CFNetServices Reference

Core Foundation > Networking

2008-07-08

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

Simultaneously published in the United States and Canada.

Contents

CFNetServices Reference 5

Overview 5
Functions by Task 5
- Creating net service objects 5
- CFNetServices Functions 6
- Modifying a net service 7
- Getting the net service type IDs 7

Functions 8
- CFNetServiceBrowserCreate 8
- CFNetServiceBrowserGetTypeID 9
- CFNetServiceBrowserInvalidate 9
- CFNetServiceBrowserScheduleWithRunLoop 10
- CFNetServiceBrowserSearchForDomains 10
- CFNetServiceBrowserSearchForServices 11
- CFNetServiceBrowserStopSearch 12
- CFNetServiceBrowserUnscheduleFromRunLoop 13
- CFNetServiceCancel 14
- CFNetServiceCreate 14
- CFNetServiceCreateCopy 16
- CFNetServiceCreateDictionaryWithTXTData 16
- CFNetServiceCreateTXTDataWithDictionary 17
- CFNetServiceGetAddressing 18
- CFNetServiceGetDomain 18
- CFNetServiceGetName 19
- CFNetServiceGetPortNumber 19
- CFNetServiceGetProtocolSpecificInformation 20
- CFNetServiceGetTargetHost 20
- CFNetServiceGetTXTData 21
- CFNetServiceGetType 21
- CFNetServiceGetTypeID 22
- CFNetServiceMonitorCreate 22
- CFNetServiceMonitorGetTypeID 24
- CFNetServiceMonitorInvalidate 24
- CFNetServiceMonitorScheduleWithRunLoop 24
- CFNetServiceMonitorStart 25
- CFNetServiceMonitorStop 26
- CFNetServiceMonitorUnscheduleFromRunLoop 27
- CFNetServiceRegister 27
- CFNetServiceRegisterWithOptions 28
- CFNetServiceResolve 29
- CFNetServiceResolveWithTimeout 30

CFNetServiceScheduleWithRunLoop 31
CFNetServiceSetClient 32
CFNetServiceSetProtocolSpecificInformation 32
CFNetServiceSetTXTData 33
CFNetServiceUnscheduleFromRunLoop 34
Callbacks 34
CFNetServiceBrowserClientCallBack 34
CFNetServiceClientCallBack 35
CFNetServiceMonitorClientCallBack 36
Data Types 37
CFNetServiceBrowserRef 37
CFNetServiceClientContext 37
CFNetServiceMonitorRef 38
CFNetServiceRef 39
Constants 39
CFNetService Registration Options 39
CFNetServiceBrowserClientCallBack Bit Flags 39
CFNetServiceMonitorType Constants 40
CFNetService Error Constants 41
Error Domains 42

Document Revision History 43

Index 45

CFNetServices Reference

Derived From:	CFType
Framework:	CoreServices
Declared in	CFNetServices.h
Companion guides	Bonjour Overview CFNetwork Programming Guide NSNetServices and CFNetServices Programming Guide

Overview

The CFNetServices API is part of Bonjour, Apple's implementation of zero-configuration networking (ZEROCONF). The CFNetServices API allows you to register a network service, such as a printer or file server, so that it can be found by name or browsed for by service type and domain. Applications can use the CFNetServices API to discover the services that are available on the network and to find all access information — such as name, IP address, and port number — needed to use each service.

In effect, Bonjour registration and discovery combine the functions of a local DNS server and AppleTalk, allowing applications to provide the kind of user-friendly browsing available in the AppleTalk Chooser using open protocols, such as Multicast DNS (mDNS). Bonjour gives applications easy access to services over local IP networks without requiring the service to support an AppleTalk stack, and without requiring a DNS server on the local network.

For a full description of Bonjour, see *Bonjour Overview*.

Functions by Task

Creating net service objects

`CFNetServiceCreate` (page 14)
Creates an instance of a Network Service object.

`CFNetServiceCreateCopy` (page 16)
Creates a copy of a CFNetService object.

`CFNetServiceMonitorCreate` (page 22)
Creates an instance of a NetServiceMonitor object that watches for record changes.

`CFNetServiceBrowserCreate` (page 8)
Creates an instance of a Network Service browser object.

