

---

# NSURLConnection Class Reference

[Cocoa](#) > [Interapplication Communication](#)



2008-02-08



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, 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

## NSConnection Class Reference 5

---

Overview	5
Tasks	5
Getting the Default Instance	5
Creating Instances	6
Running the Connection in a New Thread	6
Vending a Service	6
Getting a Remote Object	7
Getting a Conversation	7
Getting All NSConnection Objects	7
Configuring Instances	7
Getting Ports	8
Getting Statistics	8
Setting the Delegate	8
Authenticating	8
Responding to a Connection	9
Class Methods	9
allConnections	9
connectionWithReceivePort:sendPort:	9
connectionWithRegisteredName:host:	10
connectionWithRegisteredName:host:usingNameServer:	11
currentConversation	11
defaultConnection	11
rootProxyForConnectionWithRegisteredName:host:	12
rootProxyForConnectionWithRegisteredName:host:usingNameServer:	13
serviceConnectionWithName:rootObject:	13
serviceConnectionWithName:rootObject:usingNameServer:	14
Instance Methods	15
addRequestMode:	15
addRunLoop:	15
delegate	16
enableMultipleThreads	16
independentConversationQueueing	16
initWithReceivePort:sendPort:	17
invalidate	18
isValid	18
localObjects	19
multipleThreadsEnabled	19
receivePort	20
registerName:	20
registerName:withNameServer:	21

- remoteObjects 22
- removeRequestMode: 22
- removeRunLoop: 22
- replyTimeout 23
- requestModes 23
- requestTimeout 24
- rootObject 24
- rootProxy 24
- runInNewThread 25
- sendPort 25
- setDelegate: 26
- setIndependentConversationQueueing: 26
- setReplyTimeout: 27
- setRequestTimeout: 27
- setRootObject: 27
- statistics 28
- Delegate Methods 28
  - authenticateComponents:withData: 28
  - authenticationDataForComponents: 29
  - connection:handleRequest: 30
  - connection:shouldMakeNewConnection: 30
  - createConversationForConnection: 31
  - makeNewConnection:sender: 31
- Constants 32
  - NSConnection run loop mode 32
  - Connection Exception Names 32
- Notifications 32
  - NSConnectionDidDieNotification 32
  - NSConnectionDidInitializeNotification 33

---

**Document Revision History 35**

---

**Index 37**

---

# NSConnection Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	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>	Distributed Objects Programming Topics
<b>Declared in</b>	NSConnection.h
<b>Related sample code</b>	SimpleThreads TrivialThreads

## Overview

An `NSConnection` object manages the communication between objects in different threads or between a thread and a process running on a local or remote system. Connection objects form the backbone of the distributed objects mechanism and normally operate in the background. You use the methods of `NSConnection` explicitly when vending an object to other applications, when accessing such a vended object through a proxy, and when altering default communication parameters. At other times, you simply interact with a vended object or its proxy.

In Mac OS X v10.5 and later, a single connection object may be shared by multiple threads and used to access a vended object by default. Prior to Mac OS X v10.5, a separate connection object must be maintained by each thread by default; however, an application can enable sharing by invoking the `enableMultipleThreads` method of the object.

## Tasks

### Getting the Default Instance

+ [defaultConnection](#) (page 11)

Returns the default `NSConnection` object for the current thread.

## Creating Instances

- + [connectionWithReceivePort:sendPort:](#) (page 9)  
Returns an `NSConnection` object that communicates using given send and receive ports.
- [initWithReceivePort:sendPort:](#) (page 17)  
Returns an `NSConnection` object initialized with given send and receive ports.

## Running the Connection in a New Thread

- [runInNewThread](#) (page 25)  
Creates and starts a new `NSThread` object and then runs the receiving connection in the new thread.
- [enableMultipleThreads](#) (page 16)  
Configures the receiver to allow requests from multiple threads to the remote object, without requiring each thread to each maintain its own connection.
- [multipleThreadsEnabled](#) (page 19)  
Returns a Boolean value that indicates whether the receiver supports requests from multiple threads.
- [addRunLoop:](#) (page 15)  
Adds the specified run loop to the list of run loops the receiver monitors and from which it responds to requests.
- [removeRunLoop:](#) (page 22)  
Removes a given `NSRunLoop` object from the list of run loops the receiver monitors and from which it responds to requests.

## Vending a Service

- + [serviceConnectionWithName:rootObject:usingNameServer:](#) (page 14)  
Creates and returns a new connection object representing a vended service on the specified port name server.
- + [serviceConnectionWithName:rootObject:](#) (page 13)  
Creates and returns a new connection object representing a vended service on the default system port name server.
- [registerName:](#) (page 20)  
Registers the specified service using with the default system port name server.
- [registerName:withNameServer:](#) (page 21)  
Registers a service with the specified port name server.
- [setRootObject:](#) (page 27)  
Sets the object that the receiver makes available to other applications or threads.
- [rootObject](#) (page 24)  
Returns the object that the receiver (or its parent) makes available to other applications or threads.

