
NSFileHandle Class Reference

[Cocoa](#) > [File Management](#)



2008-10-15



Apple Inc.
© 2008 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, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

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

NSFileHandle Class Reference 5

Overview	5
Tasks	5
Getting a File Handle	5
Creating a File Handle	6
Getting a File Descriptor	6
Reading from a File Handle	6
Writing to a File Handle	6
Communicating Asynchronously	6
Seeking Within a File	7
Operating on a File	7
Class Methods	8
fileHandleForReadingAtPath:	8
fileHandleForUpdatingAtPath:	8
fileHandleForWritingAtPath:	9
fileHandleWithNullDevice	9
fileHandleWithStandardError	10
fileHandleWithStandardInput	10
fileHandleWithStandardOutput	11
Instance Methods	11
acceptConnectionInBackgroundAndNotify	11
acceptConnectionInBackgroundAndNotifyForModes:	12
availableData	12
closeFile	13
fileDescriptor	13
initWithFileDescriptor:	14
initWithFileDescriptor:closeOnDealloc:	14
offsetInFile	15
readDataOfLength:	15
readDataToEndOfFile	16
readInBackgroundAndNotify	16
readInBackgroundAndNotifyForModes:	17
readToEndOfFileInBackgroundAndNotify	18
readToEndOfFileInBackgroundAndNotifyForModes:	18
seekToEndOfFile	19
seekToFileOffset:	19
synchronizeFile	20
truncateFileAtOffset:	20
waitForDataInBackgroundAndNotify	20
waitForDataInBackgroundAndNotifyForModes:	21
writeData:	21

- Constants 22
 - Keys for Notification UserInfo Dictionary 22
 - Exception Names 23
 - Unused Constant 23
- Notifications 23
 - NSFileHandleConnectionAcceptedNotification 23
 - NSFileHandleDataAvailableNotification 24
 - NSFileHandleReadCompletionNotification 24
 - NSFileHandleReadToEndOfFileCompletionNotification 25

Document Revision History 27

Index 29

NSFileHandle 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.
Companion guide	Low-Level File Management Programming Topics
Declared in	NSFileHandle.h
Related sample code	AudioBurn PictureSharing PictureSharingBrowser

Overview

`NSFileHandle` objects provide an object-oriented wrapper for accessing open files or communications channels.

See the *PictureSharing* example project to examine code that creates an `NSFileHandle` object to listen for incoming connections; the file-handle object is initialized from a socket obtained through BSD calls.

Note: The deallocation of an `NSFileHandle` object deletes its descriptor and closes the represented file or channel unless the `NSFileHandle` object was created with `initWithFileDescriptor:` (page 14) or `initWithFileDescriptor:closeOnDealloc:` (page 14) with `NO` as the parameter argument.

Tasks

Getting a File Handle

- + `fileHandleForReadingAtPath:` (page 8)
Returns a file handle initialized for reading the file, device, or named socket at the specified path.
- + `fileHandleForWritingAtPath:` (page 9)
Returns a file handle initialized for writing to the file, device, or named socket at the specified path.

- + [fileHandleForUpdatingAtPath:](#) (page 8)
Returns a file handle initialized for reading and writing to the file, device, or named socket at the specified path.
- + [fileHandleWithStandardError](#) (page 10)
Returns the file handle associated with the standard error file.
- + [fileHandleWithStandardInput](#) (page 10)
Returns the file handle associated with the standard input file.
- + [fileHandleWithStandardOutput](#) (page 11)
Returns the file handle associated with the standard output file.
- + [fileHandleWithNullDevice](#) (page 9)
Returns a file handle associated with a null device.

Creating a File Handle

- [initWithFileDescriptor:](#) (page 14)
Returns a file handle initialized with a file descriptor.
- [initWithFileDescriptor:closeOnDealloc:](#) (page 14)
Returns a file handle initialized with a file handle, using a specified deallocation policy.

Getting a File Descriptor

- [fileDescriptor](#) (page 13)
Returns the file descriptor associated with the receiver.

