CFWriteStream Reference

Core Foundation

2007-05-03

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

CFWriteStream Reference 5

Overview 5
Functions by Task 5
- Creating a Write Stream 5
- Opening and Closing a Stream 5
- Writing to a Stream 6
- Scheduling a Write Stream 6
- Examining Stream Properties 6
- Setting Stream Properties 6
- Getting the CFWriteStream Type ID 6

Functions 7
- CFWriteStreamCanAcceptBytes 7
- CFWriteStreamClose 7
- CFWriteStreamCopyError 7
- CFWriteStreamCopyProperty 8
- CFWriteStreamCreateWithAllocatedBuffers 8
- CFWriteStreamCreateWithBuffer 9
- CFWriteStreamCreateWithFile 10
- CFWriteStreamGetError 10
- CFWriteStreamGetStatus 11
- CFWriteStreamGetTypeID 11
- CFWriteStreamOpen 12
- CFWriteStreamScheduleWithRunLoop 12
- CFWriteStreamSetClient 13
- CFWriteStreamSetProperty 14
- CFWriteStreamUnscheduleFromRunLoop 15
- CFWriteStreamWrite 15

Callbacks 16
- CFWriteStreamClientCallBack 16

Data Types 17
- CFWriteStreamRef 17

Document Revision History 19

Index 21

CFWriteStream Reference

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

Overview

`CFWriteStream` provides an interface for writing a byte stream either synchronously or asynchronously. You can create streams that write bytes to a block of memory, a file, or a generic socket. All streams need to be opened, using CFWriteStreamOpen (page 12), before writing.

Use CFReadStream for reading byte streams, and for the functions, such as `CFStreamCreatePairWithSocketToHost`, that create socket streams).

`CFWriteStream` is "toll-free bridged" with its Cocoa Foundation counterpart, NSOutputStream. 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 `NSOutputStream *` parameter, you can pass in a `CFWriteStreamRef`, and in a function where you see a `CFWriteStreamRef` parameter, you can pass in an `NSOutputStream` 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 Write Stream

`CFWriteStreamCreateWithAllocatedBuffers` (page 8)
Creates a writable stream for a growable block of memory.

`CFWriteStreamCreateWithBuffer` (page 9)
Creates a writable stream for a fixed-size block of memory.

`CFWriteStreamCreateWithFile` (page 10)
Creates a writable stream for a file.

Opening and Closing a Stream

`CFWriteStreamClose` (page 7)
Closes a writable stream.

`CFWriteStreamOpen` (page 12)
Opens a stream for writing.

Writing to a Stream

`CFWriteStreamWrite` (page 15)
Writes data to a writable stream.

Scheduling a Write Stream

`CFWriteStreamScheduleWithRunLoop` (page 12)
Schedules a stream into a run loop.

`CFWriteStreamUnscheduleFromRunLoop` (page 15)
Removes a stream from a particular run loop.

Examining Stream Properties

`CFWriteStreamCanAcceptBytes` (page 7)
Returns whether a writable stream can accept new data without blocking.

`CFWriteStreamCopyProperty` (page 8)
Returns the value of a property for a stream.

`CFWriteStreamCopyError` (page 7)
Returns the error associated with a stream.

`CFWriteStreamGetError` (page 10)
Returns the error status of a stream. (Deprecated. Use `CFWriteStreamCopyError` (page 7) instead.)

`CFWriteStreamGetStatus` (page 11)
Returns the current state of a stream.

Setting Stream Properties

`CFWriteStreamSetClient` (page 13)
Assigns a client to a stream, which receives callbacks when certain events occur.

`CFWriteStreamSetProperty` (page 14)
Sets the value of a property for a stream.

Getting the CFWriteStream Type ID

`CFWriteStreamGetTypeID` (page 11)
Returns the type identifier of all CFWriteStream objects.

Functions

CFWriteStreamCanAcceptBytes

Returns whether a writable stream can accept new data without blocking.

```
Boolean CFWriteStreamCanAcceptBytes (
   CFWriteStreamRef stream
);
```

Parameters

stream

The stream to examine.

Return Value

`true` if data can be written to *stream* without blocking, `false` otherwise. If *stream* cannot tell if data can be written without actually trying to write the data, this function returns `true`.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`CFStream.h`

CFWriteStreamClose

Closes a writable stream.

```
void CFWriteStreamClose (
   CFWriteStreamRef 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

CFFTPSample

Declared In

`CFStream.h`

CFWriteStreamCopyError

Returns the error associated with a stream.