## Getting a Remote Object

- + [connectionWithRegisteredName:host:](#) (page 10)  
Returns the `NSConnection` object whose send port links it to the `NSConnection` object registered with the default `NSPortNameServer` under a given name on a given host.
- + [connectionWithRegisteredName:host:usingNameServer:](#) (page 11)  
Returns the `NSConnection` object whose send port links it to the `NSConnection` object registered under a given name with a given server on a given host.
- [rootProxy](#) (page 24)  
Returns the proxy for the root object of the receiver's peer in another application or thread.
- + [rootProxyForConnectionWithRegisteredName:host:](#) (page 12)  
Returns a proxy for the root object of the `NSConnection` object registered with the default `NSPortNameServer` under a given name on a given host.
- + [rootProxyForConnectionWithRegisteredName:host:usingNameServer:](#) (page 13)  
Returns a proxy for the root object of the `NSConnection` object registered with `server` under `name` on a given host.
- [remoteObjects](#) (page 22)  
Returns all the local proxies for remote objects that have been received over the connection but not deallocated yet.
- [localObjects](#) (page 19)  
Returns the local objects that have been sent over the connection and still have proxies at the other end.

## Getting a Conversation

- + [currentConversation](#) (page 11)  
Returns a token object representing any conversation in progress in the current thread.

## Getting All NSConnection Objects

- + [allConnections](#) (page 9)  
Returns all valid `NSConnection` objects in the process.

## Configuring Instances

- [setRequestTimeout:](#) (page 27)  
Sets the timeout interval for outgoing remote messages.
- [requestTimeout](#) (page 24)  
Returns the timeout interval for outgoing remote messages.
- [setReplyTimeout:](#) (page 27)  
Sets the timeout interval for replies to outgoing remote messages
- [replyTimeout](#) (page 23)  
Returns the timeout interval for replies to outgoing remote messages.

- [setIndependentConversationQueueing:](#) (page 26)  
Sets a Boolean value that specifies whether the receiver handles remote messages atomically.
- [independentConversationQueueing](#) (page 16)  
Returns a Boolean value that indicates whether the receiver handles remote messages atomically.
- [addRequestMode:](#) (page 15)  
Adds *mode* to the set of run-loop input modes that the receiver uses for connection requests.
- [removeRequestMode:](#) (page 22)  
Removes *mode* from the set of run-loop input modes the receiver uses for connection requests.
- [requestModes](#) (page 23)  
Returns the set of request modes the receiver's receive port is registered for with its NSRunLoop object.
- [invalidate](#) (page 18)  
Invalidates (but doesn't release) the receiver.
- [isValid](#) (page 18)  
Returns a Boolean value that indicates whether the receiver is known to be valid.

## Getting Ports

- [receivePort](#) (page 20)  
Returns the NSPort object on which the receiver receives incoming network messages.
- [sendPort](#) (page 25)  
Returns the NSPort object that the receiver sends outgoing network messages through.

## Getting Statistics

- [statistics](#) (page 28)  
Returns an NSDictionary object containing various statistics for the receiver.

## Setting the Delegate

- [setDelegate:](#) (page 26)  
Sets the receiver's delegate.
- [delegate](#) (page 16)  
Returns the receiver's delegate.

## Authenticating

- [authenticateComponents:withData:](#) (page 28) *delegate method*  
Returns a Boolean value that indicates whether given authentication data is valid for a given set of components.
- [authenticationDataForComponents:](#) (page 29) *delegate method*  
Returns an NSData object to be used as an authentication stamp for an outgoing message.



## Responding to a Connection

- `connection:shouldMakeNewConnection:` (page 30) *delegate method*  
Returns a Boolean value that indicates whether the parent connection should allow a given new connection to be created.
- `connection:handleRequest:` (page 30) *delegate method*  
This method should be implemented by `NSConnection` object delegates that want to intercept distant object requests.
- `createConversationForConnection:` (page 31) *delegate method*  
Returns an arbitrary object identifying a new conversation being created for the connection in the current thread.
- `makeNewConnection:sender:` (page 31) *delegate method*  
Returns a Boolean value that indicates whether the parent should allow a given new connection to be created and configured.

## Class Methods

### **allConnections**

Returns all valid `NSConnection` objects in the process.

```
+ (NSArray *)allConnections
```

#### **Return Value**

An array containing all valid `NSConnection` objects in the process.

#### **Availability**

Available in Mac OS X v10.0 and later.

#### **See Also**

- `isValid` (page 18)

#### **Declared In**

`NSConnection.h`

### **connectionWithReceivePort:sendPort:**

Returns an `NSConnection` object that communicates using given send and receive ports.

```
+ (id)connectionWithReceivePort:(NSPort *)receivePort sendPort:(NSPort *)sendPort
```

#### **Parameters**

*receivePort*

A receive port.

*sendPort*

A send port.

#### **Return Value**

An `NSConnection` object that communicates using *receivePort* and *sendPort*.

**Discussion**

See [initWithReceivePort:sendPort:](#) (page 17) for more information.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [defaultConnection](#) (page 11)

**Related Sample Code**

SimpleThreads

TrivialThreads

**Declared In**

NSConnection.h

**connectionWithRegisteredName:host:**

Returns the `NSConnection` object whose send port links it to the `NSConnection` object registered with the default `NSPortNameServer` under a given name on a given host.

```
+ (id)connectionWithRegisteredName:(NSString *)name host:(NSString *)hostName
```

**Parameters**

*name*

The name of an `NSConnection` object.

*hostName*

The name of the host. The domain name *hostName* is an Internet domain name (for example, "sales.anycorp.com"). If *hostName* is `nil` or empty, then only the local host is searched for the named `NSConnection` object.

