

---

# NSRunLoop Class Reference

Data Management: Event Handling



2009-07-14



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, Cocoa, Mac, Mac OS, and Quartz are trademarks of Apple Inc., registered in the United States and other countries.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

OpenGL is a registered trademark of Silicon Graphics, 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

---

## **NSRunLoop Class Reference 5**

---

Overview	5
Tasks	6
Accessing Run Loops and Modes	6
Managing Timers	6
Managing Ports	6
Configuring as Server Process	6
Running a Loop	6
Scheduling and Canceling Messages	7
Class Methods	7
currentRunLoop	7
mainRunLoop	8
Instance Methods	8
acceptInputForMode:beforeDate:	8
addPort:forMode:	9
addTimer:forMode:	9
cancelPerformSelector:target:argument:	10
cancelPerformSelectorsWithTarget:	10
currentMode	11
getCFRunLoop	11
limitDateForMode:	12
performSelector:target:argument:order:modes:	12
removePort:forMode:	13
run	13
runMode:beforeDate:	14
runUntilDate:	15
Constants	16
Run Loop Modes	16

---

## **Appendix A      Deprecated NSRunLoop Methods 17**

---

Deprecated in Mac OS X v10.5	17
configureAsServer	17

---

## **Document Revision History 19**

---

---

## **Index 21**

---



# NSRunLoop 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>	Threading Programming Guide
<b>Declared in</b>	NSRunLoop.h
<b>Related sample code</b>	CocoaEcho CocoaSOAP GLUT QTAudioExtractionPanel WhackedTV

## Overview

The `NSRunLoop` class declares the programmatic interface to objects that manage input sources. An `NSRunLoop` object processes input for sources such as mouse and keyboard events from the window system, `NSPort` objects, and `NSConnection` objects. An `NSRunLoop` object also processes `NSTimer` events.

In general, your application does not need to either create or explicitly manage `NSRunLoop` objects. Each `NSThread` object, including the application's main thread, has an `NSRunLoop` object automatically created for it as needed. If you need to access the current thread's run loop, you do so with the class method `currentRunLoop` (page 7).

Note that from the perspective of `NSRunLoop`, `NSTimer` objects are not "input"—they are a special type, and one of the things that means is that they do not cause the run loop to return when they fire.

 **Warning:** The `NSRunLoop` class is generally not considered to be thread-safe and its methods should only be called within the context of the current thread. You should never try to call the methods of an `NSRunLoop` object running in a different thread, as doing so might cause unexpected results.

## Tasks

### Accessing Run Loops and Modes

- + [currentRunLoop](#) (page 7)  
Returns the `NSRunLoop` object for the current thread.
- [currentMode](#) (page 11)  
Returns the receiver's current input mode.
- [limitDateForMode:](#) (page 12)  
Performs one pass through the run loop in the specified mode and returns the date at which the next timer is scheduled to fire.
- + [mainRunLoop](#) (page 8)  
Returns the run loop of the main thread.
- [getCFRunLoop](#) (page 11)  
Returns the receiver's underlying *CFRunLoop Reference* object.

### Managing Timers

- [addTimer:forMode:](#) (page 9)  
Registers a given timer with a given input mode.

### Managing Ports

- [addPort:forMode:](#) (page 9)  
Adds a port as an input source to the specified mode of the run loop.
- [removePort:forMode:](#) (page 13)  
Removes a port from the specified input mode of the run loop.

### Configuring as Server Process

- [configureAsServer](#) (page 17) **Deprecated in Mac OS X v10.5**  
Deprecated. Does nothing. (**Deprecated**. Deprecated since Mac OS X v10.5. There is no alternative method.)

### Running a Loop

- [run](#) (page 13)  
Puts the receiver into a permanent loop, during which time it processes data from all attached input sources.
- [runMode:beforeDate:](#) (page 14)  
Runs the loop once, blocking for input in the specified mode until a given date.

