
CFStream Socket Additions

[Core Foundation](#) > [Networking](#)



2008-07-08



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

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

CFStream Socket Additions 5

Overview	5
Functions by Task	5
Creating Socket Pairs	5
Setting the Security Protocol	5
Obtaining Errors	5
Functions	6
CFReadStreamPairSetSecurityProtocol	6
CFReadStreamSOCKSError	6
CFReadStreamSOCKSErrorSubdomain	7
CFStreamCreatePairWithSocketToCFHost	8
CFStreamCreatePairWithSocketToNetService	8
Constants	9
CFStream Property Keys	9
CFStream Property SSL Settings Constants	11
CFStream Socket Security Protocol Constants	13
CFStream Socket Security Level Constants	14
CFStream SOCKS Proxy Key Constants	15
Error Domains	16
Error Subdomains	17
CFStream Errors	18

Document Revision History 21

Index 23

CFStream Socket Additions

Derived From:	CFType
Framework:	CoreServices
Companion guide	CFNetwork Programming Guide
Declared in	CFReadStream.h CFWriteStream.h

Overview

This document describes the `CFStream` functions for working with sockets. It is part of the `CFReadStream` API.

Functions by Task

Creating Socket Pairs

[CFStreamCreatePairWithSocketToCFHost](#) (page 8)

Creates readable and writable streams connected to a given `CFHost` object.

[CFStreamCreatePairWithSocketToNetService](#) (page 8)

Creates a pair of streams for a `CFNetService`.

Setting the Security Protocol

[CFReadStreamPairSetSecurityProtocol](#) (page 6)

This function sets the security protocol for the specified pair of socket streams. (**Deprecated.** Use `CFReadStreamSetProperty` and `CFWriteStreamSetProperty` in conjunction with the security constants defined in `CFReadStream`.)

Obtaining Errors

[CFReadStreamSOCKSError](#) (page 6)

This function gets error codes in the `kCFStreamErrorDomainSOCKS` domain from the `CFStreamError` returned by a stream operation.

[CFReadStreamSOCKSErrorSubdomain](#) (page 7)

Gets the error subdomain associated with errors in the `kCFStreamErrorDomainSOCKS` domain from the `CFStreamError` returned by a stream operation.

Functions

CFReadStreamPairSetSecurityProtocol

This function sets the security protocol for the specified pair of socket streams. (**Deprecated.** Use `CFReadStreamSetProperty` and `CFWriteStreamSetProperty` in conjunction with the security constants defined in `CFReadStream`.)

```
Boolean CFReadStreamPairSetSecurityProtocol (
    CFReadStreamRef socketReadStream,
    CFWriteStreamRef socketWriteStream,
    CFReadStreamSecurityProtocol securityProtocol
);
```

Parameters

socketReadStream

The read stream.

socketWriteStream

The write stream.

securityProtocol

The security protocol to be set. See [CFStream Socket Security Protocol Constants](#) (page 13) for possible values.

function result

TRUE if specified security protocol was set; otherwise, FALSE.

Discussion

Call this function before you call `CFReadStreamOpen` to open the read stream or `CFWriteStreamOpen` to open the write stream.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.2.

Declared In

`CFReadStream.h`

CFReadStreamSOCKSError

This function gets error codes in the `kCFStreamErrorDomainSOCKS` domain from the `CFStreamError` returned by a stream operation.

```
SInt32 CFReadStreamSOCKSError(CFStreamError* error);
```

Parameters

error

The error value to decode.

Discussion

Error codes in the `kCFStreamErrorDomainSOCKS` domain can come from multiple parts of the protocol stack, many of which define their own error values as part of outside specifications such as the HTTP specification.

To avoid confusion from conflicting error numbers, error codes in the `kCFStreamErrorDomainSOCKS` domain contain two parts: a subdomain, which tells which part of the protocol stack generated the error, and the error code itself.