**Return Value**

The `NSConnection` object whose send port links it to the `NSConnection` object registered with the default `NSPortNameServer` under *name* on the host named *hostName*. Returns `nil` if no `NSConnection` object can be found for *name* and *hostName*.

The returned `NSConnection` object is a child of the default `NSConnection` object for the current thread (that is, it shares the default `NSConnection` object's receive port).

**Discussion**

To get the object vended by the `NSConnection` object, use the [rootProxy](#) (page 24) instance method. The [rootProxyForConnectionWithRegisteredName:host:](#) (page 12) class method immediately returns this object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [defaultConnection](#) (page 11)

+ [connectionWithRegisteredName:host:usingNameServer:](#) (page 11)

**Declared In**

NSConnection.h

## connectionWithRegisteredName:host:usingNameServer:

Returns the `NSConnection` object whose send port links it to the `NSConnection` object registered under a given name with a given server on a given host.

```
+ (id)connectionWithRegisteredName:(NSString *)name host:(NSString *)hostName
    usingNameServer:(NSPortNameServer *)server
```

### Parameters

*name*

The connection name.

*hostName*

The host name.

*server*

The name server.

### Return Value

The `NSConnection` object whose send port links it to the `NSConnection` object registered with *server* under *name* on the host named *hostName*.

### Discussion

See [connectionWithRegisteredName:host:](#) (page 10) for more information.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`NSConnection.h`

## currentConversation

Returns a token object representing any conversation in progress in the current thread.

```
+ (id)currentConversation
```

### Return Value

A token object representing any conversation in progress in the current thread, or `nil` if there is no conversation in progress.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [createConversationForConnection:](#) (page 31)

### Declared In

`NSConnection.h`

## defaultConnection

Returns the default `NSConnection` object for the current thread.

```
+ (NSConnection *)defaultConnection
```

**Return Value**

The default `NSConnection` object for the current thread, creating it if necessary.

**Discussion**

The default `NSConnection` object uses a single `NSPort` object for both receiving and sending and is useful only for vending an object; use the [setRootObject:](#) (page 27) and [registerName:](#) (page 20) methods to do this.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

**rootProxyForConnectionWithRegisteredName:host:**

Returns a proxy for the root object of the `NSConnection` object registered with the default `NSPortNameServer` under a given name on a given host.

```
+ (NSDistantObject *)rootProxyForConnectionWithRegisteredName:(NSString *)name
  host:(NSString *)hostName
```

**Parameters**

*name*

The name under which the connection is registered.

*hostName*

The host name. The domain name *hostName* is an Internet domain name (for example, "sales.anycorp.com"). If *hostName* is `nil` or empty, then only the local host is searched for the named `NSConnection` object.

**Return Value**

a proxy for the root object of the `NSConnection` object registered with the default `NSPortNameServer` under *name* on the host named *hostName*, or `nil` if that `NSConnection` object has no root object set. Also returns `nil` if no `NSConnection` object can be found for *name* and *hostName*.

**Discussion**

The `NSConnection` object of the returned proxy is a child of the default `NSConnection` object for the current thread (that is, it shares the default `NSConnection` object's receive port).

This method invokes [connectionWithRegisteredName:host:](#) (page 10) and sends the resulting `NSConnection` object a [rootProxy](#) (page 24) message.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setRootObject:](#) (page 27)

+ [rootProxyForConnectionWithRegisteredName:host:usingNameServer:](#) (page 13)

**Declared In**

`NSConnection.h`

**rootProxyForConnectionWithRegisteredName:host:usingNameServer:**

Returns a proxy for the root object of the `NSConnection` object registered with `server` under `name` on a given host.

```
+ (NSDistantObject *)rootProxyForConnectionWithRegisteredName:(NSString *)name
    host:(NSString *)hostName usingNameServer:(NSPortNameServer *)server
```

**Parameters**

*name*

The name of an `NSConnection` object .

*hostName*

A host name.

*server*

The server.

**Return Value**

A proxy for the root object of the `NSConnection` object registered with `server` under `name` on the host named `hostName`, or `nil` if that `NSConnection` object has no root object set.

**Discussion**

See [rootProxyForConnectionWithRegisteredName:host:](#) (page 12) for more information.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

**serviceConnectionWithName:rootObject:**

Creates and returns a new connection object representing a vended service on the default system port name server.

```
+ (id)serviceConnectionWithName:(NSString *)name rootObject:(id)root
```

**Parameters**

*name*

The name of the service you want to publish.

*root*

The object to use as the root object for the published service. This is the object vended by the connection.

**Return Value**

An `NSConnection` object representing the vended service or `nil` if there was a problem setting up the connection object.

**Discussion**

This method creates the server-side of a connection object and registers it with the default system port name server. Clients wishing to connect to this service can request a communications port from the same port server and use that port to communicate.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- + [serviceConnectionWithName:rootObject:usingNameServer:](#) (page 14)
- + [connectionWithRegisteredName:host:](#) (page 10)
- [rootObject](#) (page 24)
- + [systemDefaultPortNameServer](#) (NSPortNameServer)

**Declared In**

NSConnection.h

**serviceConnectionWithName:rootObject:usingNameServer:**

Creates and returns a new connection object representing a vended service on the specified port name server.

```
+ (id)serviceConnectionWithName:(NSString *)name rootObject:(id)root
    usingNameServer:(NSPortNameServer *)server
```

**Parameters***name*

The name of the service you want to publish.

