CFHost Reference

Networking > Core Foundation

2008-07-11

Contents

CFHost Reference

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

Overview

The CFHost API allows you to create instances of the CFHost object that you can use to acquire host information, including names, addresses, and reachability information.

The process of acquiring information about a host is known as resolution. Begin by calling `CFHostCreateWithAddress` or `CFHostCreateWithName` to create an instance of a CFHost using an address or a name, respectively. If you want to resolve the host asynchronously. call `CFHostSetClient` to associate your client context and user-defined callback function with the host. Then call `CFHostScheduleWithRunLoop` to schedule the host on a run loop.

To start resolution, call `CFHostStartInfoResolution`. If you set up for asynchronous resolution, `CFHostStartInfoResolution` returns immediately. Your callback function will be called when resolution is complete. If you didn't set up for asynchronous resolution, `CFHostStartInfoResolution` blocks until resolution is complete, an error occurs, or the resolution is cancelled.

When resolution is complete, call `CFHostGetAddressing` or `CFHostGetNames` to get an array of known addresses or known names, respectively, for the host. Call `CFHostGetReachability` to get flags, declared in `SystemConfiguration/SCNetwork.h`, that describe the reachability of the host.

When you no longer need a CFHost object, call `CFHostUnscheduleFromRunLoop` and `CFHostSetClient` to disassociate the host from your user-defined client context and callback function (if it was set up for asynchronous resolution). Then dispose of it.

Functions by Task

Creating a host

`CFHostCreateCopy` (page 7)
Creates a new host object by copying.

`CFHostCreateWithAddress` (page 7)
Uses an address to create an instance of a host object.

`CFHostCreateWithName` (page 8)
: Uses a name to create an instance of a host object.

CFHost Functions

`CFHostCancelInfoResolution` (page 6)
: Cancels the resolution of a host.

`CFHostGetAddressing` (page 9)
: Gets the addresses from a host.

`CFHostGetNames` (page 9)
: Gets the names from a CFHost.

`CFHostGetReachability` (page 10)
: Gets reachability information from a host.

`CFHostStartInfoResolution` (page 12)
: Starts resolution for a host object.

`CFHostSetClient` (page 12)
: Associates a client context and a callback function with a CFHost object or disassociates a client context and callback function that were previously set.

`CFHostScheduleWithRunLoop` (page 11)
: Schedules a CFHost on a run loop.

`CFHostUnscheduleFromRunLoop` (page 13)
: Unschedules a CFHost from a run loop.

Getting the CFHost Type ID

`CFHostGetTypeID` (page 11)
: Gets the Core Foundation type identifier for the CFHost opaque type.

Functions

CFHostCancelInfoResolution

Cancels the resolution of a host.

```
void CFHostCancelInfoResolution (
   CFHostRef theHost,
   CFHostInfoType info
);
```

Parameters

theHost
: The host for which a resolution is to be cancelled. This value must not be `NULL`.

info

A value of type `CFHostInfoType` specifying the type of resolution that is to be cancelled. See CFHostInfoType Constants (page 16) for possible values.

Discussion

This function cancels the asynchronous or synchronous resolution specified by `info` for the host specified by `theHost`.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostCreateCopy

Creates a new host object by copying.

```
CFHostRef CFHostCreateCopy (
   CFAllocatorRef alloc,
   CFHostRef host
);
```

Parameters

alloc

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

addr

The host to copy. This value must not be `NULL`.

Return Value

A valid CFHost object or `NULL` if the copy could not be created. The new host contains a copy of all previously resolved data from the original host. Ownership follows the Create Rule.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostCreateWithAddress

Uses an address to create an instance of a host object.

```
CFHostRef CFHostCreateWithAddress (
   CFAllocatorRef allocator,
   CFDataRef addr
);
```

Parameters

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

addr
: A CFDataRef object containing a `sockaddr` structure for the address of the host. This value must not be `NULL`.

Return Value

A valid CFHostRef object that can be resolved, or `NULL` if the host could not be created. Ownership follows the Create Rule.

Discussion

Call `CFHostStartInfoResolution` (page 12) to resolve the return object's name and reachability information.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostCreateWithName

Uses a name to create an instance of a host object.

```
CFHostRef CFHostCreateWithName (
   CFAllocatorRef allocator,
   CFStringRef hostname
);
```

Parameters

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

hostname
: A string representing the name of the host. This value must not be `NULL`.

Return Value

A valid CFHostRef object that can be resolved, or `NULL` if the host could not be created. Ownership follows the Create Rule.

Discussion

Call `CFHostStartInfoResolution` (page 12) to resolve the object's addresses and reachability information.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostGetAddressing

Gets the addresses from a host.

```
CFArrayRef CFHostGetAddressing (
   CFHostRef theHost,
   Boolean *hasBeenResolved
);
```

Parameters

theHost

The CFHost whose addresses are to be obtained. This value must not be `NULL`.

hasBeenResolved

On return, a pointer to a Boolean that is `TRUE` if addresses were available and `FALSE` if addresses were not available. This parameter can be null.

function result

A CFArray of addresses where address is a `sockaddr` structure wrapped by a CFDataRef, or null if no addresses were available.