- [runUntilDate:](#) (page 15)  
Runs the loop until the specified date, during which time it processes data from all attached input sources.
- [acceptInputForMode:beforeDate:](#) (page 8)  
Runs the loop once or until the specified date, accepting input only for the specified mode.

## Scheduling and Canceling Messages

- [performSelector:target:argument:order:modes:](#) (page 12)  
Schedules the sending of a message on the current run loop.
- [cancelPerformSelector:target:argument:](#) (page 10)  
Cancels the sending of a previously scheduled message.
- [cancelPerformSelectorsWithTarget:](#) (page 10)  
Cancels all outstanding ordered performs scheduled with a given target.

## Class Methods

### currentRunLoop

Returns the `NSRunLoop` object for the current thread.

```
+ (NSRunLoop *)currentRunLoop
```

#### Return Value

The `NSRunLoop` object for the current thread.

#### Discussion

If a run loop does not yet exist for the thread, one is created and returned.

#### Availability

Available in Mac OS X v10.0 and later.

#### See Also

- [currentMode](#) (page 11)

#### Related Sample Code

CocoaEcho  
CocoaSOAP  
QTAudioExtractionPanel  
Quartz Composer QCTV  
WhackedTV

#### Declared In

`NSRunLoop.h`

## mainRunLoop

Returns the run loop of the main thread.

```
+ (NSRunLoop *)mainRunLoop
```

### Return Value

An object representing the main thread's run loop.

### Availability

Available in Mac OS X v10.5.

### Declared In

NSRunLoop.h

## Instance Methods

### acceptInputForMode:beforeDate:

Runs the loop once or until the specified date, accepting input only for the specified mode.

```
- (void)acceptInputForMode:(NSString *)mode beforeDate:(NSDate *)limitDate
```

### Parameters

*mode*

The mode in which to run. You may specify custom modes or use one of the modes listed in “[Run Loop Modes](#)” (page 16).

*limitDate*

The date up until which to run.

### Discussion

If no input sources or timers are attached to the run loop, this method exits immediately; otherwise, it runs the run loop once, returning as soon as one input source processes a message or the specified time elapses.

**Note:** A timer is not considered an input source and may fire multiple times while waiting for this method to return

Manually removing all known input sources and timers from the run loop is not a guarantee that the run loop will exit. Mac OS X can install and remove additional input sources as needed to process requests targeted at the receiver's thread. Those sources could therefore prevent the run loop from exiting.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [runMode:beforeDate:](#) (page 14)

### Declared In

NSRunLoop.h

## addPort:forMode:

Adds a port as an input source to the specified mode of the run loop.

```
- (void)addPort:(NSPort *)aPort forMode:(NSString *)mode
```

### Parameters

*aPort*

The port to add to the receiver.

*mode*

The mode in which to add *aPort*. You may specify a custom mode or use one of the modes listed in [“Run Loop Modes”](#) (page 16).

### Discussion

This method schedules the port with the receiver. You can add a port to multiple input modes. When the receiver is running in the specified mode, it dispatches messages destined for that port to the port’s designated handler routine.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

- [removePort:forMode:](#) (page 13)

### Declared In

NSRunLoop.h

## addTimer:forMode:

Registers a given timer with a given input mode.

```
- (void)addTimer:(NSTimer *)aTimer forMode:(NSString *)mode
```

### Parameters

*aTimer*

The timer to register with the receiver.

*mode*

The mode in which to add *aTimer*. You may specify a custom mode or use one of the modes listed in [“Run Loop Modes”](#) (page 16).

### Discussion

You can add a timer to multiple input modes. While running in the designated mode, the receiver causes the timer to fire on or after its scheduled fire date. Upon firing, the timer invokes its associated handler routine, which is a selector on a designated object.

The receiver retains *aTimer*. To remove a timer from all run loop modes on which it is installed, send an `invalidate` message to the timer.

### Availability

Available in Mac OS X v10.0 and later.