*root*

The object to use as the root object for the published service. This is the object vended by the connection.

*server*

The port name server with which to register your service.

**Return Value**

An `NSConnection` object representing the vended service or `nil` if there was a problem setting up the connection object.

**Discussion**

This method creates the server-side of a connection object and registers it with the specified port name server. Clients wishing to connect to this service can request a communications port from the same port server and use that port to communicate.

If the specified service name corresponds to a service that is autolaunched by `launchd`, this method allows the service to check in with the `launchd` process. If the service is not autolaunched by `launchd`, this method registers the new connection with the specified name. For more information about `launchd` and its role in launching services, see *System Startup Programming Topics*.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- + [connectionWithRegisteredName:host:usingNameServer:](#) (page 11)
- [rootObject](#) (page 24)

**Declared In**

NSConnection.h

## Instance Methods

### addRequestMode:

Adds *mode* to the set of run-loop input modes that the receiver uses for connection requests.

```
- (void)addRequestMode:(NSString *)mode
```

#### Parameters

*mode*

The mode to add to the receiver.

#### Discussion

The default input mode is `NSDefaultRunLoopMode`. See the `NSRunLoop` class specification for more information on input modes.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

`addPort:forMode:` (`NSRunLoop`)

#### Declared In

`NSConnection.h`

### addRunLoop:

Adds the specified run loop to the list of run loops the receiver monitors and from which it responds to requests.

```
- (void)addRunLoop:(NSRunLoop *)runloop
```

#### Parameters

*runloop*

The run loop to add to the receiver.

#### Discussion

This method is invoked automatically when a request comes in from a new run loop if `enableMultipleThreads` (page 16) has been set.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- `enableMultipleThreads` (page 16)

- `removeRunLoop:` (page 22)

#### Declared In

`NSConnection.h`

## delegate

Returns the receiver's delegate.

- (id)delegate

### Return Value

The receiver's delegate.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setDelegate:](#) (page 26)

### Declared In

NSConnection.h

## enableMultipleThreads

Configures the receiver to allow requests from multiple threads to the remote object, without requiring each thread to each maintain its own connection.

- (void)enableMultipleThreads

### Discussion

In Mac OS X v10.5 and later, multiple thread support is enabled by default and this method does nothing.

Prior to Mac OS X v10.5, multiple thread support is disabled by default and must be enabled explicitly. When disabled, each thread must create its own `NSConnection` object in order to access a given remote object. When enabled, threads may use the same `NSConnection` object to access the remote object. If this feature is disabled and an attempt is made to connect to the receiver from a thread other than the one that created it, the receiver raises an `NSObjectInaccessibleException`.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [multipleThreadsEnabled](#) (page 19)

### Declared In

NSConnection.h

## independentConversationQueueing

Returns a Boolean value that indicates whether the receiver handles remote messages atomically.

- (BOOL)independentConversationQueueing

### Return Value

YES if the receiver handles remote messages atomically, otherwise NO.

### Discussion

See [Configuring an NSConnection](#) for more information on independent conversation queueing.



**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setIndependentConversationQueueing:](#) (page 26)

**Declared In**

NSConnection.h

**initWithReceivePort:sendPort:**

Returns an `NSConnection` object initialized with given send and receive ports.

```
- (id)initWithReceivePort:(NSPort *)receivePort sendPort:(NSPort *)sendPort
```

**Parameters**

*receivePort*

The receive port for the new connection.

*sendPort*

The send port for the new connection.

**Return Value**

An `NSConnection` object initialized with *receivePort* and *sendPort*. The returned object might be different than the original receiver.

**Discussion**

The new `NSConnection` object adds *receivePort* to the current `NSRunLoop` object with `NSDefaultRunLoopMode` as the mode. If the application doesn't use an `NSApplication` object to handle events, it needs to run the `NSRunLoop` object with one of its various `run...` messages.

This method posts an `NSConnectionDidInitializeNotification` (page 33) once the connection is initialized.

The *receivePort* and *sendPort* parameters affect initialization as follows:

- If an `NSConnection` object with the same ports already exists, releases the receiver, retains the existing connection, and returns it.
- If an `NSConnection` object exists that uses the same ports, but switched in role, then the new `NSConnection` object communicates with it. Messages sent to a proxy held by either connection are forwarded through the other `NSConnection` object. This rule applies both within and across address spaces.  
This behavior is useful for setting up distributed object connections between threads within an application. See [Communicating With Distributed Objects](#) for more information.
- If *receivePort* and *sendPort* are `nil`, deallocates the receiver and returns `nil`.
- If *receivePort* is `nil`, the `NSConnection` object allocates and uses a new port of the same class as *sendPort*.
- If *sendPort* is `nil` or if both ports are the same, the `NSConnection` object uses *receivePort* for both sending and receiving and is useful only for vending an object. Use the [registerName:](#) (page 20) and [setRootObject:](#) (page 27) instance methods to vend an object.

- If an `NSConnection` object exists that uses `receivePort` as both of its ports, it's treated as the parent of the new `NSConnection` object, and its root object and all its configuration settings are applied to the new `NSConnection` object. You should neither register a name for nor set the root object of the new `NSConnection` object. See [Configuring an NSConnection](#) for more information.
- If `receivePort` and `sendPort` are different and neither is shared with another `NSConnection` object, the receiver can be used to vend an object as well as to communicate with other `NSConnection` objects. However, it has no other `NSConnection` object to communicate with until one is set up.
- The `receivePort` parameter can't be shared by `NSConnection` objects in different threads.