Calling [CFReadStreamSOCKSError](#) (page 6) returns the error code itself, which must be interpreted in the context of the result of a call to [CFReadStreamSOCKSErrorSubdomain](#) (page 7). Possible return values (beyond subdomain-specific values such as client versions and HTTP error codes) are listed in [“CFStream Errors”](#) (page 18).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`CFReadStream.h`

CFReadStreamSOCKSErrorSubdomain

Gets the error subdomain associated with errors in the `kCFStreamErrorDomainSOCKS` domain from the `CFStreamError` returned by a stream operation.

```
SInt32 CFReadStreamSOCKSErrorSubdomain(CFStreamError* error);
```

Parameters

error

The error value to decode.

Discussion

Error codes in the `kCFStreamErrorDomainSOCKS` domain can come from multiple parts of the protocol stack, many of which define their own error values as part of outside specifications such as the HTTP specification.

To avoid confusion from conflicting error numbers, error codes in the `kCFStreamErrorDomainSOCKS` domain contain two parts: a subdomain, which tells which part of the protocol stack generated the error, and the error code itself.

Calling [CFReadStreamSOCKSErrorSubdomain](#) (page 7) returns an identifier that tells which layer of the protocol stack produced the error. The possible values are listed in [“Error Subdomains”](#) (page 17). With this information, you can interpret the error codes returned by [CFReadStreamSOCKSError](#) (page 6).

Availability

Available in Mac OS X v10.2 and later.

Declared In

CFReadStream.h

CFStreamCreatePairWithSocketToCFHostCreates readable and writable streams connected to a given `CFHost` object.

```
void CFStreamCreatePairWithSocketToCFHost (
    CFAllocatorRef alloc,
    CFHostRef host,
    SInt32 port,
    CFReadStreamRef *readStream,
    CFWriteStreamRef *writeStream
);
```

Parameters*alloc*

The allocator to use to allocate memory for the `CFReadStream` and `CFWriteStream` objects. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

host

A `CFHost` object to which the streams are connected. If unresolved, the host will be resolved prior to connecting.

port

The TCP port number to which the socket streams should connect.

readStream

Upon return, contains a `CFReadStream` object connected to the host *host* on port *port*, or `NULL` if there is a failure during creation. If you pass `NULL`, the function will not create a readable stream. Ownership follows the Create Rule.

writeStream

Upon return, contains a `CFWriteStream` object connected to the host *host* on port *port*, or `NULL` if there is a failure during creation. If you pass `NULL`, the function will not create a writable stream. Ownership follows the Create Rule.

Discussion

The streams do not open a connection to the specified host until one of the streams is opened.

Most properties are shared by both streams. Setting the property for one stream automatically sets the property for the other.

Availability

Available in Mac OS X v10.3 and later.

Declared In

CFReadStream.h

CFStreamCreatePairWithSocketToNetServiceCreates a pair of streams for a `CFNetService`.

```
void CFStreamCreatePairWithSocketToNetService (
    CFAllocatorRef alloc,
    CFNetServiceRef service,
    CFReadStreamRef *readStream,
    CFWriteStreamRef *writeStream
);
```

Parameters*alloc*

The allocator to use to allocate memory for the `CFReadStream` and `CFWriteStream` objects. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

service

Reference to the `CFNetService` to which the streams are to be connected. If the service is not resolved, the service will be resolved before the streams are connected.

readstream

Upon return, contains a `CFReadStream` object connected to the service specified by *service*, or `NULL` if there is a failure during creation. If you pass `NULL`, the function will not create a readable stream. Ownership follows the Create Rule.

writestream

Upon return, contains a `CFWriteStream` object connected to the service specified by *service*, or `NULL` if there is a failure during creation. If you pass `NULL`, the function will not create a writable stream. Ownership follows the Create Rule.