### Related Sample Code

[OpenGLCaptureToMovie](#)

[Quartz Composer Live DV](#)

Quartz Composer QCTV  
 Quartz Composer Texture  
 WhackedTV

**Declared In**  
 NSRunLoop.h

## cancelPerformSelector:target:argument:

Cancels the sending of a previously scheduled message.

```
- (void)cancelPerformSelector:(SEL)aSelector target:(id)target
    argument:(id)anArgument
```

### Parameters

*aSelector*

The previously-specified selector.

*target*

The previously-specified target.

*anArgument*

The previously-specified argument.

### Discussion

You can use this method to cancel a message previously scheduled using the [performSelector:target:argument:order:modes:](#) (page 12) method. The parameters identify the message you want to cancel and must match those originally specified when the selector was scheduled. This method removes the perform request from all modes of the run loop.

### Availability

Available in Mac OS X v10.0 and later.

**Declared In**  
 NSRunLoop.h

## cancelPerformSelectorsWithTarget:

Cancels all outstanding ordered performs scheduled with a given target.

```
- (void)cancelPerformSelectorsWithTarget:(id)target
```

### Parameters

*target*

The previously-specified target.

### Discussion

This method cancels the previously scheduled messages associated with the target, ignoring the selector and argument of the scheduled operation. This is in contrast to [cancelPerformSelector:target:argument:](#) (page 10), which requires you to match the selector and argument as well as the target. This method removes the perform requests for the object from all modes of the run loop.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

NSRunLoop.h

**currentMode**

Returns the receiver's current input mode.

```
- (NSString *)currentMode
```

**Return Value**

The receiver's current input mode. This method returns the current input mode *only* while the receiver is running; otherwise, it returns `nil`.

**Discussion**

The current mode is set by the methods that run the run loop, such as [acceptInputForMode:beforeDate:](#) (page 8) and [runMode:beforeDate:](#) (page 14).

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- + [currentRunLoop](#) (page 7)
- [limitDateForMode:](#) (page 12)
- [run](#) (page 13)
- [runUntilDate:](#) (page 15)

**Declared In**

NSRunLoop.h

**getCFRunLoop**

Returns the receiver's underlying *CFRunLoop Reference* object.

```
- (CFRunLoopRef)getCFRunLoop
```

**Return Value**

The receiver's underlying *CFRunLoop Reference* object.

**Discussion**

You can use the returned run loop to configure the current run loop using Core Foundation function calls. For example, you might use this function to set up a run loop observer.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

STUCAuthoringDeviceCocoaSample

**Declared In**

NSRunLoop.h

## limitDateForMode:

Performs one pass through the run loop in the specified mode and returns the date at which the next timer is scheduled to fire.

```
- (NSDate *)limitDateForMode:(NSString *)mode
```

### Parameters

*mode*

The run loop mode to search. You may specify custom modes or use one of the modes listed in [“Run Loop Modes”](#) (page 16).

### Return Value

The date at which the next timer is scheduled to fire, or `nil` if there are no input sources for this mode.

### Discussion

The run loop is entered with an immediate timeout, so the run loop does not block, waiting for input, if no input sources need processing.

### Availability

Available in Mac OS X v10.0 and later.

### Related Sample Code

GLUT

### Declared In

NSRunLoop.h

## performSelector:target:argument:order:modes:

Schedules the sending of a message on the current run loop.

```
- (void)performSelector:(SEL)aSelector target:(id)target argument:(id)anArgument
    order:(NSUInteger)order modes:(NSArray *)modes
```

### Parameters

*aSelector*

A selector that identifies the method to invoke. This method should not have a significant return value and should take a single argument of type `id`.

*target*

The object that defines the selector in *aSelector*.

*anArgument*

The argument to pass to the method when it is invoked. Pass `nil` if the method does not take an argument.

*order*

The priority for the message. If multiple messages are scheduled, the messages with a lower order value are sent before messages with a higher order value.

*modes*

An array of input modes for which the message may be sent. You may specify custom modes or use one of the modes listed in [“Run Loop Modes”](#) (page 16).