CFNetServices Functions

`CFNetServiceBrowserInvalidate` (page 9)
Invalidates an instance of a Network Service browser object.

`CFNetServiceBrowserScheduleWithRunLoop` (page 10)
Schedules a CFNetServiceBrowser on a run loop.

`CFNetServiceBrowserSearchForDomains` (page 10)
Searches for domains.

`CFNetServiceBrowserSearchForServices` (page 11)
Searches a domain for services of a specified type.

`CFNetServiceBrowserStopSearch` (page 12)
Stops a search for domains or services.

`CFNetServiceBrowserUnscheduleFromRunLoop` (page 13)
Unschedules a CFNetServiceBrowser from a run loop and mode.

`CFNetServiceCancel` (page 14)
Cancels a service registration or a service resolution.

`CFNetServiceCreateDictionaryWithTXTData` (page 16)
Uses TXT record data to create a dictionary.

`CFNetServiceCreateTXTDataWithDictionary` (page 17)
Flattens a set of key/value pairs into a CFDataRef suitable for passing to `CFNetServiceSetTXTData` (page 33).

`CFNetServiceGetAddressing` (page 18)
Gets the IP addressing from a CFNetService.

`CFNetServiceGetTargetHost` (page 20)
Queries a CFNetService for its target hosts.

`CFNetServiceGetDomain` (page 18)
Gets the domain from a CFNetService.

`CFNetServiceGetName` (page 19)
Gets the name from a CFNetService.

`CFNetServiceGetPortNumber` (page 19)
This function gets the port number from a CFNetService.

`CFNetServiceGetProtocolSpecificInformation` (page 20)
This function gets protocol-specific information from a CFNetService. (Deprecated. Use `CFNetServiceGetTXTData` (page 21) instead.)

`CFNetServiceGetTXTData` (page 21)
Queries a network service for the contents of its TXT records.

`CFNetServiceGetType` (page 21)
Gets the type from a CFNetService.

`CFNetServiceMonitorInvalidate` (page 24)
Invalidates an instance of a Network Service monitor object.

`CFNetServiceMonitorScheduleWithRunLoop` (page 24)
Schedules a CFNetServiceMonitor on a run loop.

`CFNetServiceMonitorStart` (page 25)
Starts monitoring.

`CFNetServiceMonitorStop` (page 26)
Stops a CFNetServiceMonitor.

`CFNetServiceMonitorUnscheduleFromRunLoop` (page 27)
Unschedules a CFNetServiceMonitor from a run loop.

`CFNetServiceRegister` (page 27)
Makes a CFNetService available on the network. (Deprecated. Use `CFNetServiceRegisterWithOptions` (page 28) instead.)

`CFNetServiceRegisterWithOptions` (page 28)
Makes a CFNetService available on the network.

`CFNetServiceResolve` (page 29)
This function updates the specified CFNetService with the IP address or addresses associated with the service. Call `CFNetServiceGetAddressing` (page 18) to get the addresses. (Deprecated. Use `CFNetServiceResolveWithTimeout` (page 30) instead.)

`CFNetServiceResolveWithTimeout` (page 30)
Gets the IP address or addresses for a CFNetService.

`CFNetServiceScheduleWithRunLoop` (page 31)
Schedules a CFNetService on a run loop.

`CFNetServiceSetClient` (page 32)
Associates a callback function with a CFNetService or disassociates a callback function from a CFNetService.

`CFNetServiceSetTXTData` (page 33)
Sets the TXT record for a CFNetService.

`CFNetServiceUnscheduleFromRunLoop` (page 34)
Unschedules a CFNetService from a run loop.

Modifying a net service

`CFNetServiceSetProtocolSpecificInformation` (page 32)
Sets protocol-specific information for a CFNetService. (Deprecated. Use `CFNetServiceSetTXTData` instead.)

Getting the net service type IDs

`CFNetServiceGetTypeID` (page 22)
Gets the Core Foundation type identifier for the Network Service object.

`CFNetServiceMonitorGetTypeID` (page 24)
Gets the Core Foundation type identifier for all CFNetServiceMonitor instances.