```
CFErrorRef CFWriteStreamCopyError (
    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`

CFWriteStreamCopyProperty

Returns the value of a property for a stream.

```
CFTypeRef CFWriteStreamCopyProperty (
   CFWriteStreamRef 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 "Stream Properties."

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 interesting information about a stream. Examples include the headers from an HTTP transmission, the expected number of bytes, file permission information, and so on. Use `CFWriteStreamSetProperty` (page 14) to modify the value of a property, although some properties are read-only.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`CFStream.h`

CFWriteStreamCreateWithAllocatedBuffers

Creates a writable stream for a growable block of memory.

```
CFWriteStreamRef CFWriteStreamCreateWithAllocatedBuffers (
   CFAllocatorRef alloc,
   CFAllocatorRef bufferAllocator
);
```

Parameters

alloc

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

bufferAllocator

The allocator to use to allocate memory for the stream's memory buffers. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

Return Value

A new write stream. Ownership follows the Create Rule.

Discussion

New buffers are allocated using *bufferAllocator* as bytes are written to the stream. At any point, you can recover the bytes thus far written by asking for the property `kCFStreamPropertyDataWritten` with `CFWriteStreamCopyProperty` (page 8).

You must open the stream, using CFWriteStreamOpen (page 12), before writing to it.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`CFStream.h`

CFWriteStreamCreateWithBuffer

Creates a writable stream for a fixed-size block of memory.

```
CFWriteStreamRef CFWriteStreamCreateWithBuffer (
   CFAllocatorRef alloc,
   UInt8 *buffer,
   CFIndex bufferCapacity
);
```

Parameters

alloc

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

buffer

The memory buffer into which to write data. This buffer must exist for the lifetime of the stream.

bufferCapacity

The size of *buffer* and the maximum number of bytes that can be written.

Return Value

A new write stream, or `NULL` on failure. Ownership follows the Create Rule.

Discussion

When *buffer* is filled after writing *bufferCapacity* bytes, the stream is exhausted and its status becomes `kCFStreamStatusAtEnd`.

You must open the stream, using CFWriteStreamOpen (page 12), before writing to it.

Availability
Available in Mac OS X v10.1 and later.

Declared In
CFStream.h

CFWriteStreamCreateWithFile

Creates a writable stream for a file.

```
CFWriteStreamRef CFWriteStreamCreateWithFile (
   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 which to write. The URL must use a file scheme.

Return Value
The new write stream, or NULL on failure. Ownership follows the Create Rule.

Discussion
The stream overwrites an existing file unless you set the kCFStreamPropertyAppendToFile to kCFBooleanTrue with CFWriteStreamSetProperty (page 14), in which case the stream appends data to the file.

You must open the stream, using CFWriteStreamOpen (page 12), before writing to it.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
CFStream.h

CFWriteStreamGetError

Returns the error status of a stream. (**Deprecated.** Use CFWriteStreamCopyError (page 7) instead.)

```
CFStreamError CFWriteStreamGetError (
   CFWriteStreamRef stream
);
```

Parameters

stream

The stream to examine.

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

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
`CFStream.h`

CFWriteStreamGetStatus

Returns the current state of a stream.

```
CFStreamStatus CFWriteStreamGetStatus (
   CFWriteStreamRef 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`

CFWriteStreamGetTypeID

Returns the type identifier of all CFWriteStream objects.

```
CFTypeID CFWriteStreamGetTypeID (
   void
);
```

Return Value
The type identifier for the CFWriteStream opaque type.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
`CFStream.h`

CFWriteStreamOpen

Opens a stream for writing.