Discussion

Read and write operations on sockets can block. To prevent blocking, you can call `CFReadStreamSetClient` and `CFWriteStreamSetClient` to register to receive stream-related event notifications. Then call `CFReadStreamScheduleWithRunLoop` and `CFWriteStreamScheduleWithRunLoop` to schedule the stream on a run loop for receiving stream-related event notifications. Then call `CFReadStreamOpen` and `CFWriteStreamOpen` to open each stream.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`CFSocketStream.h`

Constants

CFStream Property Keys

Constants for `CFStream` property keys

```

const CFStringRef kCFStreamPropertyShouldCloseNativeSocket;
const CFStringRef kCFStreamPropertySocketSecurityLevel;
const CFStringRef kCFStreamPropertySOCKSProxy;
const CFStringRef kCFStreamPropertySSLPeerCertificates;
const CFStringRef kCFStreamPropertySSLSettings;
const CFStringRef kCFStreamPropertyProxyLocalByPass;
extern const CFStringRef kCFStreamPropertySocketRemoteHost;
extern const CFStringRef kCFStreamPropertySocketRemoteNetService;

```

Constants

`kCFStreamPropertyShouldCloseNativeSocket`

Should Close Native Socket property key.

If set to `kCFBooleanTrue`, the stream will close and release the underlying native socket when the stream is released. If set to `kCFBooleanFalse`, the stream will not close and release the underlying native socket when the stream is released. If a stream is created with a native socket, the default value of this property is `kCFBooleanFalse`. This property is only available for socket streams. It can be set by calling `CFReadStreamSetProperty` and `CFWriteStreamSetProperty`, and it can be copied by `CFReadStreamCopyProperty` and `CFWriteStreamCopyProperty`.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySocketNativeHandle`

Socket Native Handle property key.

Causes `CFReadStreamCopyProperty` or `CFWriteStreamCopyProperty` to return `CFData` object that contains the native handle for a socket stream. This property is only available for socket streams.

Available in Mac OS X v10.1 and later.

Declared in `CFStream.h`.

`kCFStreamPropertySocketSecurityLevel`

Socket Security Level property key.

See [CFStream Socket Security Level Constants](#) (page 14) for specific security level constants to use.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySSLPeerCertificates`

SSL Peer Certificates property key for copy operations, which return a `CFArray` object containing `SecCertificateRef` objects.

For more information, see `SSLGetPeerCertificates` in `Security/SecureTransport.h`.

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySOCKSProxy`

SOCKS proxy property key.

To set a `CFStream` object to use a SOCKS proxy, call `CFReadStreamSetProperty` or `CFWriteStreamSetProperty` with the property name set to `kCFStreamPropertySOCKSProxy` and its value set to a `CFDictionary` object having at minimum a `kCFStreamPropertySOCKSProxyHost` key and a `kCFStreamPropertySOCKSProxyPort` key. For information on these keys, see [CFStream SOCKS Proxy Key Constants](#) (page 15). `SystemConfiguration` returns a `CFDictionary` for SOCKS proxies that is usable without modification.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySSLSettings`

SSL Settings property key for set operations.

The key's value is a `CFDictionary` object containing security settings. For information on the dictionary's keys and values, see [CFStream Property SSL Settings Constants](#) (page 11). By default, there are no security settings.

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertyProxyLocalBypass`

Proxy Local Bypass property key.

The key's value is a `CFBoolean` object whose value indicates whether local hostnames should be subject to proxy handling.

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySocketRemoteHost`

The key's value is a `CFHostRef` for the remote host if it is known. If not, its value is `NULL`.

Available in Mac OS X version 10.3 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySocketRemoteNetService`

The key's value is a `CFNetServiceRef` for the remote network service if it is known. If not, its value is `NULL`.

Available in Mac OS X version 10.3 and later.

Declared in `CFSocketStream.h`.

