System Configuration Framework Reference

Networking & Internet

2009-07-30

Contents

Introduction

Companion guide	System Configuration Programming Guidelines
Declared in	SCNetworkReachability.h SystemConfiguration.h

This collection of documents describes the programming interfaces of the System Configuration framework. The System Configuration framework provides functions that determine the reachability of target hosts in both a synchronous and an asynchronous manner. It also provides error detection facilities.

Other References

SCNetworkReachability Reference

Framework:	SystemConfiguration
Declared in	SCNetworkReachability.h

Overview

The `SCNetworkReachability` programming interface allows an application to determine the status of a system's current network configuration and the reachability of a target host. A remote host is considered reachable when a data packet, sent by an application into the network stack, can leave the local device. Reachability does not guarantee that the data packet will actually be received by the host.

The `SCNetworkReachability` programming interface supports a synchronous and an asynchronous model. In the synchronous model, you get the reachability status by calling the `SCNetworkReachabilityGetFlags` (page 12) function. In the asynchronous model, you can schedule the `SCNetworkReachability` object on the run loop of a client object's thread. The client implements a callback function to receive notifications when the reachability status of a given remote host changes. Note that these functions follow Core Foundation naming conventions. A function that has "Create" or "Copy" in its name returns a reference you must release with the `CFRelease` function.

For information about detecting and interpreting errors generated by calling these functions, see *System Configuration Reference*.

Functions by Task

Creating a Reachability Reference

`SCNetworkReachabilityCreateWithAddress` (page 10)
Creates a reachability reference to the specified network address.

`SCNetworkReachabilityCreateWithAddressPair` (page 11)
Creates a reachability reference to the specified network address.

`SCNetworkReachabilityCreateWithName` (page 11)
Creates a reachability reference to the specified network host or node name.

Determining Reachability Status

`SCNetworkReachabilityGetFlags` (page 12)
Determines if the specified network target is reachable using the current network configuration.

Preparing to Determine Reachability

`SCNetworkReachabilityGetTypeID` (page 12)
: Returns the type identifier of all `SCNetworkReachability` instances.

`SCNetworkReachabilitySetCallback` (page 13)
: Assigns a client to the specified target, which receives callbacks when the reachability of the target changes.

`SCNetworkReachabilityScheduleWithRunLoop` (page 13)
: Schedules the specified network target with the specified run loop and mode.

`SCNetworkReachabilityUnscheduleFromRunLoop` (page 14)
: Unschedules the specified target from the specified run loop and mode.

`SCNetworkReachabilitySetDispatchQueue` (page 14)
: Schedules callbacks for the specified target on the specified dispatch queue.

Functions

SCNetworkReachabilityCreateWithAddress

Creates a reachability reference to the specified network address.

```
SCNetworkReachabilityRef SCNetworkReachabilityCreateWithAddress (
   CFAllocatorRef allocator,
   const struct sockaddr *address
);
```

Parameters

allocator
: The allocator to use. Pass `NULL` or `kCFAllocatorDefault` to use the default allocator.

address
: The address of the desired host.

Return Value
A new immutable reachability reference. You must release the returned value.

Discussion
You can use the reachability reference returned by this function to monitor the reachability of the target host.

Availability
Available in iOS 2.0 and later.

Related Sample Code
CryptoExercise

Declared In
`SCNetworkReachability.h`

SCNetworkReachabilityCreateWithAddressPair

Creates a reachability reference to the specified network address.

```
SCNetworkReachabilityRef SCNetworkReachabilityCreateWithAddressPair (
   CFAllocatorRef allocator,
   const struct sockaddr *localAddress,
   const struct sockaddr *remoteAddress
);
```

Parameters

allocator

The allocator to use. Pass `NULL` or `kCFAllocatorDefault` to use the default allocator.

localAddress

The local address associated with a network connection. If `NULL`, only the remote address is of interest.

remoteAddress

The remote address associated with a network connection. If `NULL`, only the local address is of interest.

Return Value

A new immutable reachability reference. You must release the returned value.

Discussion

You can use the reachability reference returned by this function to monitor the reachability of the target host.

Availability

Available in iOS 2.0 and later.

Declared In

`SCNetworkReachability.h`

SCNetworkReachabilityCreateWithName

Creates a reachability reference to the specified network host or node name.

```
SCNetworkReachabilityRef SCNetworkReachabilityCreateWithName (
   CFAllocatorRef allocator,
   const char *nodename
);
```

Parameters

allocator

The allocator to use. Pass `NULL` or `kCFAllocatorDefault` to use the default allocator.

nodename

The node name of the desired host. This name is the same as that passed to the `gethostbyname(3)` or `getaddrinfo(3)` functions.

