NSStream Class Reference

Cocoa > Data Management

2008-10-15

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

Simultaneously published in the United States and Canada.

Contents

NSStream Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.3 and later.
Companion guide	Stream Programming Guide for Cocoa
Declared in	NSStream.h
Related sample code	CocoaEcho CocoaHTTPServer CocoaSOAP

Overview

`NSStream` is an abstract class for objects representing streams. Its interface is common to all Cocoa stream classes, including its concrete subclasses `NSInputStream` and `NSOutputStream`.

`NSStream` objects provide an easy way to read and write data to and from a variety of media in a device-independent way. You can create stream objects for data located in memory, in a file, or on a network (using sockets), and you can use stream objects without loading all of the data into memory at once.

By default, `NSStream` instances that are not file-based are non-seekable, one-way streams (although custom seekable subclasses are possible). Once the data has been provided or consumed, the data cannot be retrieved from the stream.

Subclassing Notes

`NSStream` is an abstract class, incapable of instantiation and intended to be subclassed. It publishes a programmatic interface that all subclasses must adopt and provide implementations for. The two Apple-provided concrete subclasses of `NSStream`, `NSInputStream` and `NSOutputStream`, are suitable for most purposes. However, there might be situations when you want a peer subclass to `NSInputStream` and `NSOutputStream`. For example, you might want a class that implements a full-duplex (two-way) stream, or a class whose instances are capable of seeking through a stream.

Methods to Override

All subclasses must fully implement the following methods, which are presented in functional pairs:

- `open` (page 9) and `close` (page 8)

  Implement `open` to open the stream for reading or writing and make the stream available to the client directly or, if the stream object is scheduled on a run loop, to the delegate. Implement `close` to close the stream and remove the stream object from the run loop, if necessary. A closed stream should still be able to accept new properties and report its current properties. Once a stream is closed, it cannot be reopened.

- `delegate` (page 8) and `setDelegate:` (page 11)

  Return and set the delegate. By a default, a stream object must be its own delegate; so a `setDelegate:` message with an argument of `nil` should restore this delegate. Do not retain the delegate to prevent retain cycles.

  To learn about delegates and delegation, read "Delegates and Data Sources" in *Cocoa Fundamentals Guide*.

- `scheduleInRunLoop:forMode:` (page 10) and `removeFromRunLoop:forMode:` (page 10)

  Implement `scheduleInRunLoop:forMode:` to schedule the stream object on the specified run loop for the specified mode. Implement `removeFromRunLoop:forMode:` to remove the object from the run loop. See the documentation of the `NSRunLoop` class for details. Once the stream object for an open stream is scheduled on a run loop, it is the responsibility of the subclass as it processes stream data to send `stream:handleEvent:` (page 12) messages to its delegate.

- `propertyForKey:` (page 9) and `setProperty:forKey:` (page 11)

  Implement these methods to return and set, respectively, the property value for the specified key. You may add custom properties, but be sure to handle all properties defined by `NSStream` as well.

- `streamStatus` (page 12) and `streamError` (page 12)

  Implement `streamStatus` to return the current status of the stream as a `NSStreamStatus` constant; you may define new `NSStreamStatus` constants, but be sure to handle the `NSStream`-defined constants properly. Implement`streamError` to return an `NSError` object representing the current error. You might decide to return a custom `NSError` object that can provide complete and localized information about the error.

Tasks

Creating Streams

`+ getStreamsToHost:port:inputStream:outputStream:` (page 7)
Creates and returns by reference an `NSInputStream` object and `NSOutputStream` object for a socket connection with a given host on a given port.

Configuring Streams

- `propertyForKey:` (page 9)
  Returns the receiver's property for a given key.
- `setProperty:forKey:` (page 11)
  Attempts to set the value of a given property of the receiver and returns a Boolean value that indicates whether the value is accepted by the receiver.
- `delegate` (page 8)
  Returns the receiver's delegate.
- `setDelegate:` (page 11)
  Sets the receiver's delegate.

Using Streams

- `open` (page 9)
  Opens the receiving stream.
- `close` (page 8)
  Closes the receiver.
- `stream:handleEvent:` (page 12) *delegate method*
  The delegate receives this message when a given event has occurred on a given stream.

Managing Run Loops

- `scheduleInRunLoop:forMode:` (page 10)
  Schedules the receiver on a given run loop in a given mode.
- `removeFromRunLoop:forMode:` (page 10)
  Removes the receiver from a given run loop running in a given mode.

Getting Stream Information

- `streamStatus` (page 12)
  Returns the receiver's status.
- `streamError` (page 12)
  Returns an `NSError` object representing the stream error.

