

---

# CFRunLoopObserver Reference

Core Foundation



2006-02-07



Apple Inc.  
© 2003, 2006 Apple Computer, 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 and the Apple logo are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR**

**CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **CFRunLoopObserver Reference 5**

---

Overview	5
Functions	5
CFRunLoopObserverCreate	5
CFRunLoopObserverDoesRepeat	6
CFRunLoopObserverGetActivities	7
CFRunLoopObserverGetContext	7
CFRunLoopObserverGetOrder	8
CFRunLoopObserverGetTypeID	8
CFRunLoopObserverInvalidate	8
CFRunLoopObserverIsValid	9
Callbacks	9
CFRunLoopObserverCallBack	9
Data Types	10
CFRunLoopObserverContext	10
CFRunLoopObserverRef	11
Constants	11
Run Loop Activities	11

---

## **Document Revision History 13**

---

## **Index 15**

---



# CFRunLoopObserver Reference

---

<b>Derived From:</b>	CType
<b>Framework:</b>	CoreFoundation/CoreFoundation.h
<b>Companion guide</b>	Run Loops
<b>Declared in</b>	CFRunLoop.h

## Overview

A `CFRunLoopObserver` provides a general means to receive callbacks at different points within a running run loop. In contrast to sources, which fire when an asynchronous event occurs, and timers, which fire when a particular time passes, observers fire at special locations within the execution of the run loop, such as before sources are processed or before the run loop goes to sleep, waiting for an event to occur. Observers can be either one-time events or repeated every time through the run loop's loop.

Each run loop observer 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.

## Functions

### **CFRunLoopObserverCreate**

Creates a `CFRunLoopObserver` object.

```
CFRunLoopObserverRef CFRunLoopObserverCreate (
    CFAllocatorRef allocator,
    CFOptionFlags activities,
    Boolean repeats,
    CFIndex order,
    CFRunLoopObserverCallback callout,
    CFRunLoopObserverContext *context
);
```

#### **Parameters**

*allocator*

The allocator to use to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

*activities*

Set of flags identifying the activity stages of the run loop during which the observer should be called. See [Run Loop Activities](#) (page 11) for the list of stages. To have the observer called at multiple stages in the run loop, combine the [Run Loop Activities](#) (page 11) values using the bitwise-OR operator.

*repeats*

A flag identifying whether the observer should be called only once or every time through the run loop. If *repeats* is `false`, the observer is invalidated after it is called once, even if the observer was scheduled to be called at multiple stages within the run loop.

*order*

A priority index indicating the order in which run loop observers are processed. When multiple run loop observers are scheduled in the same activity stage in a given run loop mode, the observers are processed in increasing order of this parameter. Pass 0 unless there is a reason to do otherwise.

*callback*

The callback function invoked when the observer runs.

*context*

A structure holding contextual information for the run loop observer. The function copies the information out of the structure, so the memory pointed to by *context* does not need to persist beyond the function call. Can be `NULL` if the observer does not need the context's `info` pointer to keep track of state.

**Return Value**

The new `CFRunLoopObserver` object. Ownership follows the Create Rule.

**Discussion**

The run loop observer is not automatically added to a run loop. To add the observer to a run loop, use `CFRunLoopAddObserver`. An observer can be registered to only one run loop, although it can be added to multiple run loop modes within that run loop.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`CFRunLoop.h`

**CFRunLoopObserverDoesRepeat**

Returns a Boolean value that indicates whether a `CFRunLoopObserver` repeats.

```
Boolean CFRunLoopObserverDoesRepeat (
    CFRunLoopObserverRef observer
);
```

**Parameters**

*observer*

The run loop observer to examine.

**Return Value**

`true` if *observer* is processed during every pass through the run loop; `false` if *observer* is processed once and then is invalidated.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

**CFRunLoopObserverGetActivities**

Returns the run loop stages during which an observer runs.

```

CFOptionFlags CFRunLoopObserverGetActivities (
    CFRunLoopObserverRef observer
);

```

**Parameters***observer*

The run loop observer to examine.

**Return Value**A bitwise-OR combination of all the run loop stages in which *observer* is called. See [Run Loop Activities](#) (page 11) for the list of stages.**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

**CFRunLoopObserverGetContext**

Returns the context information for a CFRunLoopObserver object.

```

void CFRunLoopObserverGetContext (
    CFRunLoopObserverRef observer,
    CFRunLoopObserverContext *context
);

```

**Parameters***observer*

The run loop observer to examine.

*context*Upon return, contains the context information for *observer*. This is the same information passed to [CFRunLoopObserverCreate](#) (page 5) when creating *observer*.**Discussion**The context version number for run loop observers is currently 0. Before calling this function, you need to initialize the *version* member of *context* to 0.**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

## CFRunLoopObserverGetOrder

Returns the ordering parameter for a CFRunLoopObserver object.

```
CFIndex CFRunLoopObserverGetOrder (
    CFRunLoopObserverRef observer
);
```

### Parameters

*observer*

The run loop observer to examine.

### Return Value

The ordering parameter for *observer*. When multiple observers are scheduled in the same run loop mode and stage, this value determines the order (from small to large) in which the observers are called.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

CFRunLoop.h

## CFRunLoopObserverGetTypeID

Returns the type identifier for the CFRunLoopObserver opaque type.

```
CFTypeID CFRunLoopObserverGetTypeID (
    void
);
```

### Return Value

The type identifier for the CFRunLoopObserver opaque type.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

CFRunLoop.h

## CFRunLoopObserverInvalidate

Invalidates a CFRunLoopObserver object, stopping it from ever firing again.

```
void CFRunLoopObserverInvalidate (
    CFRunLoopObserverRef observer
);
```

### Parameters

*observer*

The run loop observer to invalidate.

### Discussion

Once invalidated, *observer* will never fire and call its callback function again. This function automatically removes *observer* from all run loop modes in which it had been added. The memory is not deallocated unless the run loop held the only reference to *observer*.



**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

**CFRunLoopObserverIsValid**

Returns a Boolean value that indicates whether a CFRunLoopObserver object is valid and able to fire.

```
Boolean CFRunLoopObserverIsValid (
    CFRunLoopObserverRef observer
);
```

**Parameters**

*observer*

The run loop observer to examine.

**Return Value**

true if *observer* is valid, otherwise false.

**Discussion**

A nonrepeating observer is automatically invalidated after it is called once.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

## Callbacks

**CFRunLoopObserverCallback**

Callback invoked when a CFRunLoopObserver object is fired.

```
typedef void (*CFRunLoopObserverCallback) (
    CFRunLoopObserverRef observer,
    CFRunLoopActivity activity,
    void *info
);
```

If you name your function `MyCallback`, you would declare it like this:

```
void MyCallback (
    CFRunLoopObserverRef observer,
    CFRunLoopActivity activity,
    void *info
);
```

**Parameters***observer*

The run loop observer that is firing.

*activity*

The current activity stage of the run loop.

*info*The *info* member of the [CFRunLoopObserverContext](#) (page 10) structure that was used when creating the run loop observer.**Discussion**You specify this callback when you create the run loop observer with [CFRunLoopObserverCreate](#) (page 5).**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

## Data Types

**CFRunLoopObserverContext**

A structure that contains program-defined data and callbacks with which you can configure a CFRunLoopObserver object's behavior.

```

struct CFRunLoopObserverContext {
    CFIndex version;
    void *info;
    CFAllocatorRetainCallback retain;
    CFAllocatorReleaseCallback release;
    CFAllocatorCopyDescriptionCallback copyDescription;
};
typedef struct CFRunLoopObserverContext CFRunLoopObserverContext;

```

**Fields***version*

Version number of the structure. Must be 0.

*info*

An arbitrary pointer to program-defined data, which can be associated with the run loop observer at creation time. This pointer is passed to all the callbacks defined in the context.

*retain*A retain callback for your program-defined *info* pointer. Can be NULL.*release*A release callback for your program-defined *info* pointer. Can be NULL.*copyDescription*A copy description callback for your program-defined *info* pointer. Can be NULL.**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

**CFRunLoopObserverRef**

A reference to a run loop observer object.

```
typedef struct __CFRunLoopObserver *CFRunLoopObserverRef;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFRunLoop.h

## Constants

### Run Loop Activities

Run loop activity stages in which run loop observers can be scheduled.

```
enum CFRunLoopActivity {
    kCFRunLoopEntry = (1 << 0),
    kCFRunLoopBeforeTimers = (1 << 1),
    kCFRunLoopBeforeSources = (1 << 2),
    kCFRunLoopBeforeWaiting = (1 << 5),
    kCFRunLoopAfterWaiting = (1 << 6),
    kCFRunLoopExit = (1 << 7),
    kCFRunLoopAllActivities = 0xFFFFFFFFU
};
typedef enum CFRunLoopActivity CFRunLoopActivity;
```

**Constants**

kCFRunLoopEntry

The entrance of the run loop, before entering the event processing loop. This activity occurs once for each call to `CFRunLoopRun` and `CFRunLoopRunInMode`.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

kCFRunLoopBeforeTimers

Inside the event processing loop before any timers are processed.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

kCFRunLoopBeforeSources

Inside the event processing loop before any sources are processed.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

`kCFRunLoopBeforeWaiting`

Inside the event processing loop before the run loop sleeps, waiting for a source or timer to fire. This activity does not occur if `CFRunLoopRunInMode` is called with a timeout of 0 seconds. It also does not occur in a particular iteration of the event processing loop if a version 0 source fires.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

`kCFRunLoopAfterWaiting`

Inside the event processing loop after the run loop wakes up, but before processing the event that woke it up. This activity occurs only if the run loop did in fact go to sleep during the current loop.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

`kCFRunLoopExit`

The exit of the run loop, after exiting the event processing loop. This activity occurs once for each call to `CFRunLoopRun` and `CFRunLoopRunInMode`.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

`kCFRunLoopAllActivities`

A combination of all the preceding stages.

Available in Mac OS X v10.0 and later.

Declared in `CFRunLoop.h`.

**Discussion**

The run loop stages in which an observer is scheduled are selected when the observer is created with [CFRunLoopObserverCreate](#) (page 5).

# Document Revision History

---

This table describes the changes to *CFRunLoopObserver Reference*.

Date	Notes
2006-02-07	Made formatting changes.
2005-08-11	Cosmetic changes to conform to documentation guidelines.
2003-01-01	First version of this document.

## REVISION HISTORY

### Document Revision History

# Index

---

## C

---

CFRunLoopObserverCallback **callback** 9  
CFRunLoopObserverContext **structure** 10  
CFRunLoopObserverCreate **function** 5  
CFRunLoopObserverDoesRepeat **function** 6  
CFRunLoopObserverGetActivities **function** 7  
CFRunLoopObserverGetContext **function** 7  
CFRunLoopObserverGetOrder **function** 8  
CFRunLoopObserverGetTypeID **function** 8  
CFRunLoopObserverInvalidate **function** 8  
CFRunLoopObserverIsValid **function** 9  
CFRunLoopObserverRef **data type** 11

## K

---

kCFRunLoopAfterWaiting **constant** 12  
kCFRunLoopAllActivities **constant** 12  
kCFRunLoopBeforeSources **constant** 11  
kCFRunLoopBeforeTimers **constant** 11  
kCFRunLoopBeforeWaiting **constant** 12  
kCFRunLoopEntry **constant** 11  
kCFRunLoopExit **constant** 12

## R

---

Run Loop Activities 11