Return Value

A new immutable reachability reference. You must release the returned value.

Discussion

You can use the reachability reference returned by this function to monitor the reachability of the target host.

Availability
Available in iOS 2.0 and later.

Declared In
`SCNetworkReachability.h`

SCNetworkReachabilityGetFlags

Determines if the specified network target is reachable using the current network configuration.

```
Boolean SCNetworkReachabilityGetFlags (
   SCNetworkReachabilityRef target,
   SCNetworkReachabilityFlags *flags
);
```

Parameters

`target`
The network reference associated with the address or name to be checked for reachability.

`flags`
A pointer to memory that, on output, is filled with flags that describe the reachability of the specified target. (See "Network Reachability Flags" (page 16) for possible values.)

Return Value
`TRUE` if the flags are valid; `FALSE` if the status could not be determined.

Availability
Available in iOS 2.0 and later.

Related Sample Code
CryptoExercise

Declared In
`SCNetworkReachability.h`

SCNetworkReachabilityGetTypeID

Returns the type identifier of all `SCNetworkReachability` instances.

```
CFTypeID SCNetworkReachabilityGetTypeID (
   void
);
```

Return Value
The type identifier of all `SCNetworkReachability` instances.

Availability
Available in iOS 2.0 and later.

Declared In
`SCNetworkReachability.h`

SCNetworkReachabilityScheduleWithRunLoop

Schedules the specified network target with the specified run loop and mode.

```
Boolean SCNetworkReachabilityScheduleWithRunLoop (
   SCNetworkReachabilityRef target,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

target
: The address or name that is set up for asynchronous notifications. Must not be `NULL`.

runLoop
: The run loop on which the target should be scheduled. Must not be `NULL`.

runLoopMode
: The mode in which to schedule the target. Must not be `NULL`.

Return Value

`TRUE` if the target is scheduled successfully; otherwise, `FALSE`.

Availability

Available in iOS 2.0 and later.

Declared In

`SCNetworkReachability.h`

SCNetworkReachabilitySetCallback

Assigns a client to the specified target, which receives callbacks when the reachability of the target changes.

```
Boolean SCNetworkReachabilitySetCallback (
   SCNetworkReachabilityRef target,
   SCNetworkReachabilityCallBack callout,
   SCNetworkReachabilityContext *context
);
```

Parameters

target
: The network reference associated with the address or name to be checked for reachability.

callout
: The function to be called when the reachability of the target changes. If `NULL`, the current client for the target is removed.

context
: The reachability context associated with the callout. This value may be `NULL`.

Return Value

`TRUE` if the notification client was successfully set; otherwise, `FALSE`.

Availability

Available in iOS 2.0 and later.

Declared In

`SCNetworkReachability.h`

SCNetworkReachabilitySetDispatchQueue

Schedules callbacks for the specified target on the specified dispatch queue.

```
Boolean SCNetworkReachabilitySetDispatchQueue (
    SCNetworkReachabilityRef target,
    dispatch_queue_t queue
);
```

Parameters

target

The address or name that is set up for asynchronous notifications. Must not be `NULL`.

queue

The libdispatch queue on which the target should run. Pass `NULL` to disable notifications and release the queue.

Return Value

`TRUE` if the target is scheduled successfully; otherwise, `FALSE`.

Availability

Available in iOS 4.0 and later.

Declared In

`SCNetworkReachability.h`

SCNetworkReachabilityUnscheduleFromRunLoop

Unschedules the specified target from the specified run loop and mode.

```
Boolean SCNetworkReachabilityUnscheduleFromRunLoop (
   SCNetworkReachabilityRef target,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

target

The address or name that is set up for asynchronous notifications. Must not be `NULL`.

runLoop

The run loop on which the target should be unscheduled. Must not be `NULL`.

runLoopMode

The mode in which to unschedule the target. Must not be `NULL`.

Return Value

`TRUE` if the target is unscheduled successfully; otherwise, `FALSE`.

Availability

Available in iOS 2.0 and later.

Declared In

`SCNetworkReachability.h`

Data Types

SCNetworkReachabilityRef

The handle to a network address or name.

```
typedef const struct __SCNetworkReachability * SCNetworkReachabilityRef;
```

Availability
Available in iOS 2.0 and later.