Class Methods

getStreamsToHost:port:inputStream:outputStream:

Creates and returns by reference an `NSInputStream` object and `NSOutputStream` object for a socket connection with a given host on a given port.

```
+ (void)getStreamsToHost:(NSHost *)host port:(NSInteger)port
    inputStream:(NSInputStream **)inputStream outputStream:(NSOutputStream
    **)outputStream
```

Parameters

host

The host to which to connect.

port

The port to connect to on *host*.

inputStream

Upon return, contains the input stream. If `nil` is passed, the stream object is not created.

outputStream

Upon return, contains the output stream. If `nil` is passed, the stream object is not created.

Discussion
If neither *port* nor *host* is properly specified, no socket connection is made.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSStream.h`

Instance Methods

close

Closes the receiver.

```
- (void)close
```

Discussion
Closing the stream terminates the flow of bytes and releases system resources that were reserved for the stream when it was opened. If the stream has been scheduled on a run loop, closing the stream implicitly removes the stream from the run loop. A stream that is closed can still be queried for its properties.

Availability
Available in Mac OS X v10.3 and later.

See Also
- open (page 9)

Declared In
`NSStream.h`

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Discussion
By default, a stream is its own delegate, and subclasses of `NSInputStream` and `NSOutputStream` must maintain this contract.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setDelegate:` (page 11)

Declared In
`NSStream.h`

open

Opens the receiving stream.

```
- (void)open
```

Discussion
A stream must be created before it can be opened. Once opened, a stream cannot be closed and reopened.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- close` (page 8)

Declared In
`NSStream.h`

propertyForKey:

Returns the receiver's property for a given key.

```
- (id)propertyForKey:(NSString *)key
```

Parameters

key

The key for one of the receiver's properties. See "Constants" (page 13) for a description of the available property-key constants and associated values.

Return Value
The receiver's property for the key *key*.

Availability
Available in Mac OS X v10.3 and later.

See Also
`- setProperty:forKey:` (page 11)

Declared In
NSStream.h

removeFromRunLoop:forMode:

Removes the receiver from a given run loop running in a given mode.

```
- (void)removeFromRunLoop:(NSRunLoop *)aRunLoop forMode:(NSString *)mode
```

Parameters

aRunLoop

The run loop on which the receiver was scheduled.

mode

The mode for the run loop.

Availability
Available in Mac OS X v10.3 and later.

See Also
- scheduleInRunLoop:forMode: (page 10)

Declared In
NSStream.h

scheduleInRunLoop:forMode:

Schedules the receiver on a given run loop in a given mode.

```
- (void)scheduleInRunLoop:(NSRunLoop *)aRunLoop forMode:(NSString *)mode
```

Parameters

aRunLoop

The run loop on which to schedule the receiver.

mode

The mode for the run loop.

Discussion
Unless the client is polling the stream, it is responsible for ensuring that the stream is scheduled on at least one run loop and that at least one of the run loops on which the stream is scheduled is being run.

Availability
Available in Mac OS X v10.3 and later.

See Also
- removeFromRunLoop:forMode: (page 10)

Declared In
NSStream.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate

The delegate for the receiver.

Discussion

By default, a stream is its own delegate, and subclasses of `NSInputStream` and `NSOutputStream` must maintain this contract. If you override this method in a subclass, passing `nil` must restore the receiver as its own delegate. Delegates are not retained.

To learn about delegates and delegation, read "Delegates and Data Sources" in *Cocoa Fundamentals Guide*.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- delegate` (page 8)

Declared In

`NSStream.h`

setProperty:forKey:

Attempts to set the value of a given property of the receiver and returns a Boolean value that indicates whether the value is accepted by the receiver.

```
- (BOOL)setProperty:(id)property forKey:(NSString *)key
```

Parameters

property

The value for *key*.

key

The key for one of the receiver's properties. See "Constants" (page 13) for a description of the available property-key constants and expected values.

Return Value

`YES` if the value is accepted by the receiver, otherwise `NO`.

Availability

Available in Mac OS X v10.3 and later.

See Also

`- propertyForKey:` (page 9)

Declared In

`NSStream.h`

streamError

Returns an `NSError` object representing the stream error.

```
- (NSError *)streamError
```

Return Value
An `NSError` object representing the stream error, or `nil` if no error has been encountered.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSStream.h`

streamStatus

Returns the receiver's status.

```
- (NSStreamStatus)streamStatus
```

Return Value
The receiver's status.

Discussion
See "Constants" (page 13) for a description of the available `NSStreamStatus` constants.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSStream.h`

Delegate Methods

stream:handleEvent:

The delegate receives this message when a given event has occurred on a given stream.

```
- (void)stream:(NSStream *)theStream handleEvent:(NSStreamEvent)streamEvent
```

Parameters

`theStream`
The stream on which *`streamEvent`* occurred.