**Discussion**

This method sets up a timer to perform the *aSelector* message on the current thread's run loop at the start of the next run loop iteration. The timer is configured to run in the modes specified by the *modes* parameter. When the timer fires, the thread attempts to dequeue the message from the run loop and perform the selector. It succeeds if the run loop is running and in one of the specified modes; otherwise, the timer waits until the run loop is in one of those modes.

This method returns before the *aSelector* message is sent. The receiver retains the *target* and *anArgument* objects until the timer for the selector fires, and then releases them as part of its cleanup.

Use this method if you want multiple messages to be sent after the current event has been processed and you want to make sure these messages are sent in a certain order.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [cancelPerformSelector:target:argument:](#) (page 10)

**Related Sample Code**

SimpleStickies

**Declared In**

NSRunLoop.h

**removePort:forMode:**

Removes a port from the specified input mode of the run loop.

```
(void)removePort:(NSPort *)aPort forMode:(NSString *)mode
```

**Parameters**

*aPort*

The port to remove from the receiver.

*mode*

The mode from which to remove *aPort*. You may specify a custom mode or use one of the modes listed in ["Run Loop Modes"](#) (page 16).

**Discussion**

If you added the port to multiple input modes, you must remove it from each mode separately.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [addPort:forMode:](#) (page 9)

**Declared In**

NSRunLoop.h

**run**

Puts the receiver into a permanent loop, during which time it processes data from all attached input sources.

- (void)run

### Discussion

If no input sources or timers are attached to the run loop, this method exits immediately; otherwise, it runs the receiver in the `NSDefaultRunLoopMode` by repeatedly invoking `runMode:beforeDate:` (page 14). In other words, this method effectively begins an infinite loop that processes data from the run loop's input sources and timers.

Manually removing all known input sources and timers from the run loop is not a guarantee that the run loop will exit. Mac OS X can install and remove additional input sources as needed to process requests targeted at the receiver's thread. Those sources could therefore prevent the run loop from exiting.

If you want the run loop to terminate, you shouldn't use this method. Instead, use one of the other run methods and also check other arbitrary conditions of your own, in a loop. A simple example would be:

```
BOOL shouldKeepRunning = YES;           // global
NSRunLoop *theRL = [NSRunLoop currentRunLoop];
while (shouldKeepRunning && [theRL runMode:NSDefaultRunLoopMode beforeDate:[NSDate
distantFuture]]);
```

where `shouldKeepRunning` is set to `NO` somewhere else in the program.

### Availability

Available in Mac OS X v10.0 and later.

### See Also

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

### Related Sample Code

Authenticator

CocoaEcho

CocoaHTTPServer

SimpleThreads

TrivialThreads

### Declared In

NSRunLoop.h

## runMode:beforeDate:

Runs the loop once, blocking for input in the specified mode until a given date.

```
- (BOOL)runMode:(NSString *)mode beforeDate:(NSDate *)limitDate
```

### Parameters

*mode*

The mode in which to run. You may specify custom modes or use one of the modes listed in [“Run Loop Modes”](#) (page 16).

*limitDate*

The date until which to block.

### Return Value

YES if the run loop ran and processed an input source or if the specified timeout value was reached; otherwise, NO if the run loop could not be started.

**Discussion**

If no input sources or timers are attached to the run loop, this method exits immediately and returns `NO`; otherwise, it returns after either the first input source is processed or `limitDate` is reached. Manually removing all known input sources and timers from the run loop does not guarantee that the run loop will exit immediately. Mac OS X may install and remove additional input sources as needed to process requests targeted at the receiver's thread. Those sources could therefore prevent the run loop from exiting.

**Note:** A timer is not considered an input source and may fire multiple times while waiting for this method to return

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [run](#) (page 13)
- [runUntilDate:](#) (page 15)

**Related Sample Code**

CocoaSOAP

**Declared In**

NSRunLoop.h

**runUntilDate:**

Runs the loop until the specified date, during which time it processes data from all attached input sources.

```
- (void)runUntilDate:(NSDate *)limitDate
```

**Parameters**

*limitDate*

The date up until which to run.