`CFNetServiceBrowserGetTypeID` (page 9)
Gets the Core Foundation type identifier for the Network Service browser object.

Functions

CFNetServiceBrowserCreate

Creates an instance of a Network Service browser object.

```
CFNetServiceBrowserRef CFNetServiceBrowserCreate (
   CFAllocatorRef alloc,
   CFNetServiceBrowserClientCallBack clientCB,
   CFNetServiceClientContext *clientContext
);
```

Parameters

alloc

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

clientCB

Callback function that is to be called when domains and services are found; cannot be `NULL`. For details, see `CFNetServiceBrowserClientCallBack` (page 34).

clientContext

Context information to be used when `clientCB` is called; cannot be `NULL`. For details, see `CFNetServiceClientContext` (page 37).

Return Value

A new browser object, or `NULL` if the instance could not be created. Ownership follows the Create Rule.

Discussion

This function creates an instance of a Network Service browser object, called a CFNetServiceBrowser, that can be used to search for domains and for services.

To use the resulting CFNetServiceBrowser in asynchronous mode, call `CFNetServiceBrowserScheduleWithRunLoop` (page 10). Then call `CFNetServiceBrowserSearchForDomains` (page 10) and `CFNetServiceBrowserSearchForServices` (page 11) to use the CFNetServiceBrowser to search for services and domains, respectively. The callback function specified by `clientCB` is called from a run loop to pass search results to your application. The search continues until you stop the search by calling `CFNetServiceBrowserStopSearch` (page 12).

If you do not call `CFNetServiceBrowserScheduleWithRunLoop` (page 10), searches with the resulting CFNetServiceBrowser are made in synchronous mode. Calls made to `CFNetServiceBrowserSearchForDomains` (page 10) and `CFNetServiceBrowserSearchForServices` (page 11) block until there are search results, in which case the callback function specified by `clientCB` is called, until the search is are stopped by calling `CFNetServiceBrowserStopSearch` (page 12) from another thread, or an error occurs.

To shut down a CFNetServiceBrowser that is running in asynchronous mode, call `CFNetServiceBrowserUnscheduleFromRunLoop` (page 13), followed by `CFNetServiceBrowserInvalidate` (page 9), and then `CFNetServiceBrowserStopSearch` (page 12).

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceBrowserGetTypeID

Gets the Core Foundation type identifier for the Network Service browser object.

```
CFTypeID CFNetServiceBrowserGetTypeID ();
```

Return Value
The type ID.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceBrowserInvalidate

Invalidates an instance of a Network Service browser object.

```
void CFNetServiceBrowserInvalidate (
   CFNetServiceBrowserRef browser
);
```

Parameters

browser

The CFNetServiceBrowser to invalidate, obtained by a previous call to `CFNetServiceBrowserCreate` (page 8).

Discussion
This function invalidates the specified instance of a Network Service browser object. Any searches using the specified instance that are in progress when this function is called are stopped. An invalidated browser cannot be scheduled on a run loop and its callback function is never called.

Special Considerations

This function is thread safe as long as another thread does not alter the same CFNetServiceBrowserRef at the same time.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceBrowserScheduleWithRunLoop

Schedules a CFNetServiceBrowser on a run loop.

```
void CFNetServiceBrowserScheduleWithRunLoop (
   CFNetServiceBrowserRef browser,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

browser

The CFNetServiceBrowser that is to be scheduled on a run loop; cannot be `NULL`.

runLoop

The run loop on which the browser is to be scheduled; cannot be `NULL`.

runLoopMode

The mode on which to schedule the browser; cannot be `NULL`.

Discussion

This function schedules the specified CFNetServiceBrowser on the run loop, thereby placing the browser in asynchronous mode. The run loop will call the browser's callback function to deliver the results of domain and service searches. The caller is responsible for ensuring that at least one of the run loops on which the browser is scheduled is being run.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetServices.h`

CFNetServiceBrowserSearchForDomains

Searches for domains.

