
NSTimer Class Reference

Data Management: Event Handling



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

NSTimer Class Reference 5

Overview	5
Repeating Versus Non-Repeating Timers	6
Scheduling Timers in Run Loops	6
Tasks	7
Creating a Timer	7
Firing a Timer	7
Stopping a Timer	7
Information About a Timer	7
Class Methods	8
scheduledTimerWithTimeInterval:invocation:repeats:	8
scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:	8
timerWithTimeInterval:invocation:repeats:	9
timerWithTimeInterval:target:selector:userInfo:repeats:	10
Instance Methods	11
fire	11
fireDate	11
initWithFireDate:interval:target:selector:userInfo:repeats:	12
invalidate	12
isValid	13
setFireDate:	13
timeInterval	14
userInfo	14

Document Revision History 17

Index 19

NSTimer Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTimer.h
Companion guides	Timer Programming Topics for Cocoa Threading Programming Guide
Related sample code	FunkyOverlayWindow ImageClient NSOperationSample Threadslmporater ThreadslmporaterMovie

Overview

You use the `NSTimer` class to create timer objects or, more simply, timers. A timer waits until a certain time interval has elapsed and then fires, sending a specified message to a target object. For example, you could create an `NSTimer` object that sends a message to a window, telling it to update itself after a certain time interval.

Timers work in conjunction with run loops. To use a timer effectively, you should be aware of how run loops operate—see `NSRunLoop` and *Threading Programming Guide*. Note in particular that run loops retain their timers, so you can release a timer after you have added it to a run loop.

A timer is not a real-time mechanism; it fires only when one of the run loop modes to which the timer has been added is running and able to check if the timer's firing time has passed. Because of the various input sources a typical run loop manages, the effective resolution of the time interval for a timer is limited to on the order of 50-100 milliseconds. If a timer's firing time occurs during a long callout or while the run loop is in a mode that is not monitoring the timer, the timer does not fire until the next time the run loop checks the timer. Therefore, the actual time at which the timer fires potentially can be a significant period of time after the scheduled firing time.

`NSTimer` objects are "toll-free bridged" with their Core Foundation counterparts, the `CFRunLoopTimerRef` type. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSTimer *` parameter, you can pass a `CFRunLoopTimerRef`, and in a function where you see a `CFRunLoopTimerRef` parameter, you can pass

an `NSTimer` instance (you cast one type to the other to suppress compiler warnings). See [Interchangeable Data Types](#) for more information on toll-free bridging. For more information about Core Foundation timers, see [CFRunLoopTimer Reference](#).

Repeating Versus Non-Repeating Timers

You specify whether a timer is repeating or non-repeating at creation time. A non-repeating timer fires once and then invalidates itself automatically, thereby preventing the timer from firing again. By contrast, a repeating timer fires and then reschedules itself on the same run loop.

A repeating timer always schedules itself based on the scheduled firing time, as opposed to the actual firing time. For example, if a timer is scheduled to fire at a particular time and every 5 seconds after that, the scheduled firing time will always fall on the original 5 second time intervals, even if the actual firing time gets delayed. If the firing time is delayed so far that it passes one or more of the scheduled firing times, the timer is fired only once for that time period; the timer is then rescheduled, after firing, for the next scheduled firing time in the future.

Scheduling Timers in Run Loops

A timer object can be registered in only one run loop at a time, although it can be added to multiple run loop modes within that run loop. There are three ways to create a timer:

- Use the [`scheduledTimerWithTimeInterval:invocation:repeats:`](#) (page 8) or [`scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:`](#) (page 8) class method to create the timer and schedule it on the current run loop in the default mode.
- Use the [`timerWithTimeInterval:invocation:repeats:`](#) (page 9) or [`timerWithTimeInterval:target:selector:userInfo:repeats:`](#) (page 10) class method to create the timer object without scheduling it on a run loop. (After creating it, you must add the timer to a run loop manually by calling the `addTimer:forMode:` method of the corresponding `NSRunLoop` object.)
- Allocate the timer and initialize it using the [`initWithFireDate:interval:target:selector:userInfo:repeats:`](#) (page 12) method. (After creating it, you must add the timer to a run loop manually by calling the `addTimer:forMode:` method of the corresponding `NSRunLoop` object.)