Reading from a File Handle

- [availableData](#) (page 12)
Returns the data available through the receiver.
- [readDataToEndOfFile](#) (page 16)
Returns the data available through the receiver up to the end of file or maximum number of bytes.
- [readDataOfLength:](#) (page 15)
Reads data up to a specified number of bytes from the receiver.

Writing to a File Handle

- [writeData:](#) (page 21)
Synchronously writes data to the file, device, pipe, or socket represented by the receiver.

Communicating Asynchronously

- [acceptConnectionInBackgroundAndNotify](#) (page 11)
Accepts a socket connection (for stream-type sockets only) in the background and creates a file handle for the “near” (client) end of the communications channel.

- [acceptConnectionInBackgroundAndNotifyForModes:](#) (page 12)
Accepts a socket connection (for stream-type sockets only) in the background and creates a file handle for the “near” (client) end of the communications channel.
- [readInBackgroundAndNotify](#) (page 16)
Reads from the file or communications channel in the background and posts a notification when finished.
- [readInBackgroundAndNotifyForModes:](#) (page 17)
Reads from the file or communications channel in the background and posts a notification when finished.
- [readToEndOfFileInBackgroundAndNotify](#) (page 18)
Reads to the end of file from the file or communications channel in the background and posts a notification when finished.
- [readToEndOfFileInBackgroundAndNotifyForModes:](#) (page 18)
Reads to the end of file from the file or communications channel in the background and posts a notification when finished.
- [waitForDataInBackgroundAndNotify](#) (page 20)
Checks to see if data is available in a background thread.
- [waitForDataInBackgroundAndNotifyForModes:](#) (page 21)
Checks to see if data is available in a background thread.

Seeking Within a File

- [offsetInFile](#) (page 15)
Returns the position of the file pointer within the file represented by the receiver.
- [seekToEndOfFile](#) (page 19)
Puts the file pointer at the end of the file referenced by the receiver and returns the new file offset.
- [seekToFileOffset:](#) (page 19)
Moves the file pointer to the specified offset within the file represented by the receiver.

Operating on a File

- [closeFile](#) (page 13)
Disallows further access to the represented file or communications channel and signals end of file on communications channels that permit writing.
- [synchronizeFile](#) (page 20)
Causes all in-memory data and attributes of the file represented by the receiver to be written to permanent storage.
- [truncateFileAtOffset:](#) (page 20)
Truncates or extends the file represented by the receiver to a specified offset within the file and puts the file pointer at that position.

Class Methods

fileHandleForReadingAtPath:

Returns a file handle initialized for reading the file, device, or named socket at the specified path.

```
+ (id)fileHandleForReadingAtPath:(NSString *)path
```

Parameters

path

The path to the file, device, or named socket to access.

Return Value

The initialized file handle, or `nil` if no file exists at *path*.

Discussion

The file pointer is set to the beginning of the file. The returned object responds only to `NSFileHandle` `read...` messages.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [availableData](#) (page 12)
- [initWithFileDescriptor:](#) (page 14)
- [readDataOfLength:](#) (page 15)
- [readDataToEndOfFile](#) (page 16)

Related Sample Code

AudioBurn

Declared In

NSFileHandle.h

fileHandleForUpdatingAtPath:

Returns a file handle initialized for reading and writing to the file, device, or named socket at the specified path.

```
+ (id)fileHandleForUpdatingAtPath:(NSString *)path
```

Parameters

path

The path to the file, device, or named socket to access.

Return Value

The initialized file handle, or `nil` if no file exists at *path*.

Discussion

The file pointer is set to the beginning of the file. The returned object responds to both `NSFileHandle` `read...` messages and [writeData:](#) (page 21).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [availableData](#) (page 12)
- [initWithFileDescriptor:](#) (page 14)
- [readDataOfLength:](#) (page 15)
- [readDataToEndOfFile](#) (page 16)

Declared In

NSFileHandle.h

fileHandleForWritingAtPath:

Returns a file handle initialized for writing to the file, device, or named socket at the specified path.

```
+ (id)fileHandleForWritingAtPath:(NSString *)path
```

Parameters

path

The path to the file, device, or named socket to access.

Return Value

The initialized file handle, or `nil` if no file exists at *path*.

