best indicator of what you can do with that object. For example, if a method returns a pointer to an `NSArray` object, you should not use this method to see if the array is mutable, as shown in the following code:

```
// DO NOT DO THIS!
if ([myArray isKindOfClass:[NSMutableArray class]])
{
    // Modify the object
}
```

If you use such constructs in your code, you might think it is alright to modify an object that in reality should not be modified. Doing so might then create problems for other code that expected the object to remain unchanged.

If the receiver is a class object, this method returns `YES` if `aClass` is a `Class` object of the same type, `NO` otherwise.

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Declared In

`NSObject.h`

isMemberOfClass:

Returns a Boolean value that indicates whether the receiver is an instance of a given class.

```
- (BOOL)isMemberOfClass:(Class)aClass
```

Parameters

aClass

A class object representing the Objective-C class to be tested.

Return Value

`YES` if the receiver is an instance of *aClass*, otherwise `NO`.

Discussion

For example, in this code, `isMemberOfClass:` would return `NO`:

```
NSMutableData *myData = [NSMutableData dataWithCapacity:30];
id anArchiver = [[NSArchiver alloc] initWithWritingWithMutableData:myData];
if ([anArchiver isMemberOfClass:[NSCoder class]])
    ...
```

Class objects may be compiler-created objects but they still support the concept of membership. Thus, you can use this method to verify that the receiver is a specific `Class` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Declared In

NSObject.h

isProxy

Returns a Boolean value that indicates whether the receiver does not descend from NSObject.

- (BOOL)isProxy

Return Value

NO if the receiver really descends from NSObject, otherwise YES.

Discussion

This method is necessary because sending [isKindOfClass:](#) (page 10) or [isMemberOfClass:](#) (page 11) to an `NSProxy` object will test the object the proxy stands in for, not the proxy itself. Use this method to test if the receiver is a proxy (or a member of some other root class).

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSObject.h

performSelector:

Sends a specified message to the receiver and returns the result of the message.

- (id)performSelector:(SEL)aSelector

Parameters

aSelector

A selector identifying the message to send. If *aSelector* is NULL, an `NSInvalidArgumentException` is raised.

Return Value

An object that is the result of the message.

Discussion

The `performSelector:` method is equivalent to sending an *aSelector* message directly to the receiver. For example, all three of the following messages do the same thing:

```
id myClone = [anObject copy];
id myClone = [anObject performSelector:@selector(copy)];
id myClone = [anObject performSelector:sel_getUid("copy")];
```

However, the `performSelector:` method allows you to send messages that aren't determined until runtime. A variable selector can be passed as the argument:

```
SEL myMethod = findTheAppropriateSelectorForTheCurrentSituation();
[anObject performSelector:myMethod];
```

The *aSelector* argument should identify a method that takes no arguments. For methods that return anything other than an object, use `NSInvocation`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [performSelector:withObject:](#) (page 13)
- [performSelector:withObject:withObject:](#) (page 13)

Declared In

`NSObject.h`

performSelector:withObject:

Sends a message to the receiver with an object as the argument.

```
- (id)performSelector:(SEL)aSelector withObject:(id)anObject
```

Parameters

aSelector

A selector identifying the message to send. If *aSelector* is `NULL`, an `NSInvalidArgumentException` is raised.

anObject

An object that is the sole argument of the message.

Return Value

An object that is the result of the message.

Discussion

This method is the same as [performSelector:](#) (page 12) except that you can supply an argument for *aSelector*. *aSelector* should identify a method that takes a single argument of type `id`. For methods with other argument types and return values, use `NSInvocation`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [performSelector:withObject:withObject:](#) (page 13)
- `methodForSelector:` (`NSObject` class)

Related Sample Code

`SearchField`

`ToolbarSample`

Declared In

`NSObject.h`

performSelector:withObject:withObject:

Sends a message to the receiver with two objects as arguments.

```
- (id)performSelector:(SEL)aSelector withObject:(id)anObject
withObject:(id)anotherObject
```

Parameters*aSelector*

A selector identifying the message to send. If *aSelector* is NULL, an `NSInvalidArgumentException` is raised.

anObject

An object that is the first argument of the message.

anotherObject

An object that is the second argument of the message

Return Value

An object that is the result of the message.

Discussion

This method is the same as [performSelector:](#) (page 12) except that you can supply two arguments for *aSelector*. *aSelector* should identify a method that can take two arguments of type `id`. For methods with other argument types and return values, use `NSInvocation`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [performSelector:withObject:](#) (page 13)
- `methodForSelector:` (NSObject class)

Declared In

NSObject.h

release

Decrements the receiver's reference count.

```
- (oneway void)release
```

Discussion

The receiver is sent a `dealloc` message when its reference count reaches 0.

You would only implement this method to define your own reference-counting scheme. Such implementations should not invoke the inherited method; that is, they should not include a release message to `super`.

For more information on the reference counting mechanism, see *Memory Management Programming Guide for Cocoa*.

Special Considerations

If garbage collection is enabled, this method is a no-op.

You must complete the object initialization (using an `init` method) before invoking `release`. For example, the following code shows an error:

```
id anObject = [MyObject alloc];
[anObject release];
```

You may call `release` from within an `init` method if initialization fails for some reason provided that you have at least called superclass's designated initializer.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CoreRecipes

Quartz Composer WWDC 2005 TextEdit

Sketch-112

TextEditPlus

WhackedTV

Declared In

NSObject.h

respondsToSelector:

Returns a Boolean value that indicates whether the receiver implements or inherits a method that can respond to a specified message.

```
- (BOOL)respondToSelector:(SEL)aSelector
```

Parameters

aSelector

A selector that identifies a message.