```
Boolean CFNetServiceBrowserSearchForDomains (
   CFNetServiceBrowserRef browser,
   Boolean registrationDomains,
   CFStreamError *error
);
```

Parameters

browser

The CFNetServiceBrowser, obtained by previously calling `CFNetServiceBrowserCreate` (page 8), that is to perform the search; cannot be `NULL`.

registrationDomains

`TRUE` to search for only registration domains; `FALSE` to search for domains that can be browsed for services. For this version of the CFNetServices API, the registration domain is the local domain maintained by the mDNS responder running on the same machine as the calling application.

error

A pointer to a `CFStreamError` structure, that, if an error occurs, will be set to the error and the error's domain and passed to your callback function. Pass `NULL` if you don't want to receive the error that may occur as a result of this particular call.

Return Value

`TRUE` if the search was started (asynchronous mode); `FALSE` if another search is already in progress for this CFNetServiceBrowser or if an error occurred.

Discussion

This function uses a CFNetServiceBrowser to search for domains. The search continues until the search is canceled by calling `CFNetServiceBrowserStopSearch` (page 12). If `registrationDomains` is `TRUE`, this function searches only for domains in which services can be registered. If `registrationDomains` is `FALSE`, this function searches for domains that can be browsed for services. When a domain is found, the callback function specified when the CFNetServiceBrowser was created is called and passed an instance of a CFStringRef containing the domain that was found.

In asynchronous mode, this function returns `TRUE` if the search was started. Otherwise, it returns `FALSE`.

In synchronous mode, this function blocks until the search is stopped by calling `CFNetServiceBrowserStopSearch` (page 12) from another thread, in which case it returns `FALSE`, or until an error occurs.

Special Considerations

This function is thread safe.

For any one CFNetServiceBrowser, only one domain search or one service search can be in progress at the same time.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetServices.h`

CFNetServiceBrowserSearchForServices

Searches a domain for services of a specified type.

```
Boolean CFNetServiceBrowserSearchForServices (
   CFNetServiceBrowserRef browser,
   CFStringRef domain,
   CFStringRef serviceType,
   CFStreamError *error
);
```

Parameters

browser

The CFNetServiceBrowser, obtained by previously calling `CFNetServiceBrowserCreate` (page 8), that is to perform the search; cannot be `NULL`.

domain

The domain to search for the service type; cannot be `NULL`. To get the domains that are available for searching, call `CFNetServiceBrowserSearchForDomains` (page 10).

type

The service type to search for; cannot be `NULL`. For a list of valid service types, see http://www.iana.org/assignments/port-numbers.

error

A pointer to a `CFStreamError` structure, that, if an error occurs, will be set to the error and the error's domain and passed to your callback function. Pass `NULL` if you don't want to receive the error that may occur as a result of this particular call.

Return Value

`TRUE` if the search was started (asynchronous mode); `FALSE` if another search is already in progress for this CFNetServiceBrowser or if an error occurred.

Discussion

This function searches the specified domain for services that match the specified service type. The search continues until the search is canceled by calling `CFNetServiceBrowserStopSearch` (page 12). When a match is found, the callback function specified when the CFNetServiceBrowser was created is called and passed an instance of a CFNetService representing the service that was found.

In asynchronous mode, this function returns `TRUE` if the search was started. Otherwise, it returns `FALSE`.

In synchronous mode, this function blocks until the search is stopped by calling `CFNetServiceBrowserStopSearch` (page 12) from another thread, in which case this function returns `FALSE`, or until an error occurs.

Special Considerations

This function is thread safe.

For any one CFNetServiceBrowser, only one domain search or one service search can be in progress at the same time.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetServices.h`

CFNetServiceBrowserStopSearch

Stops a search for domains or services.