Declared In

`CFNetwork/CFSocketStream.h`

CFStream Property SSL Settings Constants

Constants for use in a `CFDictionary` object that is the value of the `kCFStreamPropertySSLSettings` stream property key.

```
const CFStringRef kCFStreamSSLLevel;
const CFStringRef kCFStreamSSLAllowsExpiredCertificates;
const CFStringRef kCFStreamSSLAllowsExpiredRoots;
const CFStringRef kCFStreamSSLAllowsAnyRoot;
const CFStringRef kCFStreamSSLValidatesCertificateChain;
const CFStringRef kCFStreamSSLPeerName;
const CFStringRef kCFStreamSSLCertificates;
const CFStringRef kCFStreamSSLIsServer;
```

Constants

`kCFStreamSSLLevel`

Security property key whose value specifies the stream's security level.

By default, a stream's security level is `kCFStreamSocketSecurityLevelNegotiatedSSL`. For other possible values, see [CFStream Socket Security Level Constants](#) (page 14).

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLAllowsExpiredCertificates`

Security property key whose value indicates whether expired certificates are allowed.

By default, the value of this key is `kCFBooleanFalse` (expired certificates are not allowed).

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLAllowsExpiredRoots`

Security property whose value indicates whether expired root certificates are allowed.

By default, the value of this key is `kCFBooleanFalse` (expired root certificates are not allowed).

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLAllowsAnyRoot`

Security property key whose value indicates whether root certificates should be allowed.

By default, the value of this key is `kCFBooleanFalse` (root certificates are not allowed).

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLValidatesCertificateChain`

Security property key whose value indicates whether the certificate chain should be validated.

By default, the value of this key is `kCFBooleanTrue` (the certificate chain should be validated).

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLPeerName`

Security property key whose value overrides the name used for certificate verification.

By default, the host name that was used when the stream was created is used; if no host name was used, no peer name will be used. Set the value of this key to `kCFNull` to prevent name verification.

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLCertificates`

Security property key whose value is a `CFArray` of `SecCertificateRefs` except for the first element in the array, which is a `SecIdentityRef`.

For more information, see `SSLSetCertificate()` in `Security/SecureTransport.h`.

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSSLIsServer`

Security property key whose value indicates whether the connection is to act as a server in the SSL process.

By default, the value of this key is `kCFBooleanFalse` (the connection is not to act as a server). If the value of this key is `kCFBooleanTrue`, the `kCFStreamSSLCertificates` key must contain a valid value.

Available in Mac OS X v10.4 and later.

Declared in `CFSocketStream.h`.

Discussion

This enumeration defines the constants for keys in a `CFDictionary` object that is the value of the `kCFStreamPropertySSLSettings` key.

Declared In

CFNetwork/CFSocketStream.h

CFStream Socket Security Protocol Constants

Specifies constants for setting the security protocol for a socket stream.

```
typedef enum {
    kCFStreamSocketSecurityNone = 0,
    kCFStreamSocketSecuritySSLv2,
    kCFStreamSocketSecuritySSLv3,
    kCFStreamSocketSecuritySSLv23,
    kCFStreamSocketSecurityTLSv1
} CFStreamSocketSecurityProtocol;
```

Constants

kCFStreamSocketSecurityNone

Specifies that no security protocol be set for a socket stream. (**Deprecated.** Use kCFStreamSocketSecurityLevelNone.)

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.2.

Declared in CFSocketStream.h.

kCFStreamSocketSecuritySSLv2

Specifies that SSL version 2 be set as the security protocol for a socket stream. (**Deprecated.** Use kCFStreamSocketSecurityLevelSSLv2.)

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.2.

Declared in CFSocketStream.h.

kCFStreamSocketSecuritySSLv3

Specifies that SSL version 3 be set as the security protocol for a socket stream. (**Deprecated.** Use kCFStreamSocketSecurityLevelSSLv3.)

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.2.

Declared in CFSocketStream.h.

kCFStreamSocketSecuritySSLv23

Specifies that SSL version 3 be set as the security protocol for a socket stream pair. If that version is not available, specifies that SSL version 2 be set as the security protocol for a socket stream. (**Deprecated.** Use kCFStreamSocketSecurityLevelNegotiatedSSL.)

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.2.

Declared in CFSocketStream.h.