**Discussion**

If no input sources or timers are attached to the run loop, this method exits immediately; otherwise, it runs the receiver in the `NSDefaultRunLoopMode` by repeatedly invoking [runMode:beforeDate:](#) (page 14) until the specified expiration date.

Manually removing all known input sources and timers from the run loop is not a guarantee that the run loop will exit. Mac OS X can install and remove additional input sources as needed to process requests targeted at the receiver's thread. Those sources could therefore prevent the run loop from exiting.

**Availability**

Available in Mac OS X v10.0 and later.

**See Also**

- [run](#) (page 13)

**Related Sample Code**

EnhancedAudioBurn  
QTAudioContextInsert  
QTAudioExtractionPanel

**Declared In**  
NSRunLoop.h

## Constants

### Run Loop Modes

NSRunLoop defines the following run loop mode.

```
extern NSString* const NSDefaultRunLoopMode;
extern NSString* const NSRunLoopCommonModes;
```

#### Constants

NSDefaultRunLoopMode

The mode to deal with input sources other than `NSConnection` objects.

This is the most commonly used run-loop mode.

Available in Mac OS X v10.0 and later.

Declared in NSRunLoop.h.

NSRunLoopCommonModes

Objects added to a run loop using this value as the mode are monitored by all run loop modes that have been declared as a member of the set of “common” modes; see the description of `CFRunLoopAddCommonMode` for details.

Available in Mac OS X v10.5 and later.

Declared in NSRunLoop.h.

#### Declared In

Foundation/NSRunLoop.h

Additional run loop modes are defined by `NSConnection` and `NSApplication`.

<code>NSConnectionReplyMode</code>	The mode to indicate an <code>NSConnection</code> object waiting for replies.
<code>NSModalPanelRunLoopMode</code>	A run loop should be set to this mode when waiting for input from a modal panel, such as <code>NSSavePanel</code> or <code>NSOpenPanel</code> .
<code>NSEventTrackingRunLoopMode</code>	A run loop should be set to this mode when tracking events modally, such as a mouse-dragging loop.

# Deprecated NSRunLoop Methods

---

A method identified as deprecated has been superseded and may become unsupported in the future.

## Deprecated in Mac OS X v10.5

### **configureAsServer**

Deprecated. Does nothing. (Deprecated in Mac OS X v10.5. Deprecated since Mac OS X v10.5. There is no alternative method.)

- (void)configureAsServer

#### **Discussion**

On Mac OS X, this method does nothing.

#### **Availability**

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.5.

#### **Declared In**

NSRunLoop.h



# Document Revision History

---

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

Date	Notes
2009-07-14	Updated the declarations for the run loop mode constants.
2008-02-08	Updated the description of the <code>runMode:beforeDate:</code> method.
2007-06-12	Updated for Mac OS X version 10.5.
2006-05-23	Enhanced the description of the <code>run</code> method.
	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## A

---

`acceptInputForMode:beforeDate:` [instance method 8](#)  
`addPort:forMode:` [instance method 9](#)  
`addTimer:forMode:` [instance method 9](#)

## C

---

`cancelPerformSelector:target:argument:` [instance method 10](#)  
`cancelPerformSelectorsWithTarget:` [instance method 10](#)  
`configureAsServer` [instance method 17](#)  
`currentMode` [instance method 11](#)  
`currentRunLoop` [class method 7](#)

## G

---

`getCFRunLoop` [instance method 11](#)

## L

---

`limitDateForMode:` [instance method 12](#)

## M

---

`mainRunLoop` [class method 8](#)

## N

---

`NSDefaultRunLoopMode` [constant 16](#)  
`NSRunLoopCommonModes` [constant 16](#)

## P

---

`performSelector:target:argument:order:modes:` [instance method 12](#)

## R

---

`removePort:forMode:` [instance method 13](#)  
`run` [instance method 13](#)  
[Run Loop Modes 16](#)  
`runMode:beforeDate:` [instance method 14](#)  
`runUntilDate:` [instance method 15](#)