Once scheduled on a run loop, the timer fires at the specified interval until it is invalidated. A non-repeating timer invalidates itself immediately after it fires. However, for a repeating timer, you must invalidate the timer object yourself by calling its [`invalidate`](#) (page 12) method. Calling this method requests the removal of the timer from the current run loop; as a result, you should always call the [`invalidate`](#) (page 12) method from the same thread on which the timer was installed. Invalidating the timer immediately disables it so that it no longer affects the run loop. The run loop then removes and releases the timer, either just before the [`invalidate`](#) (page 12) method returns or at some later point. Once invalidated, timer objects cannot be reused.

Tasks

Creating a Timer

- + [scheduledTimerWithTimeInterval:invocation:repeats:](#) (page 8)
Creates and returns a new `NSTimer` object and schedules it on the current run loop in the default mode.
- + [scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:](#) (page 8)
Creates and returns a new `NSTimer` object and schedules it on the current run loop in the default mode.
- + [timerWithTimeInterval:invocation:repeats:](#) (page 9)
Creates and returns a new `NSTimer` object initialized with the specified invocation object.
- + [timerWithTimeInterval:target:selector:userInfo:repeats:](#) (page 10)
Creates and returns a new `NSTimer` object initialized with the specified object and selector.
- [initWithFireDate:interval:target:selector:userInfo:repeats:](#) (page 12)
Initializes a new `NSTimer` object using the specified object and selector.

Firing a Timer

- [fire](#) (page 11)
Causes the receiver's message to be sent to its target.

Stopping a Timer

- [invalidate](#) (page 12)
Stops the receiver from ever firing again and requests its removal from its run loop.

Information About a Timer

- [isValid](#) (page 13)
Returns a Boolean value that indicates whether the receiver is currently valid.
- [fireDate](#) (page 11)
Returns the date at which the receiver will fire.
- [setFireDate:](#) (page 13)
Resets the firing time of the receiver to the specified date.
- [timeInterval](#) (page 14)
Returns the receiver's time interval.
- [userInfo](#) (page 14)
Returns the receiver's `userInfo` object.

Class Methods

scheduledTimerWithTimeInterval:invocation:repeats:

Creates and returns a new `NSTimer` object and schedules it on the current run loop in the default mode.

```
+ (NSTimer *)scheduledTimerWithTimeInterval:(NSTimeInterval)seconds
    invocation:(NSInvocation *)invocation repeats:(BOOL)repeats
```

Parameters

seconds

The number of seconds between firings of the timer. If *seconds* is less than or equal to 0.0, this method chooses the nonnegative value of 0.1 milliseconds instead.

invocation

The invocation to use when the timer fires. The timer instructs the invocation object to retain its arguments.

repeats

If YES, the timer will repeatedly reschedule itself until invalidated. If NO, the timer will be invalidated after it fires.

Return Value

A new `NSTimer` object, configured according to the specified parameters.

Discussion

After *seconds* seconds have elapsed, the timer fires, invoking *invocation*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTimer.h`

scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:

Creates and returns a new `NSTimer` object and schedules it on the current run loop in the default mode.

```
+ (NSTimer *)scheduledTimerWithTimeInterval:(NSTimeInterval)seconds
    target:(id)target selector:(SEL)aSelector userInfo:(id)userInfo
    repeats:(BOOL)repeats
```

Parameters

seconds

The number of seconds between firings of the timer. If *seconds* is less than or equal to 0.0, this method chooses the nonnegative value of 0.1 milliseconds instead.

target

The object to which to send the message specified by *aSelector* when the timer fires. The target object is retained by the timer and released when the timer is invalidated.

aSelector

The message to send to *target* when the timer fires. The selector must have the following signature:

```
- (void)timerFireMethod:(NSTimer*)theTimer
```

The timer passes itself as the argument to this method.

userInfo

The user info for the timer. The object you specify is retained by the timer and released when the timer is invalidated. This parameter may be `nil`.

repeats

If YES, the timer will repeatedly reschedule itself until invalidated. If NO, the timer will be invalidated after it fires.

Return Value

A new `NSTimer` object, configured according to the specified parameters.