Declared In
`SCNetworkReachability.h`

SCNetworkReachabilityContext

Structure containing user-specified data and callbacks used with `SCNetworkReachabilitySetCallback` (page 13).

```
typedef struct {
    CFIndex version;
    void * info;
    const void *(*retain)(const void *info);
    void (*release)(const void *info);
    CFStringRef (*copyDescription)(const void *info);
} SCNetworkReachabilityContext;
```

Fields

`version`
: The version number of the structure type being passed in as a parameter to an `SCDynamicStore` creation function, such as `SCDynamicStoreCreate`. This structure is version `0`.

`info`
: A C pointer to a user-specified block of data.

`retain`
: The callback used to add a retain for the info field. If this parameter is not a pointer to a function of the correct prototype, the behavior is undefined. The value can be `NULL`.

`release`
: The calllback used to remove a retain previously added for the info field. If this parameter is not a pointer to a function of the correct prototype, the behavior is undefined. The value can be `NULL`.

`copyDescription`
: The callback used to provide a description of the *`info`* field.

Availability
Available in iOS 2.0 and later.

Declared In
`SCNetworkReachability.h`

SCNetworkReachabilityCallBack

Type of callback function used when the reachability of a network address or name changes.

```
typedef void (*SCNetworkReachabilityCallBack)    (
               SCNetworkReachabilityRef target,
               SCNetworkReachabilityFlags flags,
               void *info
);
```

Fields

`target`
: The network target being monitored for changes.

`flags`
: The flags representing the reachability status of the network address or name (see "Network Reachability Flags" (page 16) for information about these flags).

`info`
: A C pointer to a user-specified block of data.

Constants

Network Reachability Flags

Flags that indicate the reachability of a network node name or address, including whether a connection is required, and whether some user intervention might be required when establishing a connection.

```
enum {
    kSCNetworkReachabilityFlagsTransientConnection = 1<<0,
    kSCNetworkReachabilityFlagsReachable = 1<<1,
    kSCNetworkReachabilityFlagsConnectionRequired = 1<<2,
    kSCNetworkReachabilityFlagsConnectionOnTraffic = 1<<3,
    kSCNetworkReachabilityFlagsInterventionRequired = 1<<4,
    kSCNetworkReachabilityFlagsConnectionOnDemand = 1<<5,
    kSCNetworkReachabilityFlagsIsLocalAddress = 1<<16,
    kSCNetworkReachabilityFlagsIsDirect = 1<<17,
    kSCNetworkReachabilityFlagsIsWWAN = 1<<18,
    kSCNetworkReachabilityFlagsConnectionAutomatic    =
kSCNetworkReachabilityFlagsConnectionOnTraffic
};
typedef    uint32_t    SCNetworkReachabilityFlags;
```

Constants

`kSCNetworkReachabilityFlagsTransientConnection`
: The specified node name or address can be reached via a transient connection, such as PPP.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsReachable`
: The specified node name or address can be reached using the current network configuration.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsConnectionRequired`
: The specified node name or address can be reached using the current network configuration, but a connection must first be established.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsConnectionOnTraffic`
: The specified node name or address can be reached using the current network configuration, but a connection must first be established. Any traffic directed to the specified name or address will initiate the connection.

  This flag was previously named `kSCNetworkReachabilityFlagsConnectionAutomatic`.

  Available in iOS 3.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsInterventionRequired`
: The specified node name or address can be reached using the current network configuration, but a connection must first be established.

  In addition, some form of user intervention will be required to establish this connection, such as providing a password, an authentication token, etc.

  Currently, this flag is returned only when there is a dial-on-traffic configuration (`kSCNetworkReachabilityFlagsConnectionOnTraffic`), an attempt to connect has already been made, and when some error (such as no dial tone, no answer, bad password, etc.) occurred during the automatic connection attempt. In this case the PPP controller stops attempting to establish a connection until the user has intervened.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsConnectionOnDemand`
: The specified node name or address can be reached using the current network configuration, but a connection must first be established. The connection will be established "On Demand" by the `CFSocketStream` programming interface (see *CFStream Socket Additions* for information on this). Other functions will not establish the connection.

  Available in iOS 3.0 through iOS 3.2.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsIsLocalAddress`
: The specified node name or address is one that is associated with a network interface on the current system.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsIsDirect`
: Network traffic to the specified node name or address will not go through a gateway, but is routed directly to one of the interfaces in the system.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsIsWWAN`
: The specified node name or address can be reached via a cellular connection, such as EDGE or GPRS.

  Available in iOS 2.0 and later.

  Declared in `SCNetworkReachability.h`.

