

---

# NSPort Class Reference

Networking, Internet, & Web: Sockets & TCP



2009-10-19



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, eMac, 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

---

## **NSPort Class Reference 5**

---

Overview	5
Adopted Protocols	6
Tasks	6
Creating Instances	6
Validation	6
Setting the Delegate	6
Creating Connections	7
Setting Information	7
Port Monitoring	7
Class Methods	7
allocWithZone:	7
port	8
Instance Methods	8
addConnection:toRunLoop:forMode:	8
delegate	9
invalidate	9
isValid	10
removeConnection:fromRunLoop:forMode:	10
removeFromRunLoop:forMode:	10
reservedSpaceLength	11
scheduleInRunLoop:forMode:	11
sendBeforeDate:components:from:reserved:	12
sendBeforeDate:msgid:components:from:reserved:	12
setDelegate:	13
Notifications	13
NSPortDidBecomeInvalidNotification	13

---

## **Document Revision History 15**

---

## **Index 17**

---



# NSPort Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSCoding NSCopying NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Declared in</b>	NSPort.h
<b>Companion guides</b>	Distributed Objects Programming Topics Threading Programming Guide
<b>Related sample code</b>	SimpleThreads TrivialThreads

## Overview

`NSPort` is an abstract class that represents a communication channel. Communication occurs between `NSPort` objects, which typically reside in different threads or tasks. The distributed objects system uses `NSPort` objects to send `NSPortMessage` objects back and forth. You should implement interapplication communication using distributed objects whenever possible and use `NSPort` objects only when necessary.

To receive incoming messages, `NSPort` objects must be added to an `NSRunLoop` object as input sources. `NSConnection` objects automatically add their receive port when initialized.

When an `NSPort` object receives a port message, it forwards the message to its delegate in a `handleMachMessage:` or `handlePortMessage:` message. The delegate should implement only one of these methods to process the incoming message in whatever form desired. `handleMachMessage:` provides a message as a raw Mach message beginning with a `msg_header_t` structure. `handlePortMessage:` provides a message as an `NSPortMessage` object, which is an object-oriented wrapper for a Mach message. If a delegate has not been set, the `NSPort` object handles the message itself.

When you are finished using a port object, you must explicitly invalidate the port object prior to sending it a `release` message. Similarly, if your application uses garbage collection, you must invalidate the port object before removing any strong references to it. If you do not invalidate the port, the resulting port object may linger and create a memory leak. To invalidate the port object, invoke its `invalidate` method.

Foundation defines three concrete subclasses of `NSPort`. `NSMachPort` and `NSMessagePort` allow local (on the same machine) communication only. `NSSocketPort` allows for both local and remote communication, but may be more expensive than the others for the local case. When creating an `NSPort` object, using `allocWithZone:` (page 7) or `port` (page 8), an `NSMachPort` object is created instead.

**Important:** `NSPort` conforms to the `NSCoding` protocol, but only supports coding by an `NSPortCoder`. `NSPort` and its subclasses do not support archiving.

## Adopted Protocols

### NSCoding

`encodeWithCoder:`  
`initWithCoder:`

### NSCopying

`copyWithZone:`

## Tasks

### Creating Instances

- + `allocWithZone:` (page 7)  
Returns an instance of the `NSMachPort` class.
- + `port` (page 8)  
Creates and returns a new `NSPort` object capable of both sending and receiving messages.

### Validation

- `invalidate` (page 9)  
Marks the receiver as invalid and posts an `NSPortDidBecomeInvalidNotification` (page 13) to the default notification center.
- `isValid` (page 10)  
Returns a Boolean value that indicates whether the receiver is valid.

### Setting the Delegate

- `setDelegate:` (page 13)  
Sets the receiver's delegate to a given object.
- `delegate` (page 9)  
Returns the receiver's delegate.

## Creating Connections

- [addConnection:toRunLoop:forMode:](#) (page 8)  
Adds the receiver to the list of ports monitored by a given run loop for the given input mode.
- [removeConnection:fromRunLoop:forMode:](#) (page 10)  
Removes the receiver from the list of ports monitored by *runLoop* in the given input mode, *mode*.

## Setting Information

- [sendBeforeDate:components:from:reserved:](#) (page 12)  
This method is provided for subclasses that have custom types of NSPort.
- [sendBeforeDate:msgid:components:from:reserved:](#) (page 12)  
This method is provided for subclasses that have custom types of NSPort.
- [reservedSpaceLength](#) (page 11)  
Returns the number of bytes of space reserved by the receiver for sending data.

