
NSProxy Class Reference

[Cocoa](#) > [Objective-C Language](#)



2007-04-06



Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, 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

NSProxy Class Reference 5

Overview	5
Adopted Protocols	5
Tasks	6
Creating Instances	6
Deallocating Instances	6
Finalizing an Object	6
Handling Unimplemented Methods	6
Introspecting a Proxy Class	7
Describing a Proxy Class or Object	7
Class Methods	7
alloc	7
allocWithZone:	7
class	8
respondsToSelector:	8
Instance Methods	8
dealloc	8
description	9
finalize	9
forwardInvocation:	9
methodSignatureForSelector:	10

Document Revision History 11

Index 13

NSProxy Class Reference

Inherits from	none (NSProxy is a root class)
Conforms to	NSObject
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Distributed Objects Programming Topics
Declared in	NSProxy.h

Overview

`NSProxy` is an abstract superclass defining an API for objects that act as stand-ins for other objects or for objects that don't exist yet. Typically, a message to a proxy is forwarded to the real object or causes the proxy to load (or transform itself into) the real object. Subclasses of `NSProxy` can be used to implement transparent distributed messaging (for example, `NSDistantObject`) or for lazy instantiation of objects that are expensive to create.

`NSProxy` implements the basic methods required of a root class, including those defined in the `NSObject` protocol. However, as an abstract class it doesn't provide an initialization method, and it raises an exception upon receiving any message it doesn't respond to. A concrete subclass must therefore provide an initialization or creation method and override the [forwardInvocation:](#) (page 9) and [methodSignatureForSelector:](#) (page 10) methods to handle messages that it doesn't implement itself. A subclass's implementation of [forwardInvocation:](#) (page 9) should do whatever is needed to process the invocation, such as forwarding the invocation over the network or loading the real object and passing it the invocation. [methodSignatureForSelector:](#) (page 10) is required to provide argument type information for a given message; a subclass's implementation should be able to determine the argument types for the messages it needs to forward and should construct an `NSMethodSignature` object accordingly. See the `NSDistantObject`, `NSInvocation`, and `NSMethodSignature` class specifications for more information.

Adopted Protocols

- `NSObject`
- autorelease
 - class
 - conformsToProtocol:

- description
- hash
- isEqual:
- isKindOfClass:
- isKindOfClass:
- isProxy
- performSelector:
- performSelector:withObject:
- performSelector:withObject:withObject:
- release
- respondsToSelector:
- retain
- retainCount
- self
- superclass
- zone

Tasks

Creating Instances

- + [alloc](#) (page 7)
Returns a new instance of the receiving class
- + [allocWithZone:](#) (page 7)
Returns a new instance of the receiving class

Deallocating Instances

- [dealloc](#) (page 8)
Deallocates the memory occupied by the receiver.

Finalizing an Object

- [finalize](#) (page 9)
The garbage collector invokes this method on the receiver before disposing of the memory it uses.

Handling Unimplemented Methods

- [forwardInvocation:](#) (page 9)
Passes a given invocation to the real object the proxy represents.

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

Raises `NSInvalidArgumentException`. Override this method in your concrete subclass to return a proper `NSMethodSignature` object for the given selector and the class your proxy objects stand in for.

Introspecting a Proxy Class

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

Returns a Boolean value that indicates whether the receiving class responds to a given selector.

Describing a Proxy Class or Object

+ [class](#) (page 8)

Returns `self` (the class object).

- [description](#) (page 9)

Returns an `NSString` object containing the real class name and the `id` of the receiver as a hexadecimal number.

Class Methods

alloc

Returns a new instance of the receiving class

+ (id)alloc

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSProxy.h

allocWithZone:

Returns a new instance of the receiving class

+ (id)allocWithZone:(NSZone *)zone

Return Value

A new instance of the receiving class, as described in the `NSObject` class specification under the `allocWithZone:` class method.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSProxy.h

class

Returns `self` (the class object).

```
+ (Class)class
```

Return Value

`self`. Because this is a class method, it returns the class object

Availability

Available in Mac OS X v10.0 and later.

See Also

`class (NSObject)`

`class (NSObject protocol)`

Declared In

`NSProxy.h`

respondsToSelector:

Returns a Boolean value that indicates whether the receiving class responds to a given selector.

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

Parameters

aSelector

A selector.

Return Value

YES if the receiving class responds to *aSelector* messages, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSProxy.h`

Instance Methods

dealloc

Deallocates the memory occupied by the receiver.

```
- (void)dealloc
```

Discussion

This method behaves as described in the `NSObject` class specification under the `dealloc` instance method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [finalize](#) (page 9)

Declared In

NSProxy.h

description

Returns an `NSString` object containing the real class name and the `id` of the receiver as a hexadecimal number.

- (`NSString *`)description

Return Value

An `NSString` object containing the real class name and the `id` of the receiver as a hexadecimal number.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSProxy.h

finalize

The garbage collector invokes this method on the receiver before disposing of the memory it uses.

- (`void`)finalize

Discussion

This method behaves as described in the `NSObject` class specification under the `finalize` instance method. Note that a `finalize` method must be thread-safe.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [dealloc](#) (page 8)

Declared In

NSProxy.h

forwardInvocation:

Passes a given invocation to the real object the proxy represents.

- (`void`)forwardInvocation:(`NSInvocation *`)*anInvocation*

Parameters

anInvocation

The invocation to forward.

Discussion

NSProxy's implementation merely raises `NSInvalidArgumentException`. Override this method in your subclass to handle *anInvocation* appropriately, at the very least by setting its return value.

For example, if your proxy merely forwards messages to an instance variable named *realObject*, it can implement `forwardInvocation:` like this:

```
- (void)forwardInvocation:(NSInvocation *)anInvocation
{
    [anInvocation setTarget:realObject];
    [anInvocation invoke];
    return;
}
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSProxy.h

methodSignatureForSelector:

Raises `NSInvalidArgumentException`. Override this method in your concrete subclass to return a proper `NSMethodSignature` object for the given selector and the class your proxy objects stand in for.

```
- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector
```

Parameters

aSelector

The selector for which to return a method signature.

Return Value

Not applicable. The implementation provided by NSProxy raises an exception.

Discussion

Be sure to avoid an infinite loop when necessary by checking that *aSelector* isn't the selector for this method itself and by not sending any message that might invoke this method.

For example, if your proxy merely forwards messages to an instance variable named *realObject*, it can implement `methodSignatureForSelector:` like this:

```
- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector
{
    return [realObject methodSignatureForSelector:aSelector];
}
```

Availability

Available in Mac OS X v10.0 and later.

See Also

`methodSignatureForSelector:` (NSObject)

Declared In

NSProxy.h

Document Revision History

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

Date	Notes
2007-04-06	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

alloc [class method 7](#)
allocWithZone: [class method 7](#)

C

class [class method 8](#)

D

dealloc [instance method 8](#)
description [instance method 9](#)

F

finalize [instance method 9](#)
forwardInvocation: [instance method 9](#)

M

methodSignatureForSelector: [instance method 10](#)

R

respondsToSelector: [class method 8](#)