System Configuration Reference

Framework: SystemConfiguration

Declared in SystemConfiguration.h

Overview

The `SystemConfiguration` programming interface provides functions you can use to get and interpret status and error codes generated as a result of calling functions of the System Configuration framework.

Functions

SCCopyLastError

Returns an error or status code associated with the most recent function call.

```
CFErrorRef SCCopyLastError (
   void
);
```

Return Value
The most recent status or error code generated as the result of calling a function defined by the System Configuration framework. The code is represented by a Core Foundation `CFErrorRef` opaque type.

Discussion
Call the `CFErrorGetCode` function on the returned object to get the underlying error-code integer. See "Status and Error Codes" (page 20) for descriptions of these codes. For more on `CFErrorRef` objects, see *CFError Reference*.

Availability
Available in iOS 2.0 and later.

Declared In
`SystemConfiguration.h`

SCError

Returns an error or status code associated with the most recent function call.

```
int SCError (
   void
);
```

Return Value
The most recent status or error code generated as the result of calling a function defined by the System Configuration framework. See "Status and Error Codes" (page 20) for descriptions of these codes.

Availability
Available in iOS 2.0 and later.

Declared In
`SystemConfiguration.h`

SCErrorString

Returns a string describing the specified status code or error code.

```
const char * SCErrorString (
   int status
);
```

Parameters

`status`
: A status or error code described in "Status and Error Codes" (page 20). You typically get this code by calling `SCError` (page 19) or `SCCopyLastError` (page 19).

Return Value
The message string associated with the status or error identified by *`status`*.

Availability
Available in iOS 2.0 and later.

Declared In
`SystemConfiguration.h`

Constants

Status and Error Codes

The status or error code generated by the most recent System Configuration function call.

```
enum {
    kSCStatusOK = 0,
    kSCStatusFailed = 1001,
    kSCStatusInvalidArgument = 1002,
    kSCStatusAccessError = 1003,
    kSCStatusNoKey = 1004,
    kSCStatusKeyExists = 1005,
    kSCStatusLocked = 1006,
    kSCStatusNeedLock = 1007,
    kSCStatusNoStoreSession = 2001,
    kSCStatusNoStoreServer = 2002,
    kSCStatusNotifierActive = 2003,
    kSCStatusNoPrefsSession = 3001,
    kSCStatusPrefsBusy = 3002,
    kSCStatusNoConfigFile = 3003,
    kSCStatusNoLink = 3004,
    kSCStatusStale = 3005,
    kSCStatusMaxLink = 3006,
    kSCStatusReachabilityUnknown = 4001,
    kSCStatusConnectionNoService = 5001
};
```

Constants

`kSCStatusOK`
The call was successful.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusFailed`
A nonspecific failure occurred.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusInvalidArgument`
An invalid argument was specified.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusAccessError`
Permission is denied; you must be root to obtain a lock. As a result, the function could not create or access preferences.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoKey`
No such key.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusKeyExists`
The key is already defined.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusLocked`
A lock is already held.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNeedLock`
A lock is required for this operation.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoStoreSession`
The configuration daemon session is not active.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoStoreServer`
The configuration daemon is not available or no longer available.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNotifierActive`
Notifier is currently active.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoPrefsSession`
The preferences session is not active.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusPrefsBusy`
A preferences update is currently in progress.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoConfigFile`
The configuration file cannot be found.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoLink`
No such link exists.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusStale`
A write was attempted on a stale version of the object.

Available in iOS 2.0 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusMaxLink`
: The maximum link count is exceeded.

  Available in iOS 2.0 and later.

  Declared in `SystemConfiguration.h`.

`kSCStatusReachabilityUnknown`
: Network reachability cannot be determined.

  Available in iOS 2.0 and later.

  Declared in `SystemConfiguration.h`.

`kSCStatusConnectionNoService`
: Network service for the connection is not available.

  Available in iOS 2.0 through iOS 3.2.

  Declared in `SystemConfiguration.h`.

Error Domain

The error domain associated with errors reported by the System Configuration framework.

```
const CFStringRef    kCFErrorDomainSystemConfiguration;
```

Constants

`kCFErrorDomainSystemConfiguration`
: A string constant identifying a Core Foundation error domain. See *CFError Reference* for further information on error domains.

  Available in iOS 2.0 and later.

  Declared in `SystemConfiguration.h`.

Document Revision History

This table describes the changes to *System Configuration Framework Reference*.

Date	Notes
2009-07-30	Made minor corrections.
2009-03-05	TBD