This method is the designated initializer for the `NSConnection` class.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

+ [defaultConnection](#) (page 11)

**Related Sample Code**

SimpleThreads

TrivialThreads

**Declared In**

`NSConnection.h`

## invalidate

Invalidates (but doesn't release) the receiver.

- (void)invalidate

**Discussion**

After withdrawing the ports the receiver has registered with the current run loop, `invalidate` posts an [NSConnectionDidDieNotification](#) (page 32) and then invalidates all remote objects and exported local proxies.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [isValid](#) (page 18)

`removePort:forMode:(NSRunLoop)`

- [requestModes](#) (page 23)

**Declared In**

`NSConnection.h`

## isValid

Returns a Boolean value that indicates whether the receiver is known to be valid.

- (BOOL)isValid

**Return Value**

YES if the receiver is known to be valid, otherwise NO.

**Discussion**

An `NSConnection` object becomes invalid when either of its ports becomes invalid, but only notes that it has become invalid when it tries to send or receive a message. When this happens it posts an [NSConnectionDidDieNotification](#) (page 32) to the default notification center.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [invalidate](#) (page 18)
- `isValid` (`NSPort`)

**Declared In**

`NSConnection.h`

## localObjects

Returns the local objects that have been sent over the connection and still have proxies at the other end.

- (`NSArray *`)`localObjects`

**Return Value**

An array containing the local objects that have been sent over the connection and still have proxies at the other end.

**Discussion**

When an object's remote proxy is deallocated, a message is sent back to the receiver to notify it that the local object is no longer shared over the connection.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [remoteObjects](#) (page 22)

**Declared In**

`NSConnection.h`

## multipleThreadsEnabled

Returns a Boolean value that indicates whether the receiver supports requests from multiple threads.

- (`BOOL`)`multipleThreadsEnabled`

**Return Value**

YES if the receiver supports requests from multiple threads.

**Discussion**

In Mac OS X v10.5 and later, multiple threads are enabled by default.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [enableMultipleThreads](#) (page 16)

**Declared In**

NSConnection.h

**receivePort**

Returns the `NSPort` object on which the receiver receives incoming network messages.

- (NSPort \*)receivePort

**Return Value**

The `NSPort` object on which the receiver receives incoming network messages.

**Discussion**

You can inspect this object for debugging purposes or use it to create another `NSConnection` object, but shouldn't use it to send or receive messages explicitly. Don't set the delegate of the receive port; it already has a delegate established by the `NSConnection` object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [sendPort](#) (page 25)

- [initWithReceivePort:sendPort:](#) (page 17)

**Declared In**

NSConnection.h

**registerName:**

Registers the specified service using with the default system port name server.

- (BOOL)registerName:(NSString \*)name

**Parameters**

*name*

The name under which to register the receiver.

**Return Value**

YES if the operation was successful, otherwise NO (for example, if another `NSConnection` object on the same host is already registered under *name*).

**Discussion**

This method connects the receive port of the receiving `NSConnection` object with the specified service name. It registers the name using the port name server returned by the `systemDefaultPortNameServer` method of `NSPortNameServer`. If the operation is successful, other `NSConnection` objects can contact the receiver using the [connectionWithRegisteredName:host:](#) (page 10) and [rootProxyForConnectionWithRegisteredName:host:](#) (page 12) class methods.

If the receiver was already registered under a name and this method returns `NO`, the old name remains in effect. If this method is successful, it also unregisters the old name.

To unregister an `NSConnection` object, simply invoke `registerName:` and supply `nil` as the connection name.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [setRootObject:](#) (page 27)
- [registerName:withNameServer:](#) (page 21)
- + `systemDefaultPortNameServer` (`NSPortNameServer`)

**Declared In**

`NSConnection.h`

**registerName:withNameServer:**

Registers a service with the specified port name server.

- (BOOL)registerName:(NSString \*)name withNameServer:(NSPortNameServer \*)server

**Parameters**

*name*

The name under which to register the receiver.

*server*

The name server.

**Return Value**

YES if the operation was successful, otherwise NO (for example, if another `NSConnection` object on the same host is already registered under *name*).

**Discussion**

This method connects the receive port of the receiving `NSConnection` object with the specified service name. If the operation is successful, other `NSConnection` objects can contact the receiver using the [connectionWithRegisteredName:host:](#) (page 10) and [rootProxyForConnectionWithRegisteredName:host:](#) (page 12) class methods.

If the receiver was already registered under a name and this method returns `NO`, the old name remains in effect. If this method is successful, it also unregisters the old name.

To unregister an `NSConnection` object, simply invoke `registerName:` and supply `nil` as the connection name.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

## remoteObjects

Returns all the local proxies for remote objects that have been received over the connection but not deallocated yet.

- (NSArray \*)remoteObjects

### Return Value

An array containing all the local proxies for remote objects that have been received over the connection but not deallocated yet.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [localObjects](#) (page 19)

### Declared In

NSConnection.h

## removeRequestMode:

Removes *mode* from the set of run-loop input modes the receiver uses for connection requests.