`kCFStreamSocketSecurityTLSv1`

Specifies that TLS version 1 be set as the security protocol for a socket stream. (**Deprecated.** Use `kCFStreamSocketSecurityLevelTLSv1`.)

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.2.

Declared in `CFSocketStream.h`.

Discussion

This enumeration defines constants for setting the security protocol for a socket stream pair when calling `CFSocketStreamPairSetSecurityProtocol` (page 6).

Special Considerations

This enumeration is deprecated in favor of the constants described in [CFStream Socket Security Level Constants](#) (page 14).

Declared In

`CFNetwork/CFSocketStream.h`

CFStream Socket Security Level Constants

Constants for setting the security level of a socket stream.

```
const CFStringRef kCFStreamSocketSecurityLevelNone;
const CFStringRef kCFStreamSocketSecurityLevelSSLv2;
const CFStringRef kCFStreamSocketSecurityLevelSSLv3;
const CFStringRef kCFStreamSocketSecurityLevelTLSv1;
const CFStringRef kCFStreamSocketSecurityLevelNegotiatedSSL;
```

Constants

`kCFStreamSocketSecurityLevelNone`

Specifies that no security level be set.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSocketSecurityLevelSSLv2`

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

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSocketSecurityLevelSSLv3`

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

If SSL version 3 is not available, specifies that SSL version 2 be set as the security protocol for a socket stream.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSocketSecurityLevelTLSv1`

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

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSocketSecurityLevelNegotiatedSSL`

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 v10.2 and later.

Declared in `CFSocketStream.h`.

Discussion

This enumeration defines the preferred constants for setting the security protocol for a socket stream pair when calling `CFReadStreamSetProperty` or `CFWriteStreamSetProperty`.

Declared In

`CFNetwork/CFSocketStream.h`

CFStream SOCKS Proxy Key Constants

Constants for SOCKS Proxy CFDictionary keys.

```
const CFStringRef kCFStreamPropertySOCKSProxyHost;  
const CFStringRef kCFStreamPropertySOCKSProxyPort;  
const CFStringRef kCFStreamPropertySOCKSVersion;  
const CFStringRef kCFStreamSocketSOCKSVersion4;  
const CFStringRef kCFStreamSocketSOCKSVersion5;  
const CFStringRef kCFStreamPropertySOCKSUser;  
const CFStringRef kCFStreamPropertySOCKSPassword;
```

Constants

`kCFStreamPropertySOCKSProxyHost`

Constant for the SOCKS proxy host key.

This key contains a `CFString` object that represents the SOCKS proxy host. Defined to match `kSCPropNetProxiesSOCKSProxy`.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySOCKSProxyPort`

Constant for the SOCKS proxy host port key.

This key contains a `CFNumberRef` object of type `kCFNumberSInt32Type` whose value represents the port on which the proxy listens.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySOCKSVersion`

Constant for the SOCKS version key.

Its value must be `kCFStreamSocketSOCKSVersion4` or `kCFStreamSocketSOCKSVersion5` to set SOCKS4 or SOCKS5, respectively. If this key is not present, SOCKS5 is used by default.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSocketSOCKSVersion4`

Constant used in the `kCFStreamSocketSOCKSVersion` key to specify SOCKS4 as the SOCKS version for the stream.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamSocketSOCKSVersion5`

Constant used in the `kCFStreamSOCKSVersion` key to specify SOCKS5 as the SOCKS version for the stream.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySOCKSUser`

Constant for the key required to set a user name.

The value is a `CFString` object containing the user's name.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

`kCFStreamPropertySOCKSPassword`

Constant for the key required to set a user's password.

The value is a `CFString` object containing the user's password.

Available in Mac OS X v10.2 and later.

Declared in `CFSocketStream.h`.

Discussion

When setting the stream's SOCKS Proxy property, the property's value is a `CFDictionary` object containing at minimum the `kCFStreamPropertySOCKSProxyHost` and `kCFStreamPropertySOCKSProxyPort` keys. The dictionary may also contain the other keys described in this section.

Error Domains

Error domains specific to `CFSocketStream` calls.

```
extern const int kCFStreamErrorDomainSOCKS;
extern const int kCFStreamErrorDomainSSL;
extern const CFIndex kCFStreamErrorDomainWinSock;
```

Constants

`kCFStreamErrorDomainSOCKS`

This domain returns error codes from the SOCKS layer. The errors are described in

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorDomainSSL`

This domain returns error codes associated with the SSL layer. For a list of error codes, see the header `SecureTransport.h` in `Security.framework`.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorDomainWinSock`

When running `CFNetwork` code on Windows, this domain returns error codes associated with the underlying TCP/IP stack. You should also note that non-networking errors such as `ENOMEM` are delivered through the POSIX domain. See the header `winsock2.h` for relevant error codes.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

Discussion

To determine the source of an error, examine the `userInfo` dictionary included in the `CFError` object returned by a function call or call `CFErrorGetDomain` and pass in the `CFError` object and the domain whose value you want to read.

Error Subdomains

Subdomains used to determine how to interpret an error in the `kCFStreamErrorDomainSOCKS` domain.

```
enum {
    kCFStreamErrorSOCKSSubDomainNone = 0,
    kCFStreamErrorSOCKSSubDomainVersionCode = 1,
    kCFStreamErrorSOCKS4SubDomainResponse = 2,
    kCFStreamErrorSOCKS5SubDomainUserPass = 3,
    kCFStreamErrorSOCKS5SubDomainMethod = 4,
    kCFStreamErrorSOCKS5SubDomainResponse = 5
};
```

Constants

`kCFStreamErrorSOCKSSubDomainNone`

The error code returned is a SOCKS error number.

Available in Mac OS X version 10.5 and later.

`kCFStreamErrorSOCKSSubDomainVersionCode`

The error returned contains the version of SOCKS that the server wishes to use.

Available in Mac OS X version 10.5 and later.

`kCFStreamErrorSOCKS4SubDomainResponse`

The error returned is the status code that the server returned after the last operation.

Available in Mac OS X version 10.5 and later.

`kCFStreamErrorSOCKS5SubDomainUserPass`

This subdomain returns error codes associated with the last authentication attempt.

Available in Mac OS X version 10.5 and later.

`kCFStreamErrorSOCKS5SubDomainMethod`

This subdomain returns the server's desired negotiation method.

Available in Mac OS X version 10.5 and later.

`kCFStreamErrorSOCKS5SubDomainResponse`

This subdomain returns the response code sent by the server when replying to a connection request.

Available in Mac OS X version 10.5 and later.

Discussion

Error codes in the `kCFStreamErrorDomainSOCKS` domain can come from multiple parts of the protocol stack, many of which define their own error values as part of outside specifications such as the HTTP specification.

To avoid confusion from conflicting error numbers, error codes in the `kCFStreamErrorDomainSOCKS` domain contain two parts: a subdomain, which tells which part of the protocol stack generated the error, and the error code itself.

Calling `CFSocketStreamSOCKSGetErrorSubdomain` (page 7) returns an identifier that tells which layer of the protocol stack produced the error. This list of constants contains the possible values that this function will return.

Calling `CFSocketStreamSOCKSError` (page 6) returns the actual error code that the subdomain describes.

CFStream Errors

Error codes returned by the `kCFStreamErrorDomainSOCKS` error domain.

```

/* kCFStreamErrorSOCKSSubDomainNone*/
enum {
    kCFStreamErrorSOCKS5BadResponseAddr = 1,
    kCFStreamErrorSOCKS5BadState = 2,
    kCFStreamErrorSOCKSUnknownClientVersion = 3
};

/* kCFStreamErrorSOCKS4SubDomainResponse*/
enum {
    kCFStreamErrorSOCKS4RequestFailed = 91,
    kCFStreamErrorSOCKS4IdentdFailed = 92,
    kCFStreamErrorSOCKS4IdConflict = 93
};

/* kCFStreamErrorSOCKS5SubDomainMethod*/
enum {
    kSOCKS5NoAcceptableMethod = 0xFF
};