Discussion

The file pointer is set to the beginning of the file. The returned object responds only to [writeData:](#) (page 21).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithFileDescriptor:](#) (page 14)

Declared In

NSFileHandle.h

fileHandleWithNullDevice

Returns a file handle associated with a null device.

```
+ (id)fileHandleWithNullDevice
```

Return Value

A file handle associated with a null device.

Discussion

You can use null-device file handles as “placeholders” for standard-device file handles or in collection objects to avoid exceptions and other errors resulting from messages being sent to invalid file handles. Read messages sent to a null-device file handle return an end-of-file indicator (an empty `NSData` object) rather than raise an exception. Write messages are no-ops, whereas [fileDescriptor](#) (page 13) returns an illegal value. Other methods are no-ops or return “sensible” values.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithFileDescriptor:](#) (page 14)

Declared In

NSFileHandle.h

fileHandleWithStandardError

Returns the file handle associated with the standard error file.

```
+ (id)fileHandleWithStandardError
```

Return Value

The shared file handle associated with the standard error file.

Discussion

Conventionally this is a terminal device to which error messages are sent. There is one standard error file handle per process; it is a shared instance.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [fileHandleWithNullDevice](#) (page 9)

- [initWithFileDescriptor:](#) (page 14)

Declared In

NSFileHandle.h

fileHandleWithStandardInput

Returns the file handle associated with the standard input file.

```
+ (id)fileHandleWithStandardInput
```

Return Value

The shared file handle associated with the standard input file.

Discussion

Conventionally this is a terminal device on which the user enters a stream of data. There is one standard input file handle per process; it is a shared instance.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [fileHandleWithNullDevice](#) (page 9)

- [initWithFileDescriptor:](#) (page 14)

Declared In

NSFileHandle.h

fileHandleWithStandardOutput

Returns the file handle associated with the standard output file.

+ (id)fileHandleWithStandardOutput

Return Value

The shared file handle associated with the standard output file.

Discussion

Conventionally this is a terminal device that receives a stream of data from a program. There is one standard output file handle per process; it is a shared instance.

Availability

Available in Mac OS X v10.0 and later.

See Also+ [fileHandleWithNullDevice](#) (page 9)- [initWithFileDescriptor:](#) (page 14)**Declared In**

NSFileHandle.h

Instance Methods

acceptConnectionInBackgroundAndNotify

Accepts a socket connection (for stream-type sockets only) in the background and creates a file handle for the “near” (client) end of the communications channel.

- (void)acceptConnectionInBackgroundAndNotify

DiscussionThis method is asynchronous. In a separate “safe” thread it accepts a connection, creates a file handle for the other end of the connection, and returns that object to the client by posting an [NSFileHandleConnectionAcceptedNotification](#) (page 23) in the run loop of the client. The notification includes as data a *userInfo* dictionary containing the created `NSFileHandle` object; access this object using the `NSFileHandleNotificationFileHandleItem` key.The receiver must be created by an [initWithFileDescriptor:](#) (page 14) message that takes as an argument a stream-type socket created by the appropriate system routine. The object that will write data to the returned file handle must add itself as an observer of [NSFileHandleConnectionAcceptedNotification](#) (page 23).Note that this method does not continue to listen for connection requests after it posts `NSFileHandleConnectionAcceptedNotification`. If you want to keep getting notified, you need to call `acceptConnectionInBackgroundAndNotify` again in your observer method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [enqueueNotification:postingStyle:coalesceMask:forModes:](#) (NSNotificationQueue)
- [readInBackgroundAndNotify](#) (page 16)
- [readToEndOfFileInBackgroundAndNotify](#) (page 18)

Related Sample Code

PictureSharing

Declared In

NSFileHandle.h

acceptConnectionInBackgroundAndNotifyForModes:

Accepts a socket connection (for stream-type sockets only) in the background and creates a file handle for the “near” (client) end of the communications channel.

```
- (void)acceptConnectionInBackgroundAndNotifyForModes:(NSArray *)modes
```

Parameters

modes

The runloop modes in which the connection accepted notification can be posted.

