CFNetwork Error Codes Reference

Networking > Core Foundation

2008-10-15

Contents

CFNetwork Error Codes Reference

Framework:	CoreServices
Companion guide	CFNetwork Programming Guide
Declared in	CFNetworkErrors.h

Overview

Many functions in the CFNetwork API return error codes to indicate the cause of a failure. This document explains these error codes.

Constants

CFNetworkErrors Constants

This enumeration contains error codes returned under the error domain `kCFErrorDomainCFNetwork`.

```
enum CFNetworkErrors {
  kCFHostErrorHostNotFound      = 1,
  kCFHostErrorUnknown           = 2,

/* SOCKS errors */
  kCFSOCKSErrorUnknownClientVersion = 100,
  kCFSOCKSErrorUnsupportedServerVersion = 101,

/* SOCKS4-specific errors*/
  kCFSOCKS4ErrorRequestFailed   = 110,
  kCFSOCKS4ErrorIdentdFailed    = 111,
  kCFSOCKS4ErrorIdConflict      = 112,
  kCFSOCKS4ErrorUnknownStatusCode = 113,

/* SOCKS5-specific errors*/
  kCFSOCKS5ErrorBadState        = 120,
  kCFSOCKS5ErrorBadResponseAddr = 121,
  kCFSOCKS5ErrorBadCredentials  = 122,
  kCFSOCKS5ErrorUnsupportedNegotiationMethod = 123,
  kCFSOCKS5ErrorNoAcceptableMethod = 124,
/* Errors originating from CFNetServices*/
  kCFNetServiceErrorUnknown     = -72000L,
  kCFNetServiceErrorCollision   = -72001L,
  kCFNetServiceErrorNotFound    = -72002L,
  kCFNetServiceErrorInProgress  = -72003L,
  kCFNetServiceErrorBadArgument = -72004L,
  kCFNetServiceErrorCancel      = -72005L,
  kCFNetServiceErrorInvalid     = -72006L,
  kCFNetServiceErrorTimeout     = -72007L,
  kCFNetServiceErrorDNSServiceFailure = -73000L,

/* FTP errors */
  kCFFTPErrorUnexpectedStatusCode = 200,

/* HTTP errors*/
  kCFErrorHTTPAuthenticationTypeUnsupported = 300,
  kCFErrorHTTPBadCredentials    = 301,
  kCFErrorHTTPConnectionLost    = 302,
  kCFErrorHTTPParseFailure      = 303,
  kCFErrorHTTPRedirectionLoopDetected = 304,
  kCFErrorHTTPBadURL            = 305,
  kCFErrorHTTPProxyConnectionFailure = 306,
  kCFErrorHTTPBadProxyCredentials = 307,
  kCFErrorPACFileError          = 308
};
typedef enum CFNetworkErrors CFNetworkErrors;
```

Constants

`kCFHostErrorHostNotFound`

Indicates that the DNS lookup failed.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFHostErrorUnknown`

An unknown error occurred (a name server failure, for example). For additional information, you can query the `kCFGetAddrInfoFailureKey` key to obtain the value returned by `getaddrinfo(3)` and look up the value in `/usr/include/netdb.h`.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFSOCKSErrorUnknownClientVersion`

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.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFSOCKSErrorUnsupportedServerVersion`

The version of SOCKS requested by the server is not supported. You can query the `kCFSOCKSVersionKey` key to find out what version the server requested.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFSOCKS4ErrorRequestFailed`
: Request rejected by the server or request failed. This error is specific to SOCKS4.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS4ErrorIdentdFailed`
: Request rejected by the server because it could not connect to the `identd` daemon on the client. This error is specific to SOCKS4.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS4ErrorIdConflict`
: Request rejected by the server because the client program and the `identd` daemon reported different user IDs. This error is specific to SOCKS4.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS4ErrorUnknownStatusCode`
: The status code returned by the server is unknown. This error is specific to SOCKS4.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS5ErrorBadState`
: The stream is not in a state that allows the requested operation.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS5ErrorBadResponseAddr`
: The address type returned is not supported

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS5ErrorBadCredentials`
: The SOCKS server refused the client connection because of bad login credentials.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS5ErrorUnsupportedNegotiationMethod`
: The requested method is not supported. You can query the `kCFSOCKSNegotiationMethodKey` key to find out which method the server requested.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFSOCKS5ErrorNoAcceptableMethod`
: The client and server could not find a mutually agreeable authentication method.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFNetServiceErrorUnknown`
: An unknown error occurred.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorCollision`
: An attempt was made to use a name that is already in use.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorNotFound`
: Not used.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorInProgress`
: A new search could not be started because a search is already in progress.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorBadArgument`
: A required argument was not provided or was not valid.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorCancel`
: The search or service was cancelled.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorInvalid`
: Invalid data was passed to a CFNetServices function.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorTimeout`
: A search failed because it timed out.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.2 and later.