```
void CFNetServiceBrowserStopSearch (
   CFNetServiceBrowserRef browser,
   CFStreamError *error
);
```

Parameters

browser

The CFNetServiceBrowser that was used to start the search; cannot be `NULL`.

error

A pointer to a `CFStreamError` structure that will be passed to the callback function associated with this CFNetServiceBrowser (if the search is being conducted in asynchronous mode) or that is pointed to by the `error` parameter when `CFNetServiceBrowserSearchForDomains` (page 10) or `CFNetServiceBrowserSearchForServices` (page 11) returns (if the search is being conducted in synchronous mode). Set the `domain` field to `kCFStreamErrorDomainCustom` and the `error` field to an appropriate value.

Discussion

This functions stops a search started by a previous call to `CFNetServiceBrowserSearchForDomains` (page 10) or `CFNetServiceBrowserSearchForServices` (page 11). For asynchronous and synchronous searches, calling this function causes the callback function associated with the CFNetServiceBrowser to be called once for each domain or service found. If the search is asynchronous, `error` is passed to the callback function. If the search is synchronous, calling this function causes `CFNetServiceBrowserSearchForDomains` or `CFNetServiceBrowserSearchForServices` to return `FALSE`. If the `error` parameter for either call pointed to a `CFStreamError` structure, the `CFStreamError` structure contains the error code and the error code's domain as set when this function was called.

Special Considerations

This function is thread safe.

If you are stopping an asynchronous search, before calling this function, call `CFNetServiceBrowserUnscheduleFromRunLoop` (page 13), followed by `CFNetServiceBrowserInvalidate` (page 9).

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetServices.h`

CFNetServiceBrowserUnscheduleFromRunLoop

Unschedules a CFNetServiceBrowser from a run loop and mode.

```
void CFNetServiceBrowserUnscheduleFromRunLoop (
   CFNetServiceBrowserRef browser,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

browser

The CFNetServiceBrowser that is to be unscheduled; cannot be `NULL`.

runLoop

The run loop; cannot be `NULL`.

runLoopMode

The mode from which the browser is to be unscheduled; cannot be `NULL`.

Discussion

Call this function to shut down a browser that is running asynchronously. To complete the shutdown, call `CFNetServiceBrowserInvalidate` (page 9) followed by `CFNetServiceBrowserStopSearch` (page 12).

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceCancel

Cancels a service registration or a service resolution.

```
void CFNetServiceCancel (
   CFNetServiceRef theService
);
```

Parameters

theService

The CFNetService, obtained by previously calling `CFNetServiceCreate` (page 14), for which a registration or a resolution is to be canceled.

Discussion
This function cancels service registrations, started by `CFNetServiceRegister` (page 27), thereby making the service unavailable. It also cancels service resolutions, started by `CFNetServiceResolve` (page 29).

If you are shutting down an asynchronous service, you should first call `CFNetServiceUnscheduleFromRunLoop` (page 34) and `CFNetServiceSetClient` (page 32) with `clientCB` set to `NULL`. Then call this function.

If you are shutting down a synchronous service, call this function from another thread.

This function also cancels service resolutions. You would want to cancel a service resolution if your callback function has received an IP address that you've successfully used to connect to the service. In addition, you might want to cancel a service resolution if the resolution is taking longer than a user would want to wait or if the user canceled the operation.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceCreate

Creates an instance of a Network Service object.

```
CFNetServiceRef CFNetServiceCreate (
   CFAllocatorRef alloc,
   CFStringRef domain,
   CFStringRef serviceType,
   CFStringRef name,
   SInt32 port
);
```

Parameters

alloc

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

domain

The domain in which the CFNetService is to be registered; cannot be `NULL`. Call `CFNetServiceBrowserCreate` (page 8) and `CFNetServiceBrowserSearchForDomains` (page 10) to get the registration domain.

type

The type of service being registered; cannot be `NULL`. For a list of valid service types, see http://www.iana.org/assignments/port-numbers.

name

A unique name if the instance will be used to register a service. The name will become part of the instance name in the DNS records that will be created when the service is registered. If the instance will be used to resolve a service, the name should be the name of the machine or service that will be resolved.

port

Local IP port, in host byte order, on which this service accepts connections. Pass zero to get placeholder service. With a placeholder service, the service will not be discovered by browsing, but a name conflict will occur if another client tries to register the same name. Most applications do not need to use placeholder service.

Return Value

A new net service object, or `NULL` if the instance could not be created. Ownership follows the Create Rule.

Discussion

If the service depends on information in DNS TXT records, call `CFNetServiceSetProtocolSpecificInformation` (page 32).

If the CFNetService is to run in asynchronous mode, call `CFNetServiceSetClient` (page 32) to prepare the service for running in asynchronous mode. Then call `CFNetServiceScheduleWithRunLoop` (page 31) to schedule the service on a run loop. Then call `CFNetServiceRegister` (page 27) to make the service available.