`streamEvent`
The stream event that occurred,

Discussion
The delegate receives this message only if *`theStream`* is scheduled on a run loop. The message is sent on the stream object's thread. The delegate should examine *`streamEvent`* to determine the appropriate action it should take.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSStream.h`

Constants

NSStreamStatus

The type declared for the constants listed in "Stream Status Constants" (page 13).

```
typedef NSUInteger NSStreamStatus;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSStream.h`

Stream Status Constants

These constants indicate the current status of a stream. They are returned by `streamStatus` (page 12).

```
typedef enum {
    NSStreamStatusNotOpen = 0,
    NSStreamStatusOpening = 1,
    NSStreamStatusOpen = 2,
    NSStreamStatusReading = 3,
    NSStreamStatusWriting = 4,
    NSStreamStatusAtEnd = 5,
    NSStreamStatusClosed = 6,
    NSStreamStatusError = 7
};
```

Constants

`NSStreamStatusNotOpen`

The stream is not open for reading or writing. This status is returned before the underlying call to open a stream but after it's been created.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusOpening`

The stream is in the process of being opened for reading or for writing. For network streams, this status might include the time after the stream was opened, but while network DNS resolution is happening.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusOpen`
The stream is open, but no reading or writing is occurring.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusReading`
Data is being read from the stream. This status would be returned if code on another thread were to call `streamStatus` (page 12) on the stream while a `read:maxLength:` call (`NSInputStream`) was in progress.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusWriting`
Data is being written to the stream. This status would be returned if code on another thread were to call `streamStatus` (page 12) on the stream while a `write:maxLength:` call (`NSOutputStream`) was in progress.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusAtEnd`
There is no more data to read, or no more data can be written to the stream. When this status is returned, the stream is in a "non-blocking" mode and no data are available.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusClosed`
The stream is closed (`close` (page 8) has been called on it).

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamStatusError`
The remote end of the connection can't be contacted, or the connection has been severed for some other reason.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

Declared In
`NSStream.h`

NSStreamEvent

The type declared for the constants listed in "Stream Event Constants" (page 15).

```
typedef NSUInteger NSStreamEvent;
```

Availability
Available in Mac OS X v10.3 and later.

Declared In
`NSStream.h`

Stream Event Constants

One or more of these constants may be sent to the delegate as a bit field in the second parameter of `stream:handleEvent:` (page 12).

```
typedef enum {
    NSStreamEventNone = 0,
    NSStreamEventOpenCompleted = 1 << 0,
    NSStreamEventHasBytesAvailable = 1 << 1,
    NSStreamEventHasSpaceAvailable = 1 << 2,
    NSStreamEventErrorOccurred = 1 << 3,
    NSStreamEventEndEncountered = 1 << 4
};
```

Constants

`NSStreamEventNone`

No event has occurred.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamEventOpenCompleted`

The open has completed successfully.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamEventHasBytesAvailable`

The stream has bytes to be read.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamEventHasSpaceAvailable`

The stream can accept bytes for writing.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamEventErrorOccurred`

An error has occurred on the stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamEventEndEncountered`

The end of the stream has been reached.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

Declared In

`NSStream.h`

NSStream Property Keys

`NSStream` defines these string constants as keys for accessing stream properties using `propertyForKey:` (page 9) and setting properties with `setProperty:forKey:` (page 11):

```
extern NSString * const NSStreamSocketSecurityLevelKey  ;
extern NSString * const NSStreamSocketSecurityLevelNone ;
extern NSString * const NSStreamSocketSecurityLevelSSLv2 ;
extern NSString * const NSStreamSocketSecurityLevelSSLv3 ;
extern NSString * const NSStreamSocketSecurityLevelTLSv1 ;
extern NSString * const NSStreamSocketSecurityLevelNegotiatedSSL;
extern NSString * const NSStreamSOCKSProxyConfigurationKey  ;
extern NSString * const NSStreamSOCKSProxyHostKey  ;
extern NSString * const NSStreamSOCKSProxyPortKey  ;
extern NSString * const NSStreamSOCKSProxyVersionKey  ;
extern NSString * const NSStreamSOCKSProxyUserKey  ;
extern NSString * const NSStreamSOCKSProxyPasswordKey ;
extern NSString * const NSStreamSOCKSProxyVersion4  ;
extern NSString * const NSStreamSOCKSProxyVersion5  ;
extern NSString * const NSStreamDataWrittenToMemoryStreamKey ;
extern NSString * const NSStreamFileCurrentOffsetKey  ;
```

Constants

`NSStreamSocketSecurityLevelKey`

The security level of the target stream. May be one of the following values: `NSStreamSocketSecurityLevelNone`, `NSStreamSocketSecurityLevelSSLv2`, `NSStreamSocketSecurityLevelSSLv3`, `NSStreamSocketSecurityLevelTLSv1`, or `NSStreamSocketSecurityLevelNegotiatedSSL`.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamSOCKSProxyConfigurationKey`

Value is an `NSDictionary` object containing SOCKS proxy configuration information.

The dictionary returned from the System Configuration framework for SOCKS proxies usually suffices.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamDataWrittenToMemoryStreamKey`

Value is an `NSData` instance containing the data written to a memory stream.

Use this property when you have an output-stream object instantiated to collect written data in memory. The value of this property is read-only.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamFileCurrentOffsetKey`

Value is an `NSNumber` object containing the current absolute offset of the stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

Declared In

`NSStream.h`

NSStream Error Domains

`NSStream` defines these string constants to represent error domains that can be returned by `streamError` (page 12):