- (void)removeRequestMode:(NSString \*)mode

### Parameters

*mode*

The mode to remove from the set of run-loop input modes the receiver uses for connection requests.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [requestModes](#) (page 23)

removePort:forMode:(NSRunLoop)

### Declared In

NSConnection.h

## removeRunLoop:

Removes a given NSRunLoop object from the list of run loops the receiver monitors and from which it responds to requests.

- (void)removeRunLoop:(NSRunLoop \*)runloop

### Parameters

*runloop*

The run loop to remove from the receiver.

### Availability

Available in Mac OS X v10.0 and later.

**See Also**

- [addRunLoop:](#) (page 15)

**Declared In**

NSConnection.h

## replyTimeout

Returns the timeout interval for replies to outgoing remote messages.

- (NSTimeInterval)replyTimeout

**Return Value**

The timeout interval for replies to outgoing remote messages.

**Discussion**

If a non-oneway remote message is sent and no reply is received by the timeout, an `NSPortTimeoutException` is raised.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [requestTimeout](#) (page 24)  
 - [setReplyTimeout:](#) (page 27)

**Declared In**

NSConnection.h

## requestModes

Returns the set of request modes the receiver's receive port is registered for with its `NSRunLoop` object.

- (NSArray \*)requestModes

**Return Value**

An array of `NSString` objects that represents the set of request modes the receiver's receive port is registered for with its `NSRunLoop` object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [addRequestMode:](#) (page 15)  
[addPort:forMode:\(NSRunLoop\)](#)  
 - [removeRequestMode:](#) (page 22)

**Declared In**

NSConnection.h

## requestTimeout

Returns the timeout interval for outgoing remote messages.

- (NSTimeInterval)requestTimeout

### Return Value

The timeout interval for outgoing remote messages.

### Discussion

If a remote message can't be sent before the timeout, an `NSPortTimeoutException` is raised.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [replyTimeout](#) (page 23)
- [setRequestTimeout:](#) (page 27)

### Declared In

`NSConnection.h`

## rootObject

Returns the object that the receiver (or its parent) makes available to other applications or threads.

- (id)rootObject

### Return Value

The object that the receiver (or its parent) makes available to other applications or threads, or `nil` if there is no root object.

### Discussion

To get a proxy to this object in another application or thread, invoke the [rootProxyForConnectionWithRegisteredName:host:](#) (page 12) class method with the appropriate arguments.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [rootProxy](#) (page 24)
- [setRootObject:](#) (page 27)

### Declared In

`NSConnection.h`

## rootProxy

Returns the proxy for the root object of the receiver's peer in another application or thread.

- (NSDistantObject \*)rootProxy



### Return Value

The proxy for the root object of the receiver's peer in another application or thread.

### Discussion

The proxy returned can change between invocations if the peer `NSConnection` object's root object is changed.

**Note:** If the `NSConnection` object uses separate send and receive ports and has no peer, when you invoke `rootProxy` it will block for the duration of the reply timeout interval, waiting for a reply.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [rootObject](#) (page 24)

### Related Sample Code

SimpleThreads

TrivialThreads

### Declared In

`NSConnection.h`

## runInNewThread

Creates and starts a new `NSThread` object and then runs the receiving connection in the new thread.

- (void)runInNewThread

### Discussion

If the newly created thread is the first to be detached from the current thread, this method posts an `NSWillBecomeMultiThreadedNotification` with `nil` to the default notification center.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`NSConnection.h`

## sendPort

Returns the `NSPort` object that the receiver sends outgoing network messages through.

- (NSPort \*)sendPort

### Return Value

The `NSPort` object that the receiver sends outgoing network messages through.

### Discussion

You can inspect this object for debugging purposes or use it to create another `NSConnection` object, but shouldn't use it to send or receive messages explicitly. Don't set the delegate of the send port; it already has a delegate established by the `NSConnection` object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [receivePort](#) (page 20)
- [initWithReceivePort:sendPort:](#) (page 17)

**Declared In**

NSConnection.h

**setDelegate:**

Sets the receiver's delegate.

- (void)setDelegate:(id)*anObject*

**Parameters**

*anObject*

The receiver's delegate.

**Discussion**

A connection's delegate can process incoming messages itself instead of letting NSConnection object handle them. The delegate can also authenticate messages and accept, deny, or modify new connections.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSConnection.h

**setIndependentConversationQueueing:**

Sets a Boolean value that specifies whether the receiver handles remote messages atomically.

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

**Parameters**

*flag*

YES if the receiver handles remote messages atomically, otherwise NO.

**Discussion**

The default is NO. An NSConnection object normally forwards remote message to the intended recipients as they come in. See [Configuring an NSConnection](#) for more information.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [independentConversationQueueing](#) (page 16)

**Declared In**

NSConnection.h

## setReplyTimeout:

Sets the timeout interval for replies to outgoing remote messages

- (void)setReplyTimeout:(NSTimeInterval)seconds

### Parameters

*seconds*

The timeout interval for replies to outgoing remote messages.

### Discussion

If a non-oneway remote message is sent and no reply is received by the timeout, an `NSPortTimeoutException` is raised. The default timeout is the maximum possible value.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setRequestTimeout:](#) (page 27)
- [replyTimeout](#) (page 23)

### Declared In

`NSConnection.h`

## setRequestTimeout:

Sets the timeout interval for outgoing remote messages.

