

---

# CFReadStream Reference

Core Foundation



2007-05-03



Apple Inc.  
© 2003, 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, the Apple logo, Cocoa, Mac, and Mac OS 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

---

## **CFReadStream Reference 5**

---

Overview	5
Functions by Task	5
Creating a Read Stream	5
Opening and Closing a Read Stream	5
Reading from a Stream	6
Scheduling a Read Stream	6
Examining Stream Properties	6
Setting Stream Properties	6
Getting the CFReadStream Type ID	6
Functions	7
CFReadStreamClose	7
CFReadStreamCopyError	7
CFReadStreamCopyProperty	8
CFReadStreamCreateWithBytesNoCopy	8
CFReadStreamCreateWithFile	9
CFReadStreamGetBuffer	9
CFReadStreamGetError	10
CFReadStreamGetStatus	11
CFReadStreamGetTypeID	11
CFReadStreamHasBytesAvailable	11
CFReadStreamOpen	12
CFReadStreamRead	12
CFReadStreamScheduleWithRunLoop	13
CFReadStreamSetClient	14
CFReadStreamSetProperty	15
CFReadStreamUnscheduleFromRunLoop	16
Callbacks	16
CFReadStreamClientCallBack	16
Data Types	17
CFReadStreamRef	17
CFStreamClientContext	18

---

## **Document Revision History 19**

---

## **Index 21**

---



# CFReadStream Reference

---

<b>Derived From:</b>	CFType
<b>Framework:</b>	CoreFoundation/CoreFoundation.h
<b>Declared in</b>	CFStream.h

## Overview

`CFReadStream` provides an interface for reading a byte stream either synchronously or asynchronously. You can create streams that read bytes from a block of memory, a file, or a generic socket. All streams need to be opened, using [CFReadStreamOpen](#) (page 12), before reading.

Use `CFWriteStream` for writing byte streams. The `CFNetwork` framework defines an additional type of stream for reading responses to HTTP requests.

`CFReadStream` is “toll-free bridged” with its Cocoa Foundation counterpart, `NSInputStream`. 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 `NSInputStream *` parameter, you can pass in a `CFReadStreamRef`, and in a function where you see a `CFReadStreamRef` parameter, you can pass in an `NSInputStream` instance. Note, however, that you may have either a delegate or callbacks but not both. See [Interchangeable Data Types](#) for more information on toll-free bridging.

## Functions by Task

### Creating a Read Stream

[CFReadStreamCreateWithBytesNoCopy](#) (page 8)  
Creates a readable stream for a block of memory.

[CFReadStreamCreateWithFile](#) (page 9)  
Creates a readable stream for a file.

### Opening and Closing a Read Stream

[CFReadStreamClose](#) (page 7)  
Closes a readable stream.

[CFReadStreamOpen](#) (page 12)  
Opens a stream for reading.

## Reading from a Stream

[CFReadStreamRead](#) (page 12)

Reads data from a readable stream.

## Scheduling a Read Stream

[CFReadStreamScheduleWithRunLoop](#) (page 13)

Schedules a stream into a run loop.

[CFReadStreamUnscheduleFromRunLoop](#) (page 16)

Removes a read stream from a given run loop.

## Examining Stream Properties

[CFReadStreamCopyProperty](#) (page 8)

Returns the value of a property for a stream.

[CFReadStreamGetBuffer](#) (page 9)

Returns a pointer to a stream's internal buffer of unread data, if possible.

[CFReadStreamCopyError](#) (page 7)

Returns the error associated with a stream.

[CFReadStreamGetError](#) (page 10)

Returns the error status of a stream. (**Deprecated.** Use [CFReadStreamCopyError](#) (page 7) instead.)

[CFReadStreamGetStatus](#) (page 11)

Returns the current state of a stream.

[CFReadStreamHasBytesAvailable](#) (page 11)

Returns a Boolean value that indicates whether a readable stream has data that can be read without blocking.

## Setting Stream Properties

[CFReadStreamSetClient](#) (page 14)

Assigns a client to a stream, which receives callbacks when certain events occur.