Discussion

See [acceptConnectionInBackgroundAndNotify](#) (page 11) for details of how this method operates. This method differs from [acceptConnectionInBackgroundAndNotify](#) (page 11) in that *modes* specifies the run-loop mode (or modes) in which [NSFileHandleConnectionAcceptedNotification](#) (page 23) can be posted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [enqueueNotification:postingStyle:coalesceMask:forModes:](#) (NSNotificationQueue)
- [readInBackgroundAndNotifyForModes:](#) (page 17)
- [readToEndOfFileInBackgroundAndNotifyForModes:](#) (page 18)

Declared In

NSFileHandle.h

availableData

Returns the data available through the receiver.

```
- (NSData *)availableData
```

Return Value

The data currently available through the receiver.

Discussion

If the receiver is a file, returns the data obtained by reading the file from the file pointer to the end of the file. If the receiver is a communications channel, reads up to a buffer of data and returns it; if no data is available, the method blocks. Returns an empty data object if the end of file is reached. Raises `NSFileHandleOperationException` if attempts to determine file-handle type fail or if attempts to read from the file or channel fail.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [readDataOfLength:](#) (page 15)
- [readDataToEndOfFile](#) (page 16)

Declared In

`NSFileHandle.h`

closeFile

Disallows further access to the represented file or communications channel and signals end of file on communications channels that permit writing.

- (void)closeFile

Discussion

The file or communications channel is available for other uses after the file handle represented by the receiver is closed. Further read and write messages sent to a file handle to which `closeFile` has been sent raises an exception.

Sending `closeFile` to a file handle does not cause its deallocation. The deallocation of an `NSFileHandle` object deletes its descriptor and closes the represented file or channel unless the `NSFileHandle` object was created with [initWithFileDescriptor:](#) (page 14) or [initWithFileDescriptor:closeOnDealloc:](#) (page 14) with `NO` as the parameter argument.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

[PictureSharing](#)

Declared In

`NSFileHandle.h`

fileDescriptor

Returns the file descriptor associated with the receiver.

- (int)fileDescriptor

Return Value

The POSIX file descriptor associated with the receiver.

Discussion

You can send this message to file handles originating from both file descriptors and file handles and receive a valid file descriptor so long as the file handle is open. If the file handle has been closed by sending it [closeFile](#) (page 13), this method raises an exception.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithFileDescriptor:](#) (page 14)

Declared In

NSFileHandle.h

initWithFileDescriptor:

Returns a file handle initialized with a file descriptor.

```
- (id)initWithFileDescriptor:(int)fileDescriptor
```

Parameters

fileDescriptor

The POSIX file descriptor with which to initialize the file handle.

Return Value

A file handle initialized with *fileDescriptor*.

Discussion

You can create a file handle for a socket by using the result of a `socket` call as *fileDescriptor*.

Special Considerations

The object creating a file handle using this method owns *fileDescriptor* and is responsible for its disposition.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [closeFile](#) (page 13)

Declared In

NSFileHandle.h

initWithFileDescriptor:closeOnDealloc:

Returns a file handle initialized with a file handle, using a specified deallocation policy.

```
- (id)initWithFileDescriptor:(int)fileDescriptor closeOnDealloc:(BOOL)flag
```

Parameters

fileDescriptor

The POSIX file descriptor with which to initialize the file handle.

flag

YES if the file descriptor should be closed when the receiver is deallocated, otherwise NO.

Return Value

A file handle initialized with *fileDescriptor* with a deallocation policy specified by *flag*.

Special Considerations

If *flag* is NO, the object creating a file handle using this method owns *fileDescriptor* and is responsible for its disposition.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [closeFile](#) (page 13)

Declared In

NSFileHandle.h

offsetInFile

Returns the position of the file pointer within the file represented by the receiver.

```
- (unsigned long long)offsetInFile
```

Return Value

The position of the file pointer within the file represented by the receiver.

Special Considerations

Raises an exception if the message is sent to a file handle representing a pipe or socket or if the file descriptor is closed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [seekToEndOfFile](#) (page 19)
- [seekToFileOffset:](#) (page 19)

Related Sample Code

AudioBurn

Declared In

NSFileHandle.h

readDataOfLength:

Reads data up to a specified number of bytes from the receiver.

```
- (NSData *)readDataOfLength:(NSUInteger)length
```

Parameters*length*

The number of bytes to read from the receiver.