```
Boolean CFWriteStreamOpen (
   CFWriteStreamRef 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 `CFWriteStreamScheduleWithRunLoop` (page 12) and wait for the stream's client (set with `CFWriteStreamSetClient` (page 13)) to be notified or you can poll the stream using `CFWriteStreamGetStatus` (page 11), waiting for a status of `kCFStreamStatusOpen` or `kCFStreamStatusError`.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
`CFStream.h`

CFWriteStreamScheduleWithRunLoop

Schedules a stream into a run loop.

```
void CFWriteStreamScheduleWithRunLoop (
   CFWriteStreamRef stream,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

stream

The stream to schedule.

runLoop
The run loop in which to schedule *stream*.

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

Discussion
After scheduling *stream* into a run loop, its client (set with `CFWriteStreamSetClient` (page 13)) is notified when various events happen with the stream, such as when it finishes opening, when it can accept new bytes, and when an error occurs. A stream can be scheduled into multiple run loops and run loop modes. Use `CFWriteStreamUnscheduleFromRunLoop` (page 15) to later remove *stream* from the run loop.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
`CFStream.h`

CFWriteStreamSetClient

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

```
Boolean CFWriteStreamSetClient (
   CFWriteStreamRef stream,
   CFOptionFlags streamEvents,
   CFWriteStreamClientCallBack 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 call 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, `false` otherwise.

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 `CFWriteStreamScheduleWithRunLoop` (page 12) 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 writing to memory) and don't benefit from asynchronous notification.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
`CFStream.h`

CFWriteStreamSetProperty

Sets the value of a property for a stream.

```
Boolean CFWriteStreamSetProperty (
   CFWriteStreamRef stream,
   CFStringRef propertyName,
   CFTypeRef 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 "Stream Properties."

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, `false` otherwise.

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 `CFWriteStreamCopyProperty` (page 8), although some properties are write-only.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
CFFTPSample
CocoaEcho
CocoaHTTPServer
CocoaSOAP

Declared In
`CFStream.h`

CFWriteStreamUnscheduleFromRunLoop

Removes a stream from a particular run loop.

```
void CFWriteStreamUnscheduleFromRunLoop (
   CFWriteStreamRef stream,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

stream
The stream to remove.

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
CFFTPSample

Declared In
`CFStream.h`

CFWriteStreamWrite

Writes data to a writable stream.

```
CFIndex CFWriteStreamWrite (
   CFWriteStreamRef stream,
   const UInt8 *buffer,
   CFIndex bufferLength
);
```

Parameters

stream
The stream to which to write.

buffer
The buffer holding the data to write.

bufferLength
The number of bytes from *buffer* to write.

Return Value
The number of bytes successfully written, `0` if the stream has been filled to capacity (for fixed-length streams), 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. If the stream is not full, this call blocks until at least one byte is written; it does not block until all the bytes in *buffer* is written. To avoid blocking, call this function only if `CFWriteStreamCanAcceptBytes` (page 7) returns `true` or after the stream's client (set with `CFWriteStreamSetClient` (page 13)) is notified of a `kCFStreamEventCanAcceptBytes` event.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
CFFTPSample

Declared In
`CFStream.h`

Callbacks

CFWriteStreamClientCallBack

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

```
typedef void (*CFWriteStreamClientCallBack) (
    CFWriteStreamRef stream,
    CFStreamEventType eventType,
    void *clientCallBackInfo
);
```

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

```
void MyCallBack (
    CFWriteStreamRef 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 "Stream Events.".

clientCallBackInfo
The `info` member of the `CFStreamClientContext` 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 `CFWriteStreamSetClient` (page 13).

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CFStream.h`

Data Types

CFWriteStreamRef

A reference to a writable stream object.

```
typedef struct __CFWriteStream *CFWriteStreamRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`CFStream.h`

Document Revision History

This table describes the changes to *CFWriteStream Reference*.

Date	Notes
2007-05-03	Updated to include API introduced in Mac OS X v10.5.
2006-01-10	Corrected typographical errors.
2005-11-09	Removed reference to retired document.
2005-03-03	TBD
2003-01-01	First version of this document.

Document Revision History

Index

C

CFWriteStreamCanAcceptBytes function 7
CFWriteStreamClientCallBack callback 16
CFWriteStreamClose function 7
CFWriteStreamCopyError function 7
CFWriteStreamCopyProperty function 8
CFWriteStreamCreateWithAllocatedBuffers function 8
CFWriteStreamCreateWithBuffer function 9
CFWriteStreamCreateWithFile function 10
CFWriteStreamGetError function 10
CFWriteStreamGetStatus function 11
CFWriteStreamGetTypeID function 11
CFWriteStreamOpen function 12
CFWriteStreamRef data type 17
CFWriteStreamScheduleWithRunLoop function 12
CFWriteStreamSetClient function 13
CFWriteStreamSetProperty function 14
CFWriteStreamUnscheduleFromRunLoop function 15
CFWriteStreamWrite function 15