Return Value

YES if the receiver implements or inherits a method that can respond to *aSelector*, otherwise NO.

Discussion

The application is responsible for determining whether a NO response should be considered an error.

You cannot test whether an object inherits a method from its superclass by sending `respondToSelector:` to the object using the `super` keyword. This method will still be testing the object as a whole, not just the superclass's implementation. Therefore, sending `respondToSelector: to super` is equivalent to sending it to `self`. Instead, you must invoke the NSObject class method `instancesRespondToSelector:` directly on the object's superclass, as illustrated in the following code fragment.

```
if( [MySuperclass instancesRespondToSelector:@selector(aMethod)] ) {
    // invoke the inherited method
    [super aMethod];
}
```

You cannot simply use `[[self superclass] instancesRespondToSelector:@selector(aMethod)]` since this may cause the method to fail if it is invoked by a subclass.

Note that if the receiver is able to forward *aSelector* messages to another object, it will be able to respond to the message, albeit indirectly, even though this method returns NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [forwardInvocation:](#) (NSObject class)
- + [instancesRespondToSelector:](#) (NSObject class)

Declared In

NSObject.h

retain

Increments the receiver's reference count.

- (id)retain

Return Value

self.

Discussion

You send an object a `retain` message when you want to prevent it from being deallocated without your express permission.

An object is deallocated automatically when its reference count reaches 0. `retain` messages increment the reference count, and [release](#) (page 14) messages decrement it. For more information on this mechanism, see *Memory Management Programming Guide for Cocoa*.

As a convenience, `retain` returns `self` because it is often used in nested expressions:

```
NSString *systemApps = [[NSString
    stringWithCString: "/Applications"] retain];
```

You would implement this method only if you were defining your own reference-counting scheme. Such implementations must return `self` and should not invoke the inherited method by sending a `retain` message to `super`.

Special Considerations

If garbage collection is enabled, this method is a no-op.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [autorelease](#) (page 7)
- [release](#) (page 14)

Related Sample Code

CIAnnotation
CoreRecipes
GridCalendar
Reducer
Sketch-112

Declared In

NSObject.h

retainCount

Returns the receiver's reference count.

- (NSUInteger)retainCount

Return Value

The receiver's reference count.

Discussion

You might override this method in a class to implement your own reference-counting scheme. For objects that never get released (that is, their [release](#) (page 14) method does nothing), this method should return `UINT_MAX`, as defined in `<limits.h>`.

The `retainCount` method does not account for any pending [autorelease](#) (page 7) messages sent to the receiver.

Important: This method is typically of no value in debugging memory management issues. Because any number of framework objects may have retained an object in order to hold references to it, while at the same time autorelease pools may be holding any number of deferred releases on an object, it is very unlikely that you can get useful information from this method.

To understand the fundamental rules of memory management that you must abide by, read [Memory Management Rules](#). To diagnose memory management problems, use a suitable tool:

- The [LLVM/Clang Static analyzer](#) can typically find memory management problems even before you run your program.
- The Object Alloc instrument in the Instruments application (see *Instruments User Guide*) can track object allocation and destruction.
- Shark (see *Shark User Guide*) also profiles memory allocations (amongst numerous other aspects of your program).

Special Considerations

If garbage collection is enabled, the return value is undefined.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [autorelease](#) (page 7)
- [retain](#) (page 16)

Related Sample Code

SimpleThreads

Declared In

NSObject.h

self

Returns the receiver.

- (id)self

Return Value

The receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [class](#) (page 8)

Related Sample Code

GridCalendar

iSpend

Quartz Composer WWDC 2005 TextEdit

StickiesExample

TextEditPlus

Declared In

NSObject.h

superclass

Returns the class object for the receiver's superclass.

- (Class)superclass

Return Value

The class object for the receiver's superclass.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [superclass](#) (NSObject class)

Declared In

NSObject.h

zone

Returns a pointer to the zone from which the receiver was allocated.

- (NSZone *)zone

Return Value

A pointer to the zone from which the receiver was allocated.

Discussion

Objects created without specifying a zone are allocated from the default zone.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ allocWithZone: (NSObject class)

Related Sample Code

QTCoreVideo102

QTCoreVideo103

QTCoreVideo201

QTCoreVideo202

QTCoreVideo301

Declared In

NSObject.h

Document Revision History

This table describes the changes to *NSObject Protocol Reference*.

Date	Notes
2008-12-22	Updated discussion of retainCount method.
2007-07-19	Updated definition of release, added a clarification to -hash, and added links to companion documents.
2006-06-28	Updated to conform to reference consistency guidelines.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`autorelease` protocol instance method [7](#)

C

`class` protocol instance method [8](#)
`conformsToProtocol:` protocol instance method [8](#)

D

`description` protocol instance method [8](#)

H

`hash` protocol instance method [9](#)

I

`isEqual:` protocol instance method [9](#)
`isKindOfClass:` protocol instance method [10](#)
`isMemberOfClass:` protocol instance method [11](#)
`isProxy` protocol instance method [12](#)

P

`performSelector:` protocol instance method [12](#)
`performSelector:withObject:` protocol instance method [13](#)
`performSelector:withObject:withObject:` protocol instance method [13](#)

R

`release` protocol instance method [14](#)
`respondToSelector:` protocol instance method [15](#)
`retain` protocol instance method [16](#)
`retainCount` protocol instance method [17](#)

S

`self` protocol instance method [17](#)
`superclass` protocol instance method [18](#)

Z

`zone` protocol instance method [18](#)