- (void)setRequestTimeout:(NSTimeInterval)seconds

### Parameters

*seconds*

The timeout interval for outgoing remote messages.

### Discussion

If a remote message can't be sent before the timeout, an `NSPortTimeoutException` is raised. The default timeout is the maximum possible value.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setReplyTimeout:](#) (page 27)
- [requestTimeout](#) (page 24)

### Declared In

`NSConnection.h`

## setRootObject:

Sets the object that the receiver makes available to other applications or threads.

- (void)setRootObject:(id)anObject

**Parameters***anObject*

The root object for the receiver.

**Discussion**

This only affects new connection requests and [rootProxy](#) (page 24) messages to established `NSConnection` objects; applications that have proxies to the old root object can still send messages through it.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**- [rootObject](#) (page 24)**Related Sample Code**

SimpleThreads

**Declared In**`NSConnection.h`**statistics**Returns an `NSDictionary` object containing various statistics for the receiver.- (`NSDictionary *`)`statistics`**Return Value**

An `NSDictionary` object containing various statistics for the receiver, such as the number of vended objects, the number of requests and replies, and so on.

**Discussion**

The statistics dictionary should be used only for debugging purposes.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**`NSConnection.h`

## Delegate Methods

**authenticateComponents:withData:**

Returns a Boolean value that indicates whether given authentication data is valid for a given set of components.

- (`BOOL`)`authenticateComponents:(NSArray *)components withData:(NSData *)authenticationData`

**Parameters**

*components*

An array that contains `NSData` and `NSPort` objects belonging to an `NSPortMessage` object. See the `NSPortMessage` class specification for more information.

*authenticationData*

Authentication data created by the delegate of the peer `NSConnection` object with [authenticationDataForComponents:](#) (page 29).

**Return Value**

YES if the *authenticationData* provided is valid for *components*, otherwise NO.

**Discussion**

Use this message for validation of incoming messages. An `NSConnection` object raises an `NSFailedAuthenticationException` on receipt of a remote message the delegate doesn't authenticate.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

**authenticationDataForComponents:**

Returns an `NSData` object to be used as an authentication stamp for an outgoing message.

- (`NSData *`)`authenticationDataForComponents:(NSArray *)components`

**Parameters**

*components*

An array containing the elements of a network message, in the form of `NSPort` and `NSData` objects.

**Return Value**

An `NSData` object to be used as an authentication stamp for an outgoing message.

**Discussion**

The delegate should use only the `NSData` elements to create the authentication stamp. See the `NSPortMessage` class specification for more information on the components.

If [authenticationDataForComponents:](#) (page 29) returns `nil`, an `NSGenericException` will be raised. If the delegate determines that the message shouldn't be authenticated, it should return an empty `NSData` object. The delegate on the other side of the connection must then be prepared to accept an empty `NSData` object as the second parameter to [authenticateComponents:withData:](#) (page 28) and to handle the situation appropriately.

The *components* parameter will be validated on receipt by the delegate of the peer `NSConnection` object with [authenticateComponents:withData:](#) (page 28).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

**connection:handleRequest:**

This method should be implemented by `NSConnection` object delegates that want to intercept distant object requests.

```
- (BOOL)connection:(NSConnection *)conn handleRequest:(NSDistantObjectRequest *)doReq
```

**Parameters**

*conn*

The connection object for which the receiver is the delegate.

*doReq*

The distant object request.

**Return Value**

YES if the request was handled by the delegate, NO if the request should proceed as if the delegate did not intercept it.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

**connection:shouldMakeNewConnection:**

Returns a Boolean value that indicates whether the parent connection should allow a given new connection to be created.

```
- (BOOL)connection:(NSConnection *)parentConnection shouldMakeNewConnection:(NSConnection *)newConnection
```

**Parameters**

*parentConnection*

The connection object for which the receiver is the delegate.

*newConnection*

The new connection.

**Return Value**

YES if *parentConnection* should allow *newConnection* to be created and set up, NO if *parentConnection* should refuse and immediately release *newConnection*.

**Discussion**

Use this method to limit the amount of `NSConnection` objects created in your application or to change the parameters of child `NSConnection` objects.

Use [NSConnectionDidInitializeNotification](#) (page 33) instead of this delegate method if possible.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`NSConnection.h`

## createConversationForConnection:

Returns an arbitrary object identifying a new conversation being created for the connection in the current thread.

```
- (id)createConversationForConnection:(NSConnection *)conn
```

### Parameters

*conn*

The connection object for which the receiver is the delegate.

### Return Value

An arbitrary object identifying a new conversation being created for the connection in the current thread.

### Discussion

New conversations are created only if [independentConversationQueueing](#) (page 16) is YES for *conn*. If you do not implement this method, `NSConnection` object creates an instance of `NSObject`.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

+ [currentConversation](#) (page 11)

`conversation` (`NSDistantObjectRequest`)

### Declared In

`NSConnection.h`

## makeNewConnection:sender:

Returns a Boolean value that indicates whether the parent should allow a given new connection to be created and configured.

```
- (BOOL)makeNewConnection:(NSConnection *)newConnection sender:(NSConnection *)parentConnection
```

### Parameters

*newConnection*

The new connection.

*parentConnection*

The parent connection.

### Return Value

YES if *parentConnection* should allow *newConnection* to be created and configured, NO if *parentConnection* should refuse and immediately release *newConnection*.