Discussion

This function gets the addresses from a CFHost. The CFHost must have been previously resolved. To resolve a CFHost, call `CFHostStartInfoResolution` (page 12).

Special Considerations

This function gets the addresses in a thread-safe way, but the resulting data is not thread-safe. The data is returned as a "get" as opposed to a copy, so the data is not safe if the CFHost is altered from another thread.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostGetNames

Gets the names from a CFHost.

```
CFArrayRef CFHostGetNames (
   CFHostRef theHost,
   Boolean *hasBeenResolved
);
```

Parameters

theHost

The host to examine. The host must have been previously resolved. (To resolve a host, call `CFHostStartInfoResolution` (page 12).) This value must not be `NULL`.

hasBeenResolved

On return, contains `TRUE` if names were available, otherwise `FALSE`. This value may be `NULL`.

Return Value

An array containing the of names of *theHost*, or `NULL` if no names were available.

Special Considerations

This function gets the names in a thread-safe way, but the resulting data is not thread-safe. The data is returned as a “get” as opposed to a copy, so the data is not safe if the CFHost is altered from another thread.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostGetReachability

Gets reachability information from a host.

```
CFDataRef CFHostGetReachability (
   CFHostRef theHost,
   Boolean *hasBeenResolved
);
```

Parameters

theHost

The host whose reachability is to be obtained. The host must have been previously resolved. (To resolve a host, call `CFHostStartInfoResolution` (page 12).) This value must not be `NULL`.

hasBeenResolved

On return, contains `TRUE` if the reachability was available, otherwise `FALSE`. This value may be `NULL`.

Return Value

A CFData object that wraps the reachability flags (`SCNetworkConnectionFlags`) defined in `SystemConfiguration/SCNetwork.h`, or `NULL` if reachability information was not available.

Special Considerations

This function gets reachability information in a thread-safe way, but the resulting data is not thread-safe. The data is returned as a “get” as opposed to a copy, so the data is not safe if the CFHost is altered from another thread.

Availability

Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

CFHostGetTypeID

Gets the Core Foundation type identifier for the CFHost opaque type.

```
CFTypeID CFHostGetTypeID ();
```

Return Value
The Core Foundation type identifier for the CFHost opaque type.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

CFHostScheduleWithRunLoop

Schedules a CFHost on a run loop.

```
void CFHostScheduleWithRunLoop (
   CFHostRef theHost,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

theHost
    The host to be schedule on a run loop. This value must not be `NULL`.

runLoop
    The run loop on which to schedule *theHost*. This value must not be `NULL`.

runLoopMode
    The mode on which to schedule *theHost*. This value must not be `NULL`.

Discussion
Schedules *theHost* on a run loop, which causes resolutions of the host to be performed asynchronously. The caller is responsible for ensuring that at least one of the run loops on which the host is scheduled is being run.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

CFHostSetClient

Associates a client context and a callback function with a CFHost object or disassociates a client context and callback function that were previously set.

```
Boolean CFHostSetClient (
   CFHostRef theHost,
   CFHostClientCallBack clientCB,
   CFHostClientContext *clientContext
);
```

Parameters

theHost

The host to modify. The value must not be `NULL`.

clientCB

The callback function to associate with *theHost*. The callback function will be called when a resolution completes or is cancelled. If you are calling this function to disassociate a client context and callback from `theHost`, p`clientCB`ass `NULL`.

clientContext

A `CFHostClientContext` (page 15) structure whose `info` field will be passed to the callback function specified by `clientCB` when `clientCB` is called. This value must not be `NULL` when setting an association.

Pass `NULL` when disassociating a client context and a callback from a host.

Return Value
`TRUE` if the association could be set or unset, otherwise `FALSE`.

Discussion
The callback function specified by `clientCB` will be called when a resolution completes or is cancelled.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

CFHostStartInfoResolution

Starts resolution for a host object.

```
Boolean CFHostStartInfoResolution (
   CFHostRef theHost,
   CFHostInfoType info,
   CFStreamError *error
);
```

Parameters

theHost

The host, obtained by previously calling `CFHostCreateCopy` (page 7), `CFHostCreateWithAddress` (page 7), or `CFHostCreateWithName` (page 8), that is to be resolved. This value must not be `NULL`.

info

A value of type `CFHostInfoType` specifying the type of information that is to be retrieved. See CFHostInfoType Constants (page 16) for possible values.

error

A pointer to a `CFStreamError` structure, that, if an error occurs, is set to the error and the error's domain. In synchronous mode, the error indicates why resolution failed, and in asynchronous mode, the error indicates why resolution failed to start.

Return Value

`TRUE` if the resolution was started (asynchronous mode); `FALSE` if another resolution is already in progress for *theHost* or if an error occurred.

Discussion

This function retrieves the information specified by `info` and stores it in the host.

In synchronous mode, this function blocks until the resolution has completed, in which case this function returns `TRUE`, until the resolution is stopped by calling `CFHostCancelInfoResolution` (page 6) from another thread, in which case this function returns `FALSE`, or until an error occurs.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFHost.h`

CFHostUnscheduleFromRunLoop

Unschedules a CFHost from a run loop.