```

Constants

`kCFStreamErrorSOCKS5BadResponseAddr`

The address returned is not of a known type. This error code is only valid for errors in the `kCFStreamErrorSOCKSSubDomainNone` subdomain.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorSOCKS5BadState`

The stream is not in a state that allows the requested operation. This error code is only valid for errors in the `kCFStreamErrorSOCKSSubDomainNone` subdomain..

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorSOCKSUnknownClientVersion`

The SOCKS server rejected access because it does not support connections with the requested SOCKS version. SOCKS client version. You can query the `kCFSOCKSVersionKey` key to find out what version the server requested. This error code is only valid for errors in the `kCFStreamErrorSOCKSSubDomainNone` subdomain.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorSOCKS4RequestFailed`

Request rejected by the server or request failed. This error is specific to SOCKS4. This error code is only valid for errors in the `kCFStreamErrorSOCKS4SubDomainResponse` subdomain.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorSOCKS4IdentdFailed`

Request rejected by the server because it could not connect to the `identd` daemon on the client. This error is specific to SOCKS4. This error code is only valid for errors in the `kCFStreamErrorSOCKS4SubDomainResponse` subdomain.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kCFStreamErrorSOCKS4IdConflict`

Request rejected by the server because the client program and the `identd` daemon reported different user IDs. This error is specific to SOCKS4. This error code is only valid for errors in the `kCFStreamErrorSOCKS4SubDomainResponse` subdomain.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

`kSOCKS5NoAcceptableMethod`

The client and server could not find a mutually agreeable authentication method. This error code is only valid for errors in the `kCFStreamErrorSOCKS5SubDomainMethod` subdomain.

Available in Mac OS X version 10.5 and later.

Declared in `CFSocketStream.h`.

Discussion

Error codes in the `kCFStreamErrorDomainSOCKS` domain can come from multiple parts of the protocol stack, many of which define their own error values as part of outside specifications such as the HTTP specification.

To avoid confusion from conflicting error numbers, error codes in the `kCFStreamErrorDomainSOCKS` domain contain two parts: a subdomain, which tells which part of the protocol stack generated the error, and the error code itself.

Calling [CFSocketStreamSOCKSGetErrorSubdomain](#) (page 7) returns an identifier that tells which layer of the protocol stack produced the error.

Calling [CFSocketStreamSOCKSGetError](#) (page 6) returns the actual error code that the subdomain describes. This list of constants contains the possible values that this function will return. They must be interpreted within the context of the relevant error subdomain.

Document Revision History

This table describes the changes to *CFStream Socket Additions*.

Date	Notes
2008-07-08	Added documentation for <code>kCFStreamErrorDomainWinSock</code> .
2006-07-24	Updated to describe replacements for deprecated functions.
2006-02-07	New document that describes the C API for using streams with sockets.

REVISION HISTORY

Document Revision History

Index

C

CFReadStreamPairSetSecurityProtocol **function**
(Deprecated in Mac OS X v10.2) [6](#)
CFReadStreamSOCKSGetError **function** [6](#)
CFReadStreamSOCKSGetErrorSubdomain **function**
[7](#)
CFStream Errors [18](#)
CFStream Property Keys [9](#)
CFStream Property SSL Settings Constants [11](#)
CFStream Socket Security Level Constants [14](#)
CFStream Socket Security Protocol Constants [13](#)
CFStream SOCKS Proxy Key Constants [15](#)
CFStreamCreatePairWithSocketToCFHost **function**
[8](#)
CFStreamCreatePairWithSocketToNetService
function [8](#)
CFWriteStreamScheduleWithRunLoop **function** [5,6](#)

E

Error Domains [16](#)
Error Subdomains [17](#)