```
extern NSString * const NSStreamSocketSSLErrorDomain  ;
extern NSString * const NSStreamSOCKSErrorDomain  ;
```

Constants

`NSStreamSocketSSLErrorDomain`

The error domain used by `NSError` when reporting SSL errors.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamSOCKSErrorDomain`

The error domain used by `NSError` when reporting SOCKS errors.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

Declared In

`NSStream.h`

Secure-Socket Layer (SSL) Security Level

`NSStream` defines these string constants for specifying the secure-socket layer (SSL) security level.

```
NSString * const NSStreamSocketSecurityLevelNone;
NSString * const NSStreamSocketSecurityLevelSSLv2;
NSString * const NSStreamSocketSecurityLevelSSLv3;
NSString * const NSStreamSocketSecurityLevelTLSv1;
NSString * const NSStreamSocketSecurityLevelNegotiatedSSL
```

Constants

`NSStreamSocketSecurityLevelNone`

Specifies that no security level be set for a socket stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamSocketSecurityLevelSSLv2`

Specifies that SSL version 2 be set as the security protocol for a socket stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamSocketSecurityLevelSSLv3`

Specifies that SSL version 3 be set as the security protocol for a socket stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamSocketSecurityLevelTLSv1`

Specifies that TLS version 1 be set as the security protocol for a socket stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

`NSStreamSocketSecurityLevelNegotiatedSSL`

Specifies that the highest level security protocol that can be negotiated be set as the security protocol for a socket stream.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

Discussion
You access and set these values using the `NSStreamSocketSecurityLevelKey` property key.

Declared In
`NSStream.h`

SOCKS Proxy Configuration Values

`NSStream` defines these string constants for use as keys to specify SOCKS proxy configuration values in an `NSDictionary` object.

```
NSString * const NSStreamSOCKSProxyHostKey;
NSString * const NSStreamSOCKSProxyPortKey;
NSString * const NSStreamSOCKSProxyVersionKey;
NSString * const NSStreamSOCKSProxyUserKey;
NSString * const NSStreamSOCKSProxyPasswordKey;
NSString * const NSStreamSOCKSProxyVersion4;
NSString * const NSStreamSOCKSProxyVersion5
```

Constants

`NSStreamSOCKSProxyHostKey`
Value is an `NSString` object that represents the SOCKS proxy host.
Available in Mac OS X v10.3 and later.
Declared in `NSStream.h`.

`NSStreamSOCKSProxyPortKey`
Value is an `NSNumber` object containing an integer that represents the port on which the proxy listens.
Available in Mac OS X v10.3 and later.
Declared in `NSStream.h`.

`NSStreamSOCKSProxyVersionKey`
Value is either `NSStreamSOCKSProxyVersion4` or `NSStreamSOCKSProxyVersion5`.
If this key is not present, `NSStreamSOCKSProxyVersion5` is used by default.
Available in Mac OS X v10.3 and later.
Declared in `NSStream.h`.

`NSStreamSOCKSProxyUserKey`
Value is an `NSString` object containing the user's name.
Available in Mac OS X v10.3 and later.
Declared in `NSStream.h`.

`NSStreamSOCKSProxyPasswordKey`
Value is an `NSString` object containing the user's password.
Available in Mac OS X v10.3 and later.
Declared in `NSStream.h`.

`NSStreamSOCKSProxyVersion4`
Possible value for `NSStreamSOCKSProxyVersionKey`.
Available in Mac OS X v10.3 and later.
Declared in `NSStream.h`.

`NSStreamSOCKSProxyVersion5`

Possible value for `NSStreamSOCKSProxyVersionKey`.

Available in Mac OS X v10.3 and later.

Declared in `NSStream.h`.

Discussion

You set the dictionary object as the current SOCKS proxy configuration using the `NSStreamSOCKSProxyConfigurationKey` key

Declared In

`NSStream.h`

Document Revision History

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

Date	Notes
2008-10-15	Revised descriptions of NSStreamStatus constants.
2007-07-23	Added descriptions of NSStreamStatus and NSStreamEvent types and made formatting changes.
2006-05-23	Updated related conceptual document and added definition of delegate; corrected constant declarations.
	First publication of this content as a separate document.

Document Revision History

Index