```
void CFHostUnscheduleFromRunLoop (
   CFHostRef theHost,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

theService

The host to unschedule. This value must not be `NULL`.

runLoop

The run loop. This value must not be `NULL`.

runLoopMode

The mode from which the service is to be unscheduled. This value must not be `NULL`.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

Callbacks

CFHostClientCallBack

Defines a pointer to the callback function that is called when an asynchronous resolution of a CFHost completes or an error occurs for an asynchronous CFHost resolution.

```
typedef void (CFHostClientCallBack) (
   CFHostRef theHost,
   CFHostInfoType typeInfo,
   const CFStreamError *error,
   void *info);
```

If you name your callback `MyHostClientCallBack`, you would declare it like this:

```
void MyHostClientCallBack (
   CFHostRef theHost,
   CFHostInfoType typeInfo,
   const CFStreamError *error,
   void *info
);
```

Parameters

`theHost`
The host for which an asynchronous resolution has been completed.

`typeInfo`
Value of type `CFHostInfoType` representing the type of information (addresses, names, or reachability information) obtained by the completed resolution. See CFHostInfoType Constants (page 16) for possible values.

`error`
If the resolution failed, contains a `CFStreamError` structure whose `error` field contains an error code.

`info`
User-defined context information. The value pointed to by `info` is the same as the value pointed to by the `info` field of the `CFHostClientContext` (page 15) structure that was provided when the host was associated with this callback function.

Discussion
The callback function for a CFHost object is called one or more times when an asynchronous resolution completes for the specified host, when an asynchronous resolution is cancelled, or when an error occurs during an asynchronous resolution.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

Data Types

CFHostRef

An opaque reference representing an CFHost object.

```
typedef struct __CFHost* CFHostRef;
```

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

CFHostClientContext

A structure containing user-defined data and callbacks for CFHost objects.

```
struct CFHostClientContext {
    CFIndex version;
    void *info;
    CFAllocatorRetainCallBack retain;
    CFAllocatorReleaseCallBack release;
    CFAllocatorCopyDescriptionCallBack copyDescription;
} CFHostClientContext;
typedef struct CFHostClientContext CFHostClientContext;
```

Fields

`version`
: The version number of the structure type passed as a parameter to the host client function. The only valid version number is `0`.

`info`
: An arbitrary pointer to allocated memory containing user-defined data that can be associated with the host and that is passed to the callbacks.

`retain`
: The callback used to add a retain for the host on the info pointer for the life of the host, and may be used for temporary references the host needs to take. This callback returns the actual info pointer to store in the host, almost always just the pointer passed as the parameter.

`release`
: The callback used to remove a retain previously added for the host on the info pointer.

`copyDescription`
: The callback used to create a descriptive string representation of the info pointer (or the data pointed to by the info pointer) for debugging purposes. This callback is called by the `CFCopyDescription` function.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
`CFHost.h`

Constants

CFHostInfoType Constants

Values indicating the type of data that is to be resolved or the type of data that was resolved.

```
enum CFHostInfoType {
    kCFHostAddresses = 0,
    kCFHostNames = 1,
    kCFHostReachability = 2
};
typedef enum CFHostInfoType CFHostInfoType;
```

Constants

`kCFHostAddresses`

Specifies that addresses are to be resolved or that addresses were resolved.

Available in Mac OS X v10.3 and later.

Declared in `CFHost.h`.

`kCFHostNames`

Specifies that names are to be resolved or that names were resolved.

Available in Mac OS X v10.3 and later.

Declared in `CFHost.h`.

`kCFHostReachability`

Specifies that reachability information is to be resolved or that reachability information was resolved.

Available in Mac OS X v10.3 and later.

Declared in `CFHost.h`.

Availability

Available in Mac OS X version 10.3 and later.

Declared In

`CFNetwork/CFHost.h`

Error Domains

Error domains specific to `CFHost` calls.

```
extern const SInt32 kCFStreamErrorDomainNetDB;
extern const SInt32 kCFStreamErrorDomainSystemConfiguration;
```

Constants

`kCFStreamErrorDomainNetDB`

The error domain that returns errors from the network database (DNS resolver) layer (described in `/usr/include/netdb.h`).

Available in Mac OS X version 10.5 and later.

Declared in `CFHost.h`.

`kCFStreamErrorDomainSystemConfiguration`

The error domain that returns errors from the system configuration layer (described in *System Configuration Framework Reference*).

Available in Mac OS X version 10.5 and later.

Declared in `CFHost.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.

Document Revision History

This table describes the changes to *CFHost Reference*.

Date	Notes
2008-07-11	Added missing constants.
2006-07-06	Made minor formatting changes.
2006-03-08	New document that describes the C API for acquiring host information.

Index