Return Value

The data available through the receiver up to a maximum of *length* bytes.

Discussion

If the receiver is a file, returns the data obtained by reading from the file pointer to *length* or to the end of the file, whichever comes first. If the receiver is a communications channel, the method reads data from the channel up to *length*. Returns an empty `NSData` object if the file is positioned at the end of the file or if an end-of-file indicator is returned on a communications channel. Raises `NSFileHandleOperationException` if attempts to determine file-handle type fail or if attempts to read from the file or channel fail.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [availableData](#) (page 12)
- [readDataToEndOfFile](#) (page 16)

Declared In

`NSFileHandle.h`

readDataToEndOfFile

Returns the data available through the receiver up to the end of file or maximum number of bytes.

```
- (NSData *)readDataToEndOfFile
```

Return Value

The data available through the receiver up to `UINT_MAX` bytes (the maximum value for unsigned integers) or, if a communications channel, until an end-of-file indicator is returned.

Discussion

This method invokes [readDataOfLength:](#) (page 15) as part of its implementation.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [availableData](#) (page 12)

Declared In

`NSFileHandle.h`

readInBackgroundAndNotify

Reads from the file or communications channel in the background and posts a notification when finished.

```
- (void)readInBackgroundAndNotify
```


Discussion

This method performs an asynchronous [availableData](#) (page 12) operation on a file or communications channel and posts an [NSFileHandleReadCompletionNotification](#) (page 24) to the client process's run loop.

The length of the data is limited to the buffer size of the underlying operating system. The notification includes a *userInfo* dictionary that contains the data read; access this object using the `NSFileHandleNotificationDataItem` key.

Any object interested in receiving this data asynchronously must add itself as an observer of [NSFileHandleReadCompletionNotification](#) (page 24). In communication via stream-type sockets, the receiver is often the object returned in the *userInfo* dictionary of [NSFileHandleConnectionAcceptedNotification](#) (page 23).

Note that this method does not cause a continuous stream of notifications to be sent. If you wish to keep getting notified, you'll also need to call `readInBackgroundAndNotify` in your observer method.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [acceptConnectionInBackgroundAndNotify](#) (page 11)
- [readToEndOfFileInBackgroundAndNotifyForModes:](#) (page 18)
- `enqueueNotification:postingStyle:coalesceMask:forModes:` (`NSNotificationQueue`)

Related Sample Code

Moriarity

Declared In

`NSFileHandle.h`

readInBackgroundAndNotifyForModes:

Reads from the file or communications channel in the background and posts a notification when finished.

```
- (void)readInBackgroundAndNotifyForModes:(NSArray *)modes
```

Parameters

modes

The runloop modes in which the read completion notification can be posted.

Discussion

See [readInBackgroundAndNotify](#) (page 16) for details of how this method operates. This method differs from [readInBackgroundAndNotify](#) (page 16) in that *modes* specifies the run-loop mode (or modes) in which [NSFileHandleReadCompletionNotification](#) (page 24) can be posted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [acceptConnectionInBackgroundAndNotifyForModes:](#) (page 12)
- `enqueueNotification:postingStyle:coalesceMask:forModes:` (`NSNotificationQueue`)

Declared In

NSFileHandle.h

readToEndOfFileInBackgroundAndNotify

Reads to the end of file from the file or communications channel in the background and posts a notification when finished.

```
- (void)readToEndOfFileInBackgroundAndNotify
```

Discussion

This method performs an asynchronous `readToEndOfFile` operation on a file or communications channel and posts an [NSFileHandleReadToEndOfFileCompletionNotification](#) (page 25) to the client process's run loop.

The notification includes a *userInfo* dictionary that contains the data read; access this object using the `NSFileHandleNotificationDataItem` key.