Discussion

After *seconds* seconds have elapsed, the timer fires, sending the message *aSelector* to *target*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Draw Pixels

FunkyOverlayWindow

GLUT

ImageClient

TextureRange

Declared In

`NSTimer.h`

timerWithTimeInterval:invocation:repeats:

Creates and returns a new `NSTimer` object initialized with the specified invocation object.

```
+ (NSTimer *)timerWithTimeInterval:(NSTimeInterval)seconds invocation:(NSInvocation *)invocation repeats:(BOOL)repeats
```

Parameters*seconds*

The number of seconds between firings of the timer. If *seconds* is less than or equal to 0.0, this method chooses the nonnegative value of 0.1 milliseconds instead.

invocation

The invocation to use when the timer fires. The timer instructs the invocation object to retain its arguments.

repeats

If YES, the timer will repeatedly reschedule itself until invalidated. If NO, the timer will be invalidated after it fires.

Return Value

A new `NSTimer` object, configured according to the specified parameters.

Discussion

You must add the new timer to a run loop, using `addTimer:forMode:`. Then, after *seconds* have elapsed, the timer fires, invoking *invocation*. (If the timer is configured to repeat, there is no need to subsequently re-add the timer to the run loop.)

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSTimer.h

timerWithTimeInterval:target:selector:userInfo:repeats:

Creates and returns a new NSTimer object initialized with the specified object and selector.

```
+ (NSTimer *)timerWithTimeInterval:(NSTimeInterval)seconds target:(id)target
  selector:(SEL)aSelector userInfo:(id)userInfo repeats:(BOOL)repeats
```

Parameters

seconds

The number of seconds between firings of the timer. If *seconds* is less than or equal to 0.0, this method chooses the nonnegative value of 0.1 milliseconds instead.

target

The object to which to send the message specified by *aSelector* when the timer fires. The target object is retained by the timer and released when the timer is invalidated.

aSelector

The message to send to *target* when the timer fires. The selector must have the following signature:

```
- (void)timerFireMethod:(NSTimer*)theTimer
```

The timer passes itself as the argument to this method.

userInfo

Custom user info for the timer. The object you specify is retained by the timer and released when the timer is invalidated. This parameter may be `nil`.

repeats

If YES, the timer will repeatedly reschedule itself until invalidated. If NO, the timer will be invalidated after it fires.

Return Value

A new NSTimer object, configured according to the specified parameters.

Discussion

You must add the new timer to a run loop, using `addTimer:forMode:`. Then, after *seconds* seconds have elapsed, the timer fires, sending the message *aSelector* to *target*. (If the timer is configured to repeat, there is no need to subsequently re-add the timer to the run loop.)

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Cocoa OpenGL

GLChildWindowDemo

OpenGL Screensaver

Quartz Composer QCTV
SillyFrequencyLevels

Declared In
NSTimer.h

Instance Methods

fire

Causes the receiver's message to be sent to its target.

- (void)fire

Discussion

You can use this method to fire a repeating timer without interrupting its regular firing schedule. If the timer is non-repeating, it is automatically invalidated after firing, even if its scheduled fire date has not arrived.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [invalidate](#) (page 12)

Declared In

NSTimer.h

fireDate

Returns the date at which the receiver will fire.

- (NSDate *)fireDate

Return Value

The date at which the receiver will fire. If the timer is no longer valid, this method returns the last date at which the timer fired.

Discussion

Use the [isValid](#) (page 13) method to verify that the timer is valid.

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Declared In

NSTimer.h

initWithFireDate:interval:target:selector:userInfo:repeats:

Initializes a new `NSTimer` object using the specified object and selector.

```
- (id)initWithFireDate:(NSDate *)date interval:(NSTimeInterval)seconds
  target:(id)target selector:(SEL)aSelector userInfo:(id)userInfo
  repeats:(BOOL)repeats
```

Parameters

date

The time at which the timer should first fire.

seconds

For a repeating timer, this parameter contains the number of seconds between firings of the timer. If *seconds* is less than or equal to 0.0, this method chooses the nonnegative value of 0.1 milliseconds instead.

target

The object to which to send the message specified by *aSelector* when the timer fires. The target object is retained by the timer and released when the timer is invalidated.

aSelector

The message to send to *target* when the timer fires. The selector must have the following signature:

```
- (void)timerFireMethod:(NSTimer*)theTimer
```

The timer passes itself as the argument to this method.

userInfo

Custom user info for the timer. The object you specify is retained by the timer and released when the timer is invalidated. This parameter may be `nil`.

repeats

If YES, the timer will repeatedly reschedule itself until invalidated. If NO, the timer will be invalidated after it fires.

Return Value

The receiver, initialized such that, when added to a run loop, it will fire at *date* and then, if *repeats* is YES, every *seconds* after that.

Discussion

You must add the new timer to a run loop, using `addTimer:forMode:`. Upon firing, the timer sends the message *aSelector* to *target*. (If the timer is configured to repeat, there is no need to subsequently re-add the timer to the run loop.)

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

WhackedTV

Declared In

`NSTimer.h`

invalidate

Stops the receiver from ever firing again and requests its removal from its run loop.

- (void)invalidate

Discussion

This method is the only way to remove a timer from an `NSRunLoop` object. The `NSRunLoop` object removes and releases the timer, either just before the `invalidate` (page 12) method returns or at some later point.

If it was configured with target and user info objects, the receiver releases its references to those objects as well.

Special Considerations

You must send this message from the thread on which the timer was installed. If you send this message from another thread, the input source associated with the timer may not be removed from its run loop, which could prevent the thread from exiting properly.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [fire](#) (page 11)

Related Sample Code

CocoaSpeechSynthesisExample

Draw Pixels

GLUT

TextureRange

WhackedTV

Declared In

`NSTimer.h`

isValid

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

- (BOOL)isValid

Return Value

YES if the receiver is still capable of firing or NO if the timer has been invalidated and is no longer capable of firing.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSTimer.h`

setFireDate:

Resets the firing time of the receiver to the specified date.

- (void)setFireDate:(NSDate *)date

Parameters*date*

The new date at which to fire the receiver. If the new date is in the past, this method sets the fire time to the current time.

Discussion

You typically use this method to adjust the firing time of a repeating timer. Although resetting a timer's next firing time is a relatively expensive operation, it may be more efficient in some situations. For example, you could use it in situations where you want to repeat an action multiple times in the future, but at irregular time intervals. Adjusting the firing time of a single timer would likely incur less expense than creating multiple timer objects, scheduling each one on a run loop, and then destroying them.

You should not call this method on a timer that has been invalidated, which includes non-repeating timers that have already fired. You could potentially call this method on a non-repeating timer that had not yet fired, although you should always do so from the thread to which the timer is attached to avoid potential race conditions.

Availability

Available in Mac OS X v10.2 and later.

See Also

- [fireDate](#) (page 11)

Related Sample Code

GLUT

Declared In

NSTimer.h

timeInterval

Returns the receiver's time interval.

- (NSTimeInterval)timeInterval

Return Value

The receiver's time interval. If the receiver is a non-repeating timer, returns 0 (even if a time interval was set).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CubePuzzle

Declared In

NSTimer.h

userInfo

Returns the receiver's *userInfo* object.

- (id)userInfo

Return Value

The receiver's *userInfo* object.

Discussion

Do not invoke this method after the timer is invalidated. Use [isValid](#) (page 13) to test whether the timer is valid.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:](#) (page 8)

+ [timerWithTimeInterval:target:selector:userInfo:repeats:](#) (page 10)

- [invalidate](#) (page 12)

Declared In

NSTimer.h

Document Revision History

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

Date	Notes
2009-10-19	Updated the descriptions of timer behaviors during some edge cases.
2009-07-14	Corrected a minor typographical error.
2008-11-19	Augmented parameter descriptions.
2006-09-20	Clarified "repeat interval" for non-repeating timers.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

F

fire **instance method** [11](#)
fireDate **instance method** [11](#)

I

initWithFireDate:interval:target:selector:
 userInfo:repeats: **instance method** [12](#)
invalidate **instance method** [12](#)
isValid **instance method** [13](#)

S

scheduledTimerWithTimeInterval:invocation:repeats:
 class method [8](#)
scheduledTimerWithTimeInterval:target:selector:
 userInfo:repeats: **class method** [8](#)
setFireDate: **instance method** [13](#)

T

timeInterval **instance method** [14](#)
timerWithTimeInterval:invocation:repeats:
 class method [9](#)
timerWithTimeInterval:target:selector:userInfo:
 repeats: **class method** [10](#)

U

userInfo **instance method** [14](#)