### Discussion

Use this method to limit the amount of `NSConnection` objects created in your application or to change the parameters of child `NSConnection` objects.

Use [NSConnectionDidInitializeNotification](#) (page 33) instead of this delegate method if possible.

### Availability

Available in Mac OS X v10.0 and later.

**Declared In**  
 NSConnection.h

## Constants

### NSConnection run loop mode

`NSConnection` defines the following run loop mode—see `NSRunLoop` for more details.

```
extern NSString *NSConnectionReplyMode;
```

#### Constants

`NSConnectionReplyMode`

The mode to indicate an `NSConnection` object waiting for replies.

You should rarely need to use this mode.

Declared in `NSConnection.h`.

Available in Mac OS X v10.0 and later.

#### Declared In

`Foundation/NSConnection.h`

### Connection Exception Names

The name of an exception raised in case of authentication failure.

```
extern NSString *NSFailedAuthenticationException;
```

#### Constants

`NSFailedAuthenticationException`

Raised by `NSConnection` on receipt of a remote message the delegate doesn't authenticate.

Available in Mac OS X v10.0 and later.

Declared in `NSConnection.h`.

#### Declared In

`Foundation/NSConnection.h`

## Notifications

### NSConnectionDidDieNotification

Posted when an `NSConnection` object is deallocated or when it's notified that its `NSPort` object has become invalid. The notification object is the `NSConnection` object. This notification does not contain a `userInfo` dictionary.

An `NSConnection` object attached to a remote `NSSocketPort` object cannot detect when the remote port becomes invalid, even if the remote port is on the same machine. Therefore, it cannot post this notification when the connection is lost. Instead, you must detect the timeout error when the next message is sent.



The `NSConnection` object posting this notification is no longer useful, so all receivers should unregister themselves for any notifications involving the `NSConnection` object.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

`NSPortDidBecomeInvalidNotification` (NSPort notification)

**Declared In**

`NSConnection.h`

### **NSConnectionDidInitializeNotification**

Posted when an `NSConnection` object is initialized using `initWithReceivePort:sendPort:` (page 17) (the designated initializer for `NSConnection`). The notification object is the `NSConnection` object. This notification does not contain a `userInfo` dictionary.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- `initWithReceivePort:sendPort:` (page 17)

**Declared In**

`NSConnection.h`



# Document Revision History

---

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

Date	Notes
2008-02-08	Updated information for the <code>enableMultipleThreads</code> method.
2007-04-30	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

`addRequestMode:` [instance method 15](#)  
`addRunLoop:` [instance method 15](#)  
`allConnections` [class method 9](#)  
`authenticateComponents:withData:` [<NSObject> delegate method 28](#)  
`authenticationDataForComponents:` [<NSObject> delegate method 29](#)

## C

---

[Connection Exception Names 32](#)  
`connection:handleRequest:` [<NSObject> delegate method 30](#)  
`connection:shouldMakeNewConnection:` [<NSObject> delegate method 30](#)  
`connectionWithReceivePort:sendPort:` [class method 9](#)  
`connectionWithRegisteredName:host:` [class method 10](#)  
`connectionWithRegisteredName:host:usingNameServer:` [class method 11](#)  
`createConversationForConnection:` [<NSObject> delegate method 31](#)  
`currentConversation` [class method 11](#)

## D

---

`defaultConnection` [class method 11](#)  
`delegate` [instance method 16](#)

## E

---

`enableMultipleThreads` [instance method 16](#)

## I

---

`independentConversationQueueing` [instance method 16](#)  
`initWithReceivePort:sendPort:` [instance method 17](#)  
`invalidate` [instance method 18](#)  
`isValid` [instance method 18](#)

## L

---

`localObjects` [instance method 19](#)

## M

---

`makeNewConnection:sender:` [<NSObject> delegate method 31](#)  
`multipleThreadsEnabled` [instance method 19](#)

## N

---

[NSConnection run loop mode 32](#)  
[NSConnectionDidDieNotification notification 32](#)  
[NSConnectionDidInitializeNotification notification 33](#)  
`NSConnectionReplyMode` [constant 32](#)  
`NSFailedAuthenticationException` [constant 32](#)

## R

---

`receivePort` [instance method 20](#)  
`registerName:` [instance method 20](#)  
`registerName:withNameServer:` [instance method 21](#)  
`remoteObjects` [instance method 22](#)  
`removeRequestMode:` [instance method 22](#)  
`removeRunLoop:` [instance method 22](#)

replyTimeout **instance method** [23](#)  
requestModes **instance method** [23](#)  
requestTimeout **instance method** [24](#)  
rootObject **instance method** [24](#)  
rootProxy **instance method** [24](#)  
rootProxyForConnectionWithRegisteredName:host:  
    **class method** [12](#)  
rootProxyForConnectionWithRegisteredName:host:  
    usingNameServer: **class method** [13](#)  
runInNewThread **instance method** [25](#)

## S

---

sendPort **instance method** [25](#)  
serviceConnectionWithName:rootObject: **class method** [13](#)  
serviceConnectionWithName:rootObject:  
    usingNameServer: **class method** [14](#)  
setDelegate: **instance method** [26](#)  
setIndependentConversationQueueing: **instance method** [26](#)  
setReplyTimeout: **instance method** [27](#)  
setRequestTimeout: **instance method** [27](#)  
setRootObject: **instance method** [27](#)  
statistics **instance method** [28](#)