Any object interested in receiving this data asynchronously must add itself as an observer of [NSFileHandleReadToEndOfFileCompletionNotification](#) (page 25). In communication via stream-type sockets, the receiver is often the object returned in the *userInfo* dictionary of [NSFileHandleConnectionAcceptedNotification](#) (page 23).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [acceptConnectionInBackgroundAndNotify](#) (page 11)
- [readToEndOfFileInBackgroundAndNotifyForModes:](#) (page 18)
- `enqueueNotification:postingStyle:coalesceMask:forModes:` (`NSNotificationQueue`)

Related Sample Code

[PictureSharingBrowser](#)

Declared In

NSFileHandle.h

readToEndOfFileInBackgroundAndNotifyForModes:

Reads to the end of file from the file or communications channel in the background and posts a notification when finished.

```
- (void)readToEndOfFileInBackgroundAndNotifyForModes:(NSArray *)modes
```

Parameters

modes

The runloop modes in which the read completion notification can be posted.

Discussion

See [readToEndOfFileInBackgroundAndNotify](#) (page 18) for details of this method's operation. The method differs from [readToEndOfFileInBackgroundAndNotify](#) (page 18) in that *modes* specifies the run-loop mode (or modes) in which [NSFileHandleReadToEndOfFileCompletionNotification](#) (page 25) can be posted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [acceptConnectionInBackgroundAndNotifyForModes:](#) (page 12)
- `enqueueNotification:postingStyle:coalesceMask:forModes:` (NSNotificationQueue)

Declared In

NSFileHandle.h

seekToEndOfFile

Puts the file pointer at the end of the file referenced by the receiver and returns the new file offset.

- (unsigned long long)seekToEndOfFile

Return Value

The file offset with the file pointer at the end of the file. This is therefore equal to the size of the file.

Special Considerations

Raises an exception if the message is sent to an `NSFileHandle` object representing a pipe or socket or if the file descriptor is closed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [offsetInFile](#) (page 15)

Declared In

NSFileHandle.h

seekToFileOffset:

Moves the file pointer to the specified offset within the file represented by the receiver.

- (void)seekToFileOffset:(unsigned long long)offset

Parameters

offset

The offset to seek to.

Special Considerations

Raises an exception if the message is sent to an `NSFileHandle` object representing a pipe or socket, if the file descriptor is closed, or if any other error occurs in seeking.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [offsetInFile](#) (page 15)

Related Sample Code

AudioBurn

Declared In

NSFileHandle.h

synchronizeFile

Causes all in-memory data and attributes of the file represented by the receiver to be written to permanent storage.

```
- (void)synchronizeFile
```

Discussion

This method should be invoked by programs that require the file to always be in a known state. An invocation of this method does not return until memory is flushed.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileHandle.h

truncateFileAtOffset:

Truncates or extends the file represented by the receiver to a specified offset within the file and puts the file pointer at that position.

```
- (void)truncateFileAtOffset:(unsigned long long)offset
```

Parameters

offset

The offset within the file that will mark the new end of the file.

Discussion

If the file is extended (if *offset* is beyond the current end of file), the added characters are null bytes.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileHandle.h

waitForDataInBackgroundAndNotify

Checks to see if data is available in a background thread.

```
- (void)waitForDataInBackgroundAndNotify
```

Discussion

When the data becomes available, the thread notifies all observers with [NSFileHandleDataAvailableNotification](#) (page 24). After the notification has been posted, the thread is terminated.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [waitForDataInBackgroundAndNotifyForModes:](#) (page 21)

Declared In

NSFileHandle.h

waitForDataInBackgroundAndNotifyForModes:

Checks to see if data is available in a background thread.

```
- (void)waitForDataInBackgroundAndNotifyForModes:(NSArray *)modes
```

Parameters

modes

The runloop modes in which the data available notification can be posted.

Discussion

When the data becomes available, the thread notifies all observers with [NSFileHandleDataAvailableNotification](#) (page 24). After the notification has been posted, the thread is terminated. This method differs from [waitForDataInBackgroundAndNotify](#) (page 20) in that *modes* specifies the run-loop mode (or modes) in which [NSFileHandleDataAvailableNotification](#) (page 24) can be posted.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [waitForDataInBackgroundAndNotify](#) (page 20)

Declared In

NSFileHandle.h

writeData:

Synchronously writes data to the file, device, pipe, or socket represented by the receiver.

```
- (void)writeData:(NSData *)data
```

Parameters

data

The data to be written.