`kCFNetServiceErrorDNSServiceFailure`
: DNS service discovery returned an error. You can query the `kCFDNSServiceFailureKey` key to find out the error returned by DNS service discovery and look up the code in `/usr/include/dns_ds.h` or *DNS Service Discovery C Reference*.

  Declared in `CFNetworkErrors.h`.

  Available in Mac OS X v10.5 and later.

`kCFFTPErrorUnexpectedStatusCode`
: The server returned an unexpected status code.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPAuthenticationTypeUnsupported`
: The client and server could not agree on a supported authentication type.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPBadCredentials`
: The credentials provided for an authenticated connection were rejected by the server.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPConnectionLost`
: The connection to the server was dropped. This usually indicates a highly overloaded server.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPParseFailure`
: The HTTP server response could not be parsed.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPRedirectionLoopDetected`
: Too many HTTP redirects occurred before reaching a page that did not redirect the client to another page. This usually indicates a redirect loop.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPBadURL`
: The requested URL could not be retrieved.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPProxyConnectionFailure`
: A connection could not be established to the specified HTTP proxy.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.h`.

`kCFErrorHTTPBadProxyCredentials`
: The authentication credentials provided for logging into the proxy were rejected.

  Available in Mac OS X v10.5 and later.

  Declared in `CFNetworkErrors.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. These errors are all part of the `kCFErrorDomainCFNetwork` domain.

Availability
Available in Mac OS X version 10.5 and later.

Declared In
`CFNetwork/CFHost.h`

Property Keys

Keys for calls to property get/set functions such as `CFReadStreamSetProperty` and `CFReadStreamCopyProperty`.

```
extern const CFStringRef kCFGetAddrInfoFailureKey;
extern const CFStringRef kCFSOCKSStatusCodeKey;
extern const CFStringRef kCFSOCKSVersionKey;
extern const CFStringRef kCFSOCKSNegotiationMethodKey;
extern const CFStringRef kCFDNSServiceFailureKey;
extern const CFStringRef kCFFTPStatusCodeKey;
```

Constants

`kCFGetAddrInfoFailureKey`

Querying this key returns the last error code returned by `getaddrinfo(3)` in response to a DNS lookup. To interpret the results, look up the error code in `/usr/include/netdb.h`.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFSOCKSStatusCodeKey`

Querying this key returns the last status code sent by the SOCKS server in response to the previous operation.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFSOCKSVersionKey`

Querying this key returns the SOCKS version in use by the current connection.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFSOCKSNegotiationMethodKey`

Querying this key returns the negotiation method requested by the SOCKS server.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFDNSServiceFailureKey`

Querying this key returns the last error returned by the DNS resolver libraries in response to the previous operation. To interpret the results, look up the error codes in `/usr/include/dns_sd.h` or *DNS Service Discovery C Reference*.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

`kCFFTPStatusCodeKey`

Querying this key returns the last status code sent by the FTP server in response to the previous operation.

Available in Mac OS X v10.5 and later.

Declared in `CFNetworkErrors.h`.

Error Domains

High-level error domains.

```
extern const CFStringRef kCFErrorDomainCFNetwork;
extern const CFStringRef kCFErrorDomainWinSock;
```

Constants

`kCFErrorDomainCFNetwork`

Error domain that returns error codes specific to the CFNetwork stack.

Declared in `CFNetworkErrors.h`.

Available in Mac OS X version 10.5 and later.

`kCFErrorDomainWinSock`

Error domain that returns error codes specific to the underlying network layer when running CFNetwork code on Windows.

Declared in `CFNetworkErrors.h`.

Available in Mac OS X version 10.5 and later.

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.

Document Revision History

This table describes the changes to *CFNetwork Error Codes Reference*.

Date	Notes
2008-10-15	Corrected typos.
2008-07-07	First version.

Index