If the CFNetService is to run in synchronous mode, call `CFNetServiceRegister` (page 27).

To terminate a service that is running in asynchronous mode, call `CFNetServiceCancel` (page 14) and `CFNetServiceUnscheduleFromRunLoop` (page 34).

To terminate a service that is running in synchronous mode, call `CFNetServiceCancel` (page 14).

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.2 and later.

Declared In
CFNetServices.h

CFNetServiceCreateCopy

Creates a copy of a CFNetService object.

```
CFNetServiceRef CFNetServiceCreateCopy (
   CFAllocatorRef alloc,
   CFNetServiceRef service
);
```

Parameters

alloc

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

service

CFNetServiceRef to be copied; cannot be NULL. If service is not a valid CFNetServiceRef, the behavior of this function is undefined.

Return Value
Copy of service, including all previously resolved data, or NULL if service could not be copied. Ownership follows the Create Rule.

Discussion
This function creates a copy of the CFNetService specified by service.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.3 and later.

Declared In
CFNetServices.h

CFNetServiceCreateDictionaryWithTXTData

Uses TXT record data to create a dictionary.

```
CFDictionaryRef CFNetServiceCreateDictionaryWithTXTData (
   CFAllocatorRef alloc,
   CFDataRef txtRecord
);
```

Parameters

alloc

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

txtRecord

TXT record data as returned by CFNetServiceGetTXTData (page 21).

Return Value
A dictionary containing the key/value pairs parsed from `txtRecord`, or `NULL` if `txtRecord` cannot be parsed. Each key in the dictionary is a CFString object, and each value is a CFData object. Ownership follows the Create Rule.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceCreateTXTDataWithDictionary

Flattens a set of key/value pairs into a CFDataRef suitable for passing to `CFNetServiceSetTXTData` (page 33).

```
CFDataRef CFNetServiceCreateTXTDataWithDictionary (
   CFAllocatorRef alloc,
   CFDictionaryRef keyValuePairs
);
```

Parameters

alloc

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

keyValuePairs

CFDictionaryRef containing the key/value pairs that are to be placed in a TXT record. Each key must be a CFStringRef and each value should be a CFDataRef or a CFStringRef. (See the discussion below for additional information about values that are CFStringRefs.) This function fails if any other data types are provided. The length of a key and its value should not exceed 255 bytes.

Return Value
A CFData object containing the flattened form of *keyValuePairs*, or `NULL` if the dictionary could not be flattened. Ownership follows the Create Rule.

Discussion
This function flattens the key/value pairs in the dictionary specified by `keyValuePairs` into a CFDataRef suitable for passing to `CFNetServiceSetTXTData` (page 33). Note that this function is not a general purpose function for flattening CFDictionaryRefs.

The keys in the dictionary referenced by `keyValuePairs` must be CFStringRefs and the values must be CFDataRefs. Any values that are CFStringRefs are converted to CFDataRefs representing the flattened UTF-8 bytes of the string. The types of the values are not encoded in the CFDataRefs, so any CFStringRefs that are converted to CFDataRefs remain CFDataRefs when the CFDataRef produced by this function is processed by `CFNetServiceCreateDictionaryWithTXTData` (page 16).

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CFNetServices.h

CFNetServiceGetAddressing

Gets the IP addressing from a CFNetService.

```
CFArrayRef CFNetServiceGetAddressing (
   CFNetServiceRef theService
);
```

Parameters

theService
The CFNetService whose IP addressing is to be obtained; cannot be NULL.

Return Value
A CFArray containing a CFDataRef for each IP address returned, or NULL. Each CFDataRef consists of a sockaddr structure containing the IP address of the service. This function returns NULL if the service's addressing is unknown because CFNetServiceResolve (page 29) has not been called for theService.

Discussion
This function gets the IP addressing from a CFNetService. Typically, the CFNetService was obtained by calling CFNetServiceBrowserSearchForServices (page 11). Before calling this function, call CFNetServiceResolve (page 29) to update the CFNetService with its IP addressing.