[CFReadStreamSetProperty](#) (page 15)

Sets the value of a property for a stream.

## Getting the CFReadStream Type ID

[CFReadStreamGetTypeID](#) (page 11)

Returns the type identifier the `CFReadStream` opaque type.

## Functions

### CFReadStreamClose

Closes a readable stream.

```
void CFReadStreamClose (
    CFReadStreamRef stream
);
```

#### Parameters

*stream*

The stream to close.

#### Discussion

This function terminates the flow of bytes and releases any system resources required by the stream. The stream is removed from any run loops in which it was scheduled. Once closed, the stream cannot be reopened.

#### Availability

Available in Mac OS X v10.1 and later.

#### Related Sample Code

CFFTPSSample

CFNetworkHTTPDownload

ImageClient

#### Declared In

CFStream.h

### CFReadStreamCopyError

Returns the error associated with a stream.

```
CFErrorRef CFReadStreamCopyError (
    CFReadStreamRef stream
);
```

#### Parameters

*stream*

The stream to examine.

#### Return Value

A CFError object that describes the current problem with stream, or NULL if there is no error. Ownership follows the Create Rule.

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

CFStream.h

## CFReadStreamCopyProperty

Returns the value of a property for a stream.

```

CTypeRef CFReadStreamCopyProperty (
    CFReadStreamRef stream,
    CFStringRef propertyName
);

```

### Parameters

*stream*

The stream to examine.

*propertyName*

The name of the stream property to obtain. The available properties for standard Core Foundation streams are listed in *CFStream Reference*.

### Return Value

The value of the property *propertyName*. Ownership follows the Create Rule.

### Discussion

Each type of stream can define a set of properties that either describe or configure individual streams. A property can be any information about a stream, other than the actual data the stream handles. Examples include the headers from an HTTP transmission, the expected number of bytes, file permission information, and so on. Use [CFReadStreamSetProperty](#) (page 15) to modify the value of a property, although some properties are read-only.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

CFFTPSample

ImageClient

### Declared In

CFStream.h

## CFReadStreamCreateWithBytesNoCopy

Creates a readable stream for a block of memory.

```

CFReadStreamRef CFReadStreamCreateWithBytesNoCopy (
    CFAllocatorRef alloc,
    const UInt8 *bytes,
    CFIndex length,
    CFAllocatorRef bytesDeallocator
);

```

### Parameters

*alloc*

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

*bytes*

The memory buffer to read. This memory must exist for the lifetime of the new stream.



*length*

The size of *bytes*.

*bytesDeallocator*

The allocator to use to deallocate *bytes* when the stream is deallocated. Pass `kCFAllocatorNull` to prevent the stream from deallocating *bytes*.

#### Return Value

The new read stream, or `NULL` on failure. Ownership follows the Create Rule.

#### Discussion

You must open the stream, using [CFReadStreamOpen](#) (page 12), before reading from it.

#### Availability

Available in Mac OS X v10.1 and later.

#### Declared In

CFStream.h

## CFReadStreamCreateWithFile

Creates a readable stream for a file.

```
CFReadStreamRef CFReadStreamCreateWithFile (
    CFAllocatorRef alloc,
    CFURLRef fileURL
);
```

#### Parameters

*alloc*

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

*fileURL*

The URL of the file to read. The URL must use the file scheme.

#### Return Value

The new readable stream object, or `NULL` on failure. Ownership follows the Create Rule.

#### Discussion

You must open the stream, using [CFReadStreamOpen](#) (page 12), before reading from it.

#### Availability

Available in Mac OS X v10.1 and later.

#### Related Sample Code

CFFTPSample

#### Declared In

CFStream.h

## CFReadStreamGetBuffer

Returns a pointer to a stream's internal buffer of unread data, if possible.

```
const UInt8 * CFReadStreamGetBuffer (
    CFReadStreamRef stream,
    CFIndex maxBytesToRead,
    CFIndex *numBytesRead
);
```

**Parameters***stream*

The stream to examine.