F

functions
CFWriteStreamScheduleWithRunLoop [5,6](#)

K

kCFStreamErrorDomainSOCKS **constant** [16](#)
kCFStreamErrorDomainSSL **constant** [16](#)
kCFStreamErrorDomainWinSock **constant** [16](#)
kCFStreamErrorSOCKS4IdConflict **constant** [19](#)
kCFStreamErrorSOCKS4IdentdFailed **constant** [19](#)

kCFStreamErrorSOCKS4RequestFailed **constant** [19](#)
kCFStreamErrorSOCKS4SubDomainResponse **constant**
[17](#)
kCFStreamErrorSOCKS5BadResponseAddr **constant**
[18](#)
kCFStreamErrorSOCKS5BadState **constant** [18](#)
kCFStreamErrorSOCKS5SubDomainMethod **constant**
[17](#)
kCFStreamErrorSOCKS5SubDomainResponse **constant**
[17](#)
kCFStreamErrorSOCKS5SubDomainUserPass **constant**
[17](#)
kCFStreamErrorSOCKSSubDomainNone **constant** [17](#)
kCFStreamErrorSOCKSSubDomainVersionCode
constant [17](#)
kCFStreamErrorSOCKSUnknownClientVersion
constant [18](#)
kCFStreamPropertyProxyLocalBypass **constant** [11](#)
kCFStreamPropertyShouldCloseNativeSocket
constant [10](#)
kCFStreamPropertySocketNativeHandle **constant**
[10](#)
kCFStreamPropertySocketRemoteHost **constant** [11](#)
kCFStreamPropertySocketRemoteNetService
constant [11](#)
kCFStreamPropertySocketSecurityLevel **constant**
[10](#)
kCFStreamPropertySOCKSPassword **constant** [16](#)
kCFStreamPropertySOCKSProxy **constant** [10](#)
kCFStreamPropertySOCKSProxyHost **constant** [15](#)
kCFStreamPropertySOCKSProxyPort **constant** [15](#)
kCFStreamPropertySOCKSUser **constant** [16](#)
kCFStreamPropertySOCKSVersion **constant** [15](#)
kCFStreamPropertySSLPeerCertificates **constant**
[10](#)
kCFStreamPropertySSLSettings **constant** [11](#)
kCFStreamSocketSecurityLevelNegotiatedSSL
constant [15](#)
kCFStreamSocketSecurityLevelNone **constant** [14](#)
kCFStreamSocketSecurityLevelSSLv2 **constant** [14](#)
kCFStreamSocketSecurityLevelSSLv3 **constant** [14](#)
kCFStreamSocketSecurityLevelTLSv1 **constant** [14](#)

kCFStreamSocketSecurityNone **constant** (Deprecated in Mac OS X v10.2) [13](#)

kCFStreamSocketSecuritySSLv2 **constant** (Deprecated in Mac OS X v10.2) [13](#)

kCFStreamSocketSecuritySSLv23 **constant** (Deprecated in Mac OS X v10.2) [13](#)

kCFStreamSocketSecuritySSLv3 **constant** (Deprecated in Mac OS X v10.2) [13](#)

kCFStreamSocketSecurityTLSv1 **constant** (Deprecated in Mac OS X v10.2) [14](#)

kCFStreamSocketSOCKSVersion4 **constant** [15](#)

kCFStreamSocketSOCKSVersion5 **constant** [16](#)

kCFStreamSSLAllowsAnyRoot **constant** [12](#)

kCFStreamSSLAllowsExpiredCertificates **constant** [12](#)

kCFStreamSSLAllowsExpiredRoots **constant** [12](#)

kCFStreamSSLCertificates **constant** [12](#)

kCFStreamSSLIsServer **constant** [12](#)

kCFStreamSSLLevel **constant** [11](#)

kCFStreamSSLPeerName **constant** [12](#)

kCFStreamSSLValidatesCertificateChain **constant** [12](#)

kSOCKS5NoAcceptableMethod **constant** [19](#)