Special Considerations

This function gets the data in a thread-safe way, but the data itself is not safe if the service is altered from another thread.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
CFNetServices.h

CFNetServiceGetDomain

Gets the domain from a CFNetService.

```
CFStringRef CFNetServiceGetDomain (
   CFNetServiceRef theService
);
```

Parameters

theService
The CFNetService whose domain is to be obtained; cannot be NULL.

Return Value
A CFString object containing the domain of the CFNetService.

Discussion
This function gets the domain from a CFNetService.

Special Considerations

This function is thread safe. The function gets the data in a thread-safe way, but the data is not safe if the service is altered from another thread.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceGetName

Gets the name from a CFNetService.

```
CFStringRef CFNetServiceGetName (
   CFNetServiceRef theService
);
```

Parameters

`theService`

The CFNetService whose name is to be obtained; cannot be `NULL`.

Return Value
A CFString object containing the name of the service represented by the CFNetService.

Discussion
This function gets the name from a CFNetService.

Special Considerations

This function is thread safe. The function gets the data in a thread-safe way, but the data is not safe if the service is altered from another thread.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceGetPortNumber

This function gets the port number from a CFNetService.

```
extern SInt32 CFNetServiceGetPortNumber(
    CFNetServiceRef theService);
```

Parameters

`theService`

The CFNetService whose protocol-specific information is to be obtained; cannot be `NULL`. Note that in order to get protocol-specific information, you must resolve `theService` by calling `CFNetServiceResolve` (page 29) or `CFNetServiceResolveWithTimeout` (page 30) before calling this function.

Return Value
A CFString object containing the protocol-specific information, or `NULL` if there is no information.

Special Considerations

This function gets the data in a thread-safe way, but the data itself is not safe if the service is altered from another thread.

Availability

Available in Mac OS X version 10.2 and later.

Deprecated in Mac OS X version 10.4.

Declared In

`CFNetServices.h`

CFNetServiceGetProtocolSpecificInformation

This function gets protocol-specific information from a CFNetService. (**Deprecated.** Use `CFNetServiceGetTXTData` (page 21) instead.)

```
CFStringRef CFNetServiceGetProtocolSpecificInformation (
   CFNetServiceRef theService
);
```

Parameters

`theService`

The CFNetService whose protocol-specific information is to be obtained; cannot be `NULL`. Note that in order to get protocol-specific information, you must resolve `theService` by calling `CFNetServiceResolve` (page 29) or `CFNetServiceResolveWithTimeout` (page 30) before calling this function.

Return Value

A CFString object containing the protocol-specific information, or `NULL` if there is no information.

Special Considerations

This function gets the data in a thread-safe way, but the data itself is not safe if the service is altered from another thread.

Availability

Available in Mac OS X version 10.2 and later.

Deprecated in Mac OS X version 10.4.

Declared In

`CFNetServices.h`

CFNetServiceGetTargetHost

Queries a CFNetService for its target hosts.

```
CFStringRef CFNetServiceGetTargetHost (
   CFNetServiceRef theService
);
```

Parameters

`theService`

Network service to be queried.

Return Value
The target host name of the machine providing the service or `NULL` is of the service's target host is not known. (The target host will not be known if it has not been resolved.)

Special Considerations

This function is thread safe, but the target host name is not safe if the service is altered from another thread.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceGetTXTData

Queries a network service for the contents of its TXT records.

```
CFDataRef CFNetServiceGetTXTData (
   CFNetServiceRef theService
);
```

Parameters

`theService`

Reference for the network service whose TXT record data is to be obtained; cannot be `NULL`. Note that in order to get TXT record data, you must resolve `theService` by calling `CFNetServiceResolve` (page 29) or `CFNetServiceResolveWithTimeout` (page 30) before calling this function.

Return Value
CFDataRef object containing the requested TXT data and suitable for passing to `CFNetServiceCreateDictionaryWithTXTData` (page 16), or `NULL` if the service's TXT data has not been resolved.

Discussion
This function gets the data from the service's TXT records.

Special Considerations

This function gets the data in a thread-safe way, but the data itself is not safe if the service is altered from another thread.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceGetType

Gets the type from a CFNetService.

```
CFStringRef CFNetServiceGetType (
   CFNetServiceRef theService
);
```

Parameters

theService

The CFNetService whose type is to be obtained; cannot be `NULL`.

Return Value
A CFString object containing the type from a CFNetService.

Discussion
This function gets the type of a CFNetService.

Special Considerations

This function is thread safe. The function gets the data in a thread-safe way, but the data is not safe if the service is altered from another thread.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceGetTypeID

Gets the Core Foundation type identifier for the Network Service object.

```
CFTypeID CFNetServiceGetTypeID ();
```

Return Value
The type ID.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceMonitorCreate

Creates an instance of a NetServiceMonitor object that watches for record changes.

```
CFNetServiceMonitorRef CFNetServiceMonitorCreate (
   CFAllocatorRef alloc,
   CFNetServiceRef theService,
   CFNetServiceMonitorClientCallBack clientCB,
   CFNetServiceClientContext *clientContext
);
```

Parameters

alloc

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

theService

CFNetService to be monitored.

clientCB

Pointer to callback function that is to be called when a record associated with `theService` changes; cannot be `NULL`.

clientContext

Pointer to user-defined contextual information that is to be passed to the callback specified by `clientCB` when the callback is called; cannot be `NULL`. For details, see CFNetServiceClientContext (page 37).

Return Value

A new instance of a CFNetServiceMonitor, or `NULL` if the monitor could not be created. Ownership follows the Create Rule.

Discussion

This function creates a CFNetServiceMonitor that watches for changes in records associated with `theService`.

If the CFNetServiceMonitor is to run in asynchronous mode, call `CFNetServiceMonitorScheduleWithRunLoop` (page 24) to schedule the monitor on a run loop. Then call `CFNetServiceMonitorStart` (page 25) to start monitoring. When a change occurs, the callback function specified by `clientCB` will be called. For details, see CFNetServiceMonitorClientCallBack (page 36).

If the CFNetServiceMonitor is to run in synchronous mode, call `CFNetServiceMonitorStart` (page 25).

To stop a monitor that is running in asynchronous mode, call `CFNetServiceMonitorStop` (page 26) and `CFNetServiceMonitorUnscheduleFromRunLoop` (page 27).

To stop a monitor that is running in synchronous mode, call `CFNetServiceMonitorStop` (page 26).

If you no longer need to monitor record changes, call `CFNetServiceMonitorStop` (page 26) to stop the monitor and then call `CFNetServiceMonitorInvalidate` (page 24)to invalidate the monitor so it cannot be used again. Then call `CFRelease` to release the memory associated with CFNetServiceMonitorRef.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CFNetServices.h`

CFNetServiceMonitorGetTypeID

Gets the Core Foundation type identifier for all CFNetServiceMonitor instances.

```
CFTypeID CFNetServiceMonitorGetTypeID ();
```

Return Value
The type ID.

Special Considerations

This function is thread safe.

Version Notes
Introduced in Mac OS X v10.4.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceMonitorInvalidate

Invalidates an instance of a Network Service monitor object.

```
void CFNetServiceMonitorInvalidate (
   CFNetServiceMonitorRef monitor
);
```

Parameters

monitor

CFNetServiceMonitor to invalidate; cannot be `NULL`.

Discussion
This function invalidates the specified Network Service monitor so that it cannot be used again. Before you call this function, you should call `CFNetServiceMonitorStop` (page 26). If the monitor has not already been stopped, this function stops the monitor for you.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceMonitorScheduleWithRunLoop

Schedules a CFNetServiceMonitor on a run loop.

```
void CFNetServiceMonitorScheduleWithRunLoop (
   CFNetServiceMonitorRef monitor,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

theService

The CFNetServiceMonitor that is to be scheduled on a run loop; cannot be `NULL`.

runLoop

The run loop on which the monitor is to be scheduled; cannot be `NULL`.

runLoopMode

The mode on which to schedule the monitor; cannot be `NULL`.

Discussion

Schedules the specified monitor on a run loop, which places the monitor in asynchronous mode. The caller is responsible for ensuring that at least one of the run loops on which the monitor is scheduled is being run.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.4 and later.

Declared In

`CFNetServices.h`

CFNetServiceMonitorStart

Starts monitoring.

```
Boolean CFNetServiceMonitorStart (
   CFNetServiceMonitorRef monitor,
   CFