
CFFileDescriptor Reference

[Core Foundation](#) > [File Management](#)



2007-05-23



Apple Inc.
© 2007 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 and the Apple logo are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

UNIX is a registered trademark of The Open Group

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

CFFileDescriptor Reference 5

Overview	5
Functions by Task	6
Creating a CFFileDescriptor	6
Getting Information About a File Descriptor	6
Invalidating a File Descriptor	6
Managing Callbacks	6
Creating a Run Loop Source	7
Getting the CFFileDescriptor Type ID	7
Functions	7
CFFileDescriptorCreate	7
CFFileDescriptorCreateRunLoopSource	8
CFFileDescriptorDisableCallbacks	8
CFFileDescriptorEnableCallbacks	9
CFFileDescriptorGetContext	9
CFFileDescriptorGetNativeDescriptor	10
CFFileDescriptorGetTypeID	10
CFFileDescriptorInvalidate	10
CFFileDescriptorIsValid	11
Data Types	11
CFFileDescriptorNativeDescriptor	11
CFFileDescriptorCallBack	12
CFFileDescriptorContext	12
CFFileDescriptorRef	12
Constants	13
Callback Identifiers	13

Document Revision History 15

Index 17

CFFileDescriptor Reference

Derived From:	CType
Framework:	CoreFoundation/CoreFoundation.h
Declared in	CFFileDescriptor.h

Overview

The CFFileDescriptor provides an opaque type to monitor file descriptors for read and write activity via CFRunLoop.

You use CFFileDescriptor to monitor file descriptors for read and write activity via CFRunLoop using callbacks. Each call back is one-shot, and must be re-enabled if you want to get another one.

You can re-enable the callback in the callback function itself, but you must completely service the file descriptor before doing so. For example, if you create a CFFileDescriptor for a pipe and get a callback because there are bytes to be read, then if you don't read all of the bytes but nevertheless re-enable the CFFileDescriptor for read activity, you'll get called back again immediately.

You can monitor kqueue file descriptors for read activity to find out when an event the kqueue is filtering for has occurred. You are responsible for understanding the use of the kevent() API and inserting and removing filters from the kqueue file descriptor yourself.

The following example takes a UNIX process ID as argument, and watches up to 20 seconds, and reports if the process terminates in that time:

```
// cc test.c -framework CoreFoundation -0
#include <CoreFoundation/CoreFoundation.h>
#include <unistd.h>
#include <sys/event.h>
static void noteProcDeath(CFFileDescriptorRef fdref, CFOptionFlags callBackTypes,
void *info) {
    struct kevent kev;
    int fd = CFFileDescriptorGetNativeDescriptor(fdref);
    kevent(fd, NULL, 0, &kev, 1, NULL);
    // take action on death of process here
    printf("process with pid '%u' died\n", (unsigned int)kev.ident);
    CFFileDescriptorInvalidate(fdref);
    CFRelease(fdref); // the CFFileDescriptorRef is no longer of any use in this
example
}
// one argument, an integer pid to watch, required
int main(int argc, char *argv[]) {
    if (argc < 2) exit(1);
    int fd = kqueue();
    struct kevent kev;
```

```

    EV_SET(&kev, atoi(argv[1]), EVFILT_PROC, EV_ADD|EV_ENABLE, NOTE_EXIT, 0,
    NULL);
    kevent(fd, &kev, 1, NULL, 0, NULL);
    CFFileDescriptorRef fdref = CFFileDescriptorCreate(kCFAllocatorDefault, fd,
    true, noteProcDeath, NULL);
    CFFileDescriptorEnableCallbacks(fdref, kCFFileDescriptorReadCallback);
    CFRRunLoopSourceRef source =
    CFFileDescriptorCreateRunLoopSource(kCFAllocatorDefault, fdref, 0);
    CFRRunLoopAddSource(CFRRunLoopGetMain(), source, kCFRunLoopDefaultMode);
    CFRelease(source);
    // run the run loop for 20 seconds
    CFRRunLoopRunInMode(kCFRunLoopDefaultMode, 20.0, false);
    return 0;
}

```

Functions by Task

Creating a CFFileDescriptor

[CFFileDescriptorCreate](#) (page 7)

Creates a new CFFileDescriptor.

Getting Information About a File Descriptor

[CFFileDescriptorGetNativeDescriptor](#) (page 10)

Returns the native file descriptor for a given CFFileDescriptor.

[CFFileDescriptorIsValid](#) (page 11)

Returns a Boolean value that indicates whether the native file descriptor for a given CFFileDescriptor is valid.

[CFFileDescriptorGetContext](#) (page 9)

Gets the context for a given CFFileDescriptor.

Invalidating a File Descriptor

[CFFileDescriptorInvalidate](#) (page 10)

Invalidates the native file descriptor for a given CFFileDescriptor.

Managing Callbacks

[CFFileDescriptorEnableCallbacks](#) (page 9)

Enables callbacks for a given CFFileDescriptor.

[CFFileDescriptorDisableCallbacks](#) (page 8)

Disables callbacks for a given CFFileDescriptor.

Creating a Run Loop Source

[CFFileDescriptorCreateRunLoopSource](#) (page 8)

Creates a new runloop source for a given CFFileDescriptor.

Getting the CFFileDescriptor Type ID

[CFFileDescriptorGetTypeID](#) (page 10)

Returns the type identifier for the CFFileDescriptor opaque type.

Functions

CFFileDescriptorCreate

Creates a new CFFileDescriptor.

```
CFFileDescriptorRef CFFileDescriptorCreate (
    CFAllocatorRef allocator,
    CFFileDescriptorNativeDescriptor fd,
    Boolean closeOnInvalidate,
    CFFileDescriptorCallBack callout,
    const CFFileDescriptorContext *context
);
```

Parameters

allocator

The allocator to use to allocate memory for the new bag and its storage for values. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

fd

The file descriptor for the new CFFileDescriptor.

closeOnInvalidate

`true` if the new CFFileDescriptor should close *fd* when it is invalidated, otherwise `false`.

callout

The CFFileDescriptorCallBack for the new CFFileDescriptor.

context

Contextual information for the new CFFileDescriptor.

Return Value

A new CFFileDescriptor or `NULL` if there was a problem creating the object. Ownership follows the Create Rule.

Availability

Available in Mac OS X v10.5 and later.

See Also

[CFFileDescriptorGetContext](#) (page 9)

[CFFileDescriptorInvalidate](#) (page 10)

Declared In

CFFileDescriptor.h

CFFileDescriptorCreateRunLoopSource

Creates a new runloop source for a given CFFileDescriptor.

```
CFRunLoopSourceRef CFFileDescriptorCreateRunLoopSource (
    CFAllocatorRef allocator,
    CFFileDescriptorRef f,
    CFIndex order
);
```

Parameters*allocator*

The allocator to use to allocate memory for the new bag and its storage for values. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

f

A CFFileDescriptor.

order

The order for the new run loop (see `CFRunLoopSourceCreate`).

Return Value

A new runloop source for *f*, or `NULL` if there was a problem creating the object. Ownership follows the Create Rule.

Discussion

The context for the new runloop (see `CFRunLoopSourceCreate`) is the same as the context passed in when the CFFileDescriptor was created (see [CFFileDescriptorCreate](#) (page 7)).

Availability

Available in Mac OS X v10.5 and later.

Declared In

CFFileDescriptor.h

CFFileDescriptorDisableCallbacks

Disables callbacks for a given CFFileDescriptor.

```
void CFFileDescriptorDisableCallbacks (
    CFFileDescriptorRef f,
    CFOptionFlags callbackTypes
);
```

Parameters*f*

A CFFileDescriptor.

callbackTypes

A bitmask that specifies which callbacks to disable (see [“Callback Identifiers”](#) (page 13) for possible components).

Availability

Available in Mac OS X v10.5 and later.

See Also

[CFFileDescriptorEnableCallbacks](#) (page 9)

Declared In

CFFileDescriptor.h

CFFileDescriptorEnableCallbacks

Enables callbacks for a given CFFileDescriptor.

```
void CFFileDescriptorEnableCallbacks (
    CFFileDescriptorRef f,
    CFOptionFlags callbackTypes
);
```

Parameters

f

A CFFileDescriptor.

callbackTypes

A bitmask that specifies which callbacks to enable (see “[Callback Identifiers](#)” (page 13) for possible components).

Availability

Available in Mac OS X v10.5 and later.

See Also

[CFFileDescriptorDisableCallbacks](#) (page 8)

Declared In

CFFileDescriptor.h

CFFileDescriptorGetContext

Gets the context for a given CFFileDescriptor.

```
void CFFileDescriptorGetContext (
    CFFileDescriptorRef f,
    CFFileDescriptorContext *context
);
```

Parameters

f

A CFFileDescriptor.

context

Upon return, contains the context passed to *f* in [CFFileDescriptorCreate](#) (page 7).

Availability

Available in Mac OS X v10.5 and later.

See Also[CFFileDescriptorCreate](#) (page 7)**Declared In**

CFFileDescriptor.h

CFFileDescriptorGetNativeDescriptor

Returns the native file descriptor for a given CFFileDescriptor.

```
CFFileDescriptorNativeDescriptor CFFileDescriptorGetNativeDescriptor (
    CFFileDescriptorRef f
);
```

Parameters*f*

A CFFileDescriptor.

Return ValueThe native file descriptor for *f*.**Availability**

Available in Mac OS X v10.5 and later.

See Also[CFFileDescriptorInvalidate](#) (page 10)**Declared In**

CFFileDescriptor.h

CFFileDescriptorGetTypeID

Returns the type identifier for the CFFileDescriptor opaque type.

```
CTypeID CFFileDescriptorGetTypeID (
    void
);
```

Return Value

The type identifier for the CFFileDescriptor opaque type.

Availability

Available in Mac OS X v10.5 and later.

Declared In

CFFileDescriptor.h

CFFileDescriptorInvalidate

Invalidates the native file descriptor for a given CFFileDescriptor.

```
void CFFileDescriptorInvalidate (
    CFFileDescriptorRef f,
);
```

Parameters

f
A CFFileDescriptor.

Availability

Available in Mac OS X v10.5 and later.

See Also

[CFFileDescriptorIsValid](#) (page 11)

[CFFileDescriptorGetNativeDescriptor](#) (page 10)

Declared In

CFFileDescriptor.h

CFFileDescriptorIsValid

Returns a Boolean value that indicates whether the native file descriptor for a given CFFileDescriptor is valid.

```
Boolean CFFileDescriptorIsValid (
    CFFileDescriptorRef f,
);
```

Parameters

f
A CFFileDescriptor.

Return Value

true if the native file descriptor for *f* is valid, otherwise false.

Availability

Available in Mac OS X v10.5 and later.

See Also

[CFFileDescriptorInvalidate](#) (page 10)

Declared In

CFFileDescriptor.h

Data Types

CFFileDescriptorNativeDescriptor

Defines a type for the native file descriptor.

```
typedef int CFFileDescriptorNativeDescriptor;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

CFFileDescriptor.h

CFFileDescriptorCallback

Defines a structure for a callback for a CFFileDescriptor.

```
typedef void (*CFFileDescriptorCallback) (
    CFFileDescriptorRef f,
    CFOptionFlags callBackTypes,
    void *info
);
```

Declared In

CFFileDescriptor.h

CFFileDescriptorContext

Defines a structure for the context of a CFFileDescriptor.

```
typedef struct {
    CFIndex version;
    void * info;
    void * (*retain)(void *info);
    void (*release)(void *info);
    CFStringRef (*copyDescription)(void *info);
} CFFileDescriptorContext;
```

Fields

version

The version number of this structure. If not one of the defined version numbers for this opaque type, the behavior is undefined. The current version of this structure is 0.

info

retain

The retain callback used by the CFFileDescriptor.

release

The release callback used by the CFFileDescriptor.

copyDescription

The callback used to create a descriptive string representation of the CFFileDescriptor.

Availability

Available in Mac OS X v10.5 and later.

Declared In

CFFileDescriptor.h

CFFileDescriptorRef

A reference to an CFFileDescriptor object.

```
typedef struct __CFileDescriptor * CFileDescriptorRef;
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

CFileDescriptor.h

Constants

Callback Identifiers

Constants that identify the read and write callbacks.

```
enum {
    kCFileDescriptorReadCallback = 1 << 0,
    kCFileDescriptorWriteCallback = 1 << 1
};
```

Constants

kCFileDescriptorReadCallback

Identifies the read callback.

Available in Mac OS X v10.5 and later.

Declared in CFileDescriptor.h.

kCFileDescriptorWriteCallback

Identifies the write callback.

Available in Mac OS X v10.5 and later.

Declared in CFileDescriptor.h.

Declared In

CFileDescriptor.h

Document Revision History

This table describes the changes to *CFFileDescriptor Reference*.

Date	Notes
2007-05-23	New document that describes the opaque type to monitor file descriptors for read and write activity via CFRunLoop.

REVISION HISTORY

Document Revision History

Index

C

Callback Identifiers [13](#)

CFFileDescriptorCallback **data type** [12](#)

CFFileDescriptorContext **data type** [12](#)

CFFileDescriptorCreate **function** [7](#)

CFFileDescriptorCreateRunLoopSource **function** [8](#)

CFFileDescriptorDisableCallbacks **function** [8](#)

CFFileDescriptorEnableCallbacks **function** [9](#)

CFFileDescriptorGetContext **function** [9](#)

CFFileDescriptorGetNativeDescriptor **function**
[10](#)

CFFileDescriptorGetTypeID **function** [10](#)

CFFileDescriptorInvalidate **function** [10](#)

CFFileDescriptorIsValid **function** [11](#)

CFFileDescriptorNativeDescriptor **data type** [11](#)

CFFileDescriptorRef **data type** [12](#)

K

kCFFileDescriptorReadCallback **constant** [13](#)

kCFFileDescriptorWriteCallback **constant** [13](#)