*maxBytesToRead*The maximum number of bytes to read. If greater than 0, *maxBytesToRead* limits the number of bytes read; if 0 or less, all available bytes are read.*numBytesRead*On return, contains the length of returned buffer. If *stream* is not open or has encountered an error, *numBytesRead* is set to -1.**Return Value**

A pointer to the internal buffer of unread data for *stream*, if possible; NULL otherwise. The buffer is good only until the next stream operation called on the stream. You should neither change the contents of the returned buffer nor attempt to deallocate the buffer; it is still owned by the stream. The bytes returned in the buffer are considered read from the stream.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

CFStream.h

**CFReadStreamGetError**Returns the error status of a stream. (**Deprecated.** Use [CFReadStreamCopyError](#) (page 7) instead.)

```
CFStreamError CFReadStreamGetError (
    CFReadStreamRef stream
);
```

**Parameters***stream*

The stream to examine.

**Return Value**The error status of *stream* returned in a `CFStreamError` structure.

The error field is 0 if no error has occurred. If the error field is not 0, the `domain` field contains a code that identifies the domain in which the value of the `error` field should be interpreted.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

CFFTPSample

**Declared In**

CFStream.h

## CFReadStreamGetStatus

Returns the current state of a stream.

```
CFStreamStatus CFReadStreamGetStatus (
    CFReadStreamRef stream
);
```

### Parameters

*stream*

The stream to examine.

### Return Value

The current state of *stream*. See `CFStreamStatus` for the list of possible states.

### Availability

Available in Mac OS X v10.1 and later.

### Declared In

CFStream.h

## CFReadStreamGetTypeID

Returns the type identifier the `CFReadStream` opaque type.

```
CTypeID CFReadStreamGetTypeID (
    void
);
```

### Return Value

The type identifier for the `CFReadStream` opaque type.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

CFFTPSSample

### Declared In

CFStream.h

## CFReadStreamHasBytesAvailable

Returns a Boolean value that indicates whether a readable stream has data that can be read without blocking.

```
Boolean CFReadStreamHasBytesAvailable (
    CFReadStreamRef stream
);
```

### Parameters

*stream*

The stream to examine.

**Return Value**

TRUE if data can be read from *stream* without blocking, otherwise FALSE. If *stream* cannot tell if data is available without actually trying to read the data, this function returns TRUE.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

CFStream.h

**CFReadStreamOpen**

Opens a stream for reading.

```
Boolean CFReadStreamOpen (
    CFReadStreamRef stream
);
```

**Parameters**

*stream*

The stream to open.

**Return Value**

TRUE if *stream* was successfully opened, FALSE otherwise. If *stream* is not in the `kCFStreamStatusNotOpen` state, this function returns FALSE.

**Discussion**

Opening a stream causes it to reserve all the system resources it requires. If the stream can open in the background without blocking, this function always returns `true`. To learn when a background open operation completes, you can either schedule the stream into a run loop with [CFReadStreamScheduleWithRunLoop](#) (page 13) and wait for the stream's client (set with [CFReadStreamSetClient](#) (page 14)) to be notified or you can poll the stream using [CFReadStreamGetStatus](#) (page 11), waiting for a status of `kCFStreamStatusOpen` or `kCFStreamStatusError`.

You do not need to wait until a stream has finished opening in the background before calling the [CFReadStreamRead](#) (page 12) function. The read operation will simply block until the open has completed.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

CFFTPSample

CFNetworkHTTPDownload

ImageClient

**Declared In**

CFStream.h

**CFReadStreamRead**

Reads data from a readable stream.

```

CFIndex CFReadStreamRead (
    CFReadStreamRef stream,
    UInt8 *buffer,
    CFIndex bufferSize
);

```

**Parameters***stream*

The stream from which to read.

*buffer*

The buffer into which to place the data.

*bufferLength*The size of *buffer* and the maximum number of bytes to read.**Return Value**

The number of bytes read; 0 if the stream has reached its end; or -1 if either the stream is not open or an error occurs.

**Discussion**

If *stream* is in the process of opening, this function waits until it has completed. This function blocks until at least one byte is available; it does not block until *buffer* is filled. To avoid blocking, call this function only if [CFReadStreamHasBytesAvailable](#) (page 11) returns TRUE or after the stream's client (set with [CFReadStreamSetClient](#) (page 14)) is notified of a `kCFStreamEventHasBytesAvailable` event.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

CFFTPSample

CFNetworkHTTPDownload

ImageClient

**Declared In**

CFStream.h

**CFReadStreamScheduleWithRunLoop**

Schedules a stream into a run loop.

```

void CFReadStreamScheduleWithRunLoop (
    CFReadStreamRef stream,
    CFRunLoopRef runLoop,
    CFStringRef runLoopMode
);

```

**Parameters***stream*

The stream to schedule.

*runLoop*The run loop with which to schedule *stream*.*runLoopMode*The run loop mode of *runLoop* in which to schedule *stream*.

**Discussion**

After scheduling *stream* with a run loop, its client (set with [CFReadStreamSetClient](#) (page 14)) is notified when various events happen with the stream, such as when it finishes opening, when it has bytes available, and when an error occurs. A stream can be scheduled with multiple run loops and run loop modes. Use [CFReadStreamUnscheduleFromRunLoop](#) (page 16) to later remove *stream* from the run loop.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

CFFTPsample

CFNetworkHTTPDownload

ImageClient

**Declared In**

CFStream.h

**CFReadStreamSetClient**

Assigns a client to a stream, which receives callbacks when certain events occur.

```
Boolean CFReadStreamSetClient (
    CFReadStreamRef stream,
    CFOptionFlags streamEvents,
    CFReadStreamClientCallback clientCB,
    CFStreamClientContext *clientContext
);
```

**Parameters**

*stream*

The stream to modify.

*streamEvents*

The set of events for which the client should receive callbacks. The events are listed in `CFStreamEventType`. If you pass `kCFStreamEventNone`, the current client for *stream* is removed.

*clientCB*

The client callback function to be called when one of the events requested in *streamEvents* occurs. If NULL, the current client for *stream* is removed.

*clientContext*

A structure holding contextual information for the stream client. The function copies the information out of the structure, so the memory pointed to by *clientContext* does not need to persist beyond the function call. If NULL, the current client for *stream* is removed.

**Return Value**

TRUE if the stream supports asynchronous notification, otherwise FALSE.

**Discussion**

To avoid polling and blocking, you can register a client to hear about interesting events that occur on a stream. Only one client per stream is allowed; registering a new client replaces the previous one.

Once you have set a client, you need to schedule the stream in a run loop using [CFReadStreamScheduleWithRunLoop](#) (page 13) so that the client can receive the asynchronous notifications. You can schedule each stream in multiple run loops (for example, if you are using a thread pool). It is the caller's responsibility to ensure that at least one of the scheduled run loops is being run, otherwise the callback cannot be called.

Although all Core Foundation streams currently support asynchronous notification, future stream types may not. If a stream does not support asynchronous notification, this function returns `false`. Typically, such streams never block for device I/O (for example, a stream reading memory) and don't benefit from asynchronous notification.

#### Availability

Available in Mac OS X v10.1 and later.

#### Related Sample Code

CFFTPSample

CFNetworkHTTPDownload

ImageClient

#### Declared In

CFStream.h

## CFReadStreamSetProperty

Sets the value of a property for a stream.

```
Boolean CFReadStreamSetProperty (
    CFReadStreamRef stream,
    CFStringRef propertyName,
    CTypeRef propertyValue
);
```

#### Parameters

*stream*

The stream to modify.

*propertyName*

The name of the property to set. The available properties for standard Core Foundation streams are listed in *CFStream Reference*.

*propertyValue*

The value to which to set the property *propertyName* for *stream*. The allowed data type of the value depends on the property being set.

#### Return Value

TRUE if *stream* recognizes and accepts the given property-value pair, otherwise FALSE.