## Port Monitoring

- [removeFromRunLoop:forMode:](#) (page 10)  
This method should be implemented by a subclass to stop monitoring of a port when removed from a give run loop in a given input mode.
- [scheduleInRunLoop:forMode:](#) (page 11)  
This method should be implemented by a subclass to set up monitoring of a port when added to a given run loop in a given input mode.

## Class Methods

### allocWithZone:

Returns an instance of the NSMachPort class.

```
+ (id)allocWithZone:(NSZone *)zone
```

#### Parameters

*zone*

The memory zone in which to allocate the new object.

#### Return Value

An instance of the NSMachPort class.

#### Discussion

For backward compatibility on Mach, `allocWithZone:` returns an instance of the NSMachPort class when sent to the NSPort class. Otherwise, it returns an instance of a concrete subclass that can be used for messaging between threads or processes on the local machine, or, in the case of NSSocketPort, between processes on separate machines.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSPort.h

**port**

Creates and returns a new `NSPort` object capable of both sending and receiving messages.

```
+ (NSPort *)port
```

**Return Value**

A new `NSPort` object capable of both sending and receiving messages.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

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

**Related Sample Code**

SimpleThreads

TrivialThreads

**Declared In**

NSPort.h

## Instance Methods

**addConnection:toRunLoop:forMode:**

Adds the receiver to the list of ports monitored by a given run loop for the given input mode.

```
- (void)addConnection:(NSConnection *)connection toRunLoop:(NSRunLoop *)runLoop
    forMode:(NSString *)mode
```

**Parameters**

*connection*

The connection object that invoked this method.

*runLoop*

The run loop to which to add the receiver.

*mode*

The run loop mode in which to add the receiver.

**Discussion**

You should not call this method directly. The method is provided for subclassers who wish to provide their own custom types of `NSPort`. The `NSConnection` object, *connection*, calls this method at the appropriate times.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

`addPort:forMode:` (NSRunLoop)

### Declared In

NSPort.h

## delegate

Returns the receiver's delegate.

- (id < NSPortDelegate >)delegate

### Return Value

The receiver's delegate.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [setDelegate:](#) (page 13)

### Declared In

NSPort.h

## invalidate

Marks the receiver as invalid and posts an [NSPortDidBecomeInvalidNotification](#) (page 13) to the default notification center.

- (void)invalidate

### Discussion

You must call this method before releasing a port object (or removing strong references to it if your application is garbage collected).

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [isValid](#) (page 10)

### Related Sample Code

SimpleThreads

### Declared In

NSPort.h

## isValid

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

- (BOOL)isValid

### Return Value

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

### Discussion

An `NSPort` object becomes invalid when its underlying communication resource, which is operating system dependent, is closed or damaged.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [invalidate](#) (page 9)

### Declared In

`NSPort.h`

## removeConnection:fromRunLoop:forMode:

Removes the receiver from the list of ports monitored by `runLoop` in the given input mode, `mode`.

```
- (void)removeConnection:(NSConnection *)connection fromRunLoop:(NSRunLoop *)runLoop
    forMode:(NSString *)mode
```

### Parameters

*connection*

The connection object that invoked this method.

*runLoop*

The run loop to which to add the receiver.

*mode*

The run loop mode in which to add the receiver.

### Discussion

You should not call this method directly. The method is provided for subclasses who wish to provide their own custom types of `NSPort`. The `NSConnection` object, *connection*, calls this method at the appropriate times.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`NSPort.h`

## removeFromRunLoop:forMode:

This method should be implemented by a subclass to stop monitoring of a port when removed from a give run loop in a given input mode.

- (void)removeFromRunLoop:(NSRunLoop \*)*runLoop* forMode:(NSString \*)*mode*

#### Parameters

*runLoop*

The run loop from which to remove the receiver.

*mode*

The run loop mode from which to remove the receiver

#### Discussion

This method should not be called directly.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [scheduleInRunLoop:forMode:](#) (page 11)

#### Declared In

NSPort.h

## reservedSpaceLength

Returns the number of bytes of space reserved by the receiver for sending data.

- (NSUInteger)reservedSpaceLength

#### Return Value

The number of bytes reserved by the receiver for sending data. The default length is 0.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

NSPort.h

## scheduleInRunLoop:forMode:

This method should be implemented by a subclass to set up monitoring of a port when added to a given run loop in a given input mode.

- (void)scheduleInRunLoop:(NSRunLoop \*)*runLoop* forMode:(NSString \*)*mode*

#### Parameters

*runLoop*

The run loop to which to add the receiver.

*mode*

The run loop mode to which to add the receiver

#### Discussion

This method should not be called directly.

#### Availability

Available in Mac OS X v10.0 and later.

**See Also**

- [removeFromRunLoop:forMode:](#) (page 10)

**Declared In**

NSPort.h

**sendBeforeDate:components:from:reserved:**

This method is provided for subclasses that have custom types of NSPort.

```
- (BOOL)sendBeforeDate:(NSDate *)limitDate components:(NSMutableArray *)components
    from:(NSPort *)receivePort reserved:(NSUInteger)headerSpaceReserved
```

**Parameters**

*limitDate*

The last instant that a message may be sent.

*components*

The message components.

*receivePort*

The receive port.

*headerSpaceReserved*

The number of bytes reserved for the header.

**Discussion**

NSConnection calls this method at the appropriate times. This method should not be called directly. This method could raise an NSInvalidSendPortException, NSInvalidReceivePortException, or an NSPortSendException, depending on the type of send port and the type of error.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSPort.h

**sendBeforeDate:msgid:components:from:reserved:**

This method is provided for subclasses that have custom types of NSPort.

```
- (BOOL)sendBeforeDate:(NSDate *)limitDate msgid:(NSUInteger)msgID
    components:(NSMutableArray *)components from:(NSPort *)receivePort
    reserved:(NSUInteger)headerSpaceReserved
```

**Parameters**

*limitDate*

The last instant that a message may be sent.

*msgID*

The message ID.

*components*

The message components.

*receivePort*

The receive port.

*headerSpaceReserved*

The number of bytes reserved for the header.

#### Discussion

`NSConnection` calls this method at the appropriate times. This method should not be called directly. This method could raise an `NSInvalidSendPortException`, `NSInvalidReceivePortException`, or an `NSPortSendException`, depending on the type of send port and the type of error.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

`NSPort.h`

## setDelegate:

Sets the receiver's delegate to a given object.

```
- (void)setDelegate:(id < NSPortDelegate >)anObject
```

#### Parameters

*anObject*

The delegate for the receiver.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [delegate](#) (page 9)

#### Declared In

`NSPort.h`

## Notifications

### NSPortDidBecomeInvalidNotification

Posted from the [invalidate](#) (page 9) method, which is invoked when the `NSPort` is deallocated or when it notices that its communication channel has been damaged. The notification object is the `NSPort` object that has become invalid. This notification does not contain a `userInfo` dictionary.

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

The `NSPort` object posting this notification is no longer useful, so all receivers should unregister themselves for any notifications involving the `NSPort`. A method receiving this notification should check to see which port became invalid before attempting to do anything. In particular, observers that receive all `NSPortDidBecomeInvalidNotification` messages should be aware that communication with the window server is handled through an `NSPort`. If this port becomes invalid, drawing operations will cause a fatal error.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSPort.h

# Document Revision History

---

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

Date	Notes
2009-10-19	Updated the companion document links.
2009-04-14	Updated for Mac OS X v10.6. Delegate methods move to NSPortDelegate Protocol Reference.
2007-07-19	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

---

`addConnection:toRunLoop:forMode:` [instance method 8](#)  
`allocWithZone:` [class method 7](#)

## D

---

`delegate` [instance method 9](#)

## I

---

`invalidate` [instance method 9](#)  
`isValid` [instance method 10](#)

## N

---

`NSPortDidBecomeInvalidNotification` [notification 13](#)

## P

---

`port` [class method 8](#)

## R

---

`removeConnection:fromRunLoop:forMode:` [instance method 10](#)  
`removeFromRunLoop:forMode:` [instance method 10](#)  
`reservedSpaceLength` [instance method 11](#)

## S

---

`scheduleInRunLoop:forMode:` [instance method 11](#)  
`sendBeforeDate:components:from:reserved:` [instance method 12](#)  
`sendBeforeDate:msgid:components:from:reserved:` [instance method 12](#)  
`setDelegate:` [instance method 13](#)