Discussion

If the receiver is a file, writing takes place at the file pointer's current position. After it writes the data, the method advances the file pointer by the number of bytes written. Raises an exception if the file descriptor is closed or is not valid, if the receiver represents an unconnected pipe or socket endpoint, if no free space is left on the file system, or if any other writing error occurs.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [availableData](#) (page 12)
- [readDataOfLength:](#) (page 15)
- [readDataToEndOfFile](#) (page 16)

Related Sample Code

PictureSharing

Declared In

NSFileHandle.h

Constants

Keys for Notification UserInfo Dictionary

Strings that are used as keys in a userinfo dictionary in a file handle notification.

```
NSString * const NSFileHandleNotificationFileHandleItem;
NSString * const NSFileHandleNotificationDataItem;
```

Constants

NSFileHandleNotificationFileHandleItem

A key in the userinfo dictionary in a [NSFileHandleConnectionAcceptedNotification](#) (page 23) notification.

The corresponding value is the `NSFileHandle` object representing the “near” end of a socket connection.

Available in Mac OS X v10.0 and later.

Declared in `NSFileHandle.h`.

NSFileHandleNotificationDataItem

A key in the userinfo dictionary in a [NSFileHandleReadCompletionNotification](#) (page 24) and [NSFileHandleReadToEndOfFileCompletionNotification](#) (page 25).

The corresponding value is an `NSData` object containing the available data read from a socket connection.

Available in Mac OS X v10.0 and later.

Declared in `NSFileHandle.h`.

Declared In

NSFileHandle.h

Exception Names

Constant that defines the name of a file operation exception.

```
extern NSString *NSFileHandleOperationException;
```

Constants

`NSFileHandleOperationException`

Raised by `NSFileHandle` if attempts to determine file-handle type fail or if attempts to read from a file or channel fail.

Available in Mac OS X v10.0 and later.

Declared in `NSFileHandle.h`.

Declared In

`NSFileHandle.h`

Unused Constant

Constant that is currently unused.

```
NSString * const NSFileHandleNotificationMonitorModes;
```

Constants

`NSFileHandleNotificationMonitorModes`

Currently unused.

Available in Mac OS X v10.0 and later.

Declared in `NSFileHandle.h`.

Declared In

`NSFileHandle.h`

Notifications

`NSFileHandle` posts several notifications related to asynchronous background I/O operations. They are set to post when the run loop of the thread that started the asynchronous operation is idle.

NSFileHandleConnectionAcceptedNotification

This notification is posted when an `NSFileHandle` object establishes a socket connection between two processes, creates an `NSFileHandle` object for one end of the connection, and makes this object available to observers by putting it in the `userInfo` dictionary. To cause the posting of this notification, you must send either [acceptConnectionInBackgroundAndNotify](#) (page 11) or [acceptConnectionInBackgroundAndNotifyForModes:](#) (page 12) to an `NSFileHandle` object representing a server stream-type socket.

The notification object is the `NSFileHandle` object that sent the notification. The `userInfo` dictionary contains the following information:

Key	Value
NSFileHandleNotificationFileHandleItem	The <code>NSFileHandle</code> object representing the “near” end of a socket connection
@“NSFileHandleError”	An <code>NSNumber</code> object containing an integer representing the UNIX-type error which occurred

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileHandle.h

NSFileHandleDataAvailableNotification

This notification is posted when the background thread determines that data is currently available for reading in a file or at a communications channel. The observers can then issue the appropriate messages to begin reading the data. To cause the posting of this notification, you must send either [waitForDataInBackgroundAndNotify](#) (page 20) or [waitForDataInBackgroundAndNotifyForModes:](#) (page 21) to an appropriate `NSFileHandle` object.

The notification object is the `NSFileHandle` object that sent the notification. This notification does not contain a `userInfo` dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileHandle.h

NSFileHandleReadCompletionNotification

This notification is posted when the background thread reads the data currently available in a file or at a communications channel. It makes the data available to observers by putting it in the `userInfo` dictionary. To cause the posting of this notification, you must send either [readInBackgroundAndNotify](#) (page 16) or [readInBackgroundAndNotifyForModes:](#) (page 17) to an appropriate `NSFileHandle` object.

The notification object is the `NSFileHandle` object that sent the notification. The `userInfo` dictionary contains the following information:

Key	Value
NSFileHandleNotificationDataItem	An <code>NSData</code> object containing the available data read from a socket connection
@“NSFileHandleError”	An <code>NSNumber</code> object containing an integer representing the UNIX-type error which occurred

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileHandle.h

NSFileHandleReadToEndOfFileCompletionNotification

This notification is posted when the background thread reads all data in the file or, if a communications channel, until the other process signals the end of data. It makes the data available to observers by putting it in the *userInfo* dictionary. To cause the posting of this notification, you must send either [readToEndOfFileInBackgroundAndNotify](#) (page 18) or [readToEndOfFileInBackgroundAndNotifyForModes:](#) (page 18) to an appropriate NSFileHandle object.

The notification object is the NSFileHandle object that sent the notification. The *userInfo* dictionary contains the following information:

Key	Value
NSFileHandleNotificationDataItem	An NSData object containing the available data read from a socket connection
@ "NSFileHandleError"	An NSNumber object containing an integer representing the UNIX-type error which occurred

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSFileHandle.h

Document Revision History

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

Date	Notes
2008-10-15	Clarified description of <code>closeFile</code> and added link to related sample code project.
2007-01-08	Added definition of <code>NSFileHandleNotificationMonitorModes</code> .
2006-12-12	Updated for Mac OS X v10.5.
2006-05-23	Added declarations for <code>NSFileHandleNotificationDataItem</code> and <code>NSFileHandleNotificationFileHandleItem</code> .
	Added declarations for <code>NSFileHandleNotificationDataItem</code> and <code>NSFileHandleNotificationFileHandleItem</code> .
	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`acceptConnectionInBackgroundAndNotify` **instance method** [11](#)
`acceptConnectionInBackgroundAndNotifyForModes:` **instance method** [12](#)
`availableData` **instance method** [12](#)

C

`closeFile` **instance method** [13](#)

E

Exception Names [23](#)

F

`fileDescriptor` **instance method** [13](#)
`fileHandleForReadingAtPath:` **class method** [8](#)
`fileHandleForUpdatingAtPath:` **class method** [8](#)
`fileHandleForWritingAtPath:` **class method** [9](#)
`fileHandleWithNullDevice` **class method** [9](#)
`fileHandleWithStandardError` **class method** [10](#)
`fileHandleWithStandardInput` **class method** [10](#)
`fileHandleWithStandardOutput` **class method** [11](#)

I

`initWithFileDescriptor:` **instance method** [14](#)
`initWithFileDescriptor:closeOnDealloc:` **instance method** [14](#)

K

Keys for Notification UserInfo Dictionary [22](#)

N

`NSFileHandleConnectionAcceptedNotification` **notification** [23](#)
`NSFileHandleDataAvailableNotification` **notification** [24](#)
`NSFileHandleNotificationDataItem` **constant** [22](#)
`NSFileHandleNotificationFileHandleItem` **constant** [22](#)
`NSFileHandleNotificationMonitorModes` **constant** [23](#)
`NSFileHandleOperationException` **constant** [23](#)
`NSFileHandleReadCompletionNotification` **notification** [24](#)
`NSFileHandleReadToEndOfFileCompletionNotification` **notification** [25](#)

O

`offsetInFile` **instance method** [15](#)

R

`readDataOfLength:` **instance method** [15](#)
`readDataToEndOfFile` **instance method** [16](#)
`readInBackgroundAndNotify` **instance method** [16](#)
`readInBackgroundAndNotifyForModes:` **instance method** [17](#)
`readToEndOfFileInBackgroundAndNotify` **instance method** [18](#)
`readToEndOfFileInBackgroundAndNotifyForModes:` **instance method** [18](#)

S

seekToEndOfFile **instance method** [19](#)
seekToFileOffset: **instance method** [19](#)
synchronizeFile **instance method** [20](#)

T

truncateFileAtOffset: **instance method** [20](#)

U

Unused Constant [23](#)

W

waitForDataInBackgroundAndNotify **instance method** [20](#)
waitForDataInBackgroundAndNotifyForModes: **instance method** [21](#)
writeData: **instance method** [21](#)