#### Discussion

Each type of stream can define a set of properties that either describe or configure individual streams. A property can be any interesting information about a stream. Examples include the headers from an HTTP transmission, the expected number of bytes, file permission information, and so on. Properties that can be set configure the behavior of the stream and may be modifiable only at particular times, such as before the stream has been opened. (In fact, you should assume that you can set properties only before opening the stream, unless otherwise noted.) To read the value of a property use [CFReadStreamCopyProperty](#) (page 8), although some properties are write-only.

**Availability**

Available in Mac OS X v10.2 and later.

**Related Sample Code**

CFFTPSSample

CocoaEcho

CocoaHTTPServer

CocoaSOAP

ImageClient

**Declared In**

CFStream.h

**CFReadStreamUnscheduleFromRunLoop**

Removes a read stream from a given run loop.

```
void CFReadStreamUnscheduleFromRunLoop (
    CFReadStreamRef stream,
    CFRunLoopRef runLoop,
    CFStringRef runLoopMode
);
```

**Parameters**

*stream*

The stream to unschedule.

*runLoop*

The run loop from which to remove *stream*.

*runLoopMode*

The run loop mode of *runLoop* from which to remove *stream*.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

CFFTPSSample

CFNetworkHTTPDownload

ImageClient

**Declared In**

CFStream.h

## Callbacks

**CFReadStreamClientCallback**

Callback invoked when certain types of activity takes place on a readable stream.



```
typedef void (*CFReadStreamClientCallback) (
    CFReadStreamRef stream,
    CFStreamEventType eventType,
    void *clientCallbackInfo
);
```

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

```
void MyCallback (
    CFReadStreamRef stream,
    CFStreamEventType eventType,
    void *clientCallbackInfo
);
```

### Parameters

*stream*

The stream that experienced the event *eventType*.

*eventType*

The event that caused the callback to be called. The possible events are listed in `CFStreamEventType`.

*clientCallbackInfo*

The `info` member of the `CFStreamClientContext` (page 18) structure that was used when setting the client for *stream*.

### Discussion

This callback is called only for the events requested when setting the client with `CFReadStreamSetClient` (page 14).

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`CFStream.h`

## Data Types

### CFReadStreamRef

A reference to a readable stream object.

```
typedef struct __CFReadStream *CFReadStreamRef;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

`CFStream.h`

## CFStreamClientContext

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

```

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

```

### Fields

version

Version number of the structure. Must be 0.

info

An arbitrary pointer to program-defined data, which can be associated with the client. This pointer is passed to the callbacks defined in the context and to the client callback function [CFReadStreamClientCallback](#) (page 16).

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

CFStream.h

# Document Revision History

---

This table describes the changes to *CFReadStream Reference*.

Date	Notes
2007-05-03	Updated to include API introduced in Mac OS X v10.5.
2006-05-23	Corrected typographical errors.
2006-04-04	Corrected minor typographical error.
2006-01-10	Moved generic constants to CFStream Reference.
2005-11-09	Removed reference to retired document.
2005-03-03	Added information about toll-free bridging.
2003-08-01	Added descriptions of new Mac OS X v10.3 API.
2003-01-01	First version of this document.

## REVISION HISTORY

### Document Revision History

# Index

---

## C

---

CFReadStreamClientCallback **callback** 16  
CFReadStreamClose **function** 7  
CFReadStreamCopyError **function** 7  
CFReadStreamCopyProperty **function** 8  
CFReadStreamCreateWithBytesNoCopy **function** 8  
CFReadStreamCreateWithFile **function** 9  
CFReadStreamGetBuffer **function** 9  
CFReadStreamGetError **function** 10  
CFReadStreamGetStatus **function** 11  
CFReadStreamGetTypeID **function** 11  
CFReadStreamHasBytesAvailable **function** 11  
CFReadStreamOpen **function** 12  
CFReadStreamRead **function** 12  
CFReadStreamRef **data type** 17  
CFReadStreamScheduleWithRunLoop **function** 13  
CFReadStreamSetClient **function** 14  
CFReadStreamSetProperty **function** 15  
CFReadStreamUnscheduleFromRunLoop **function** 16  
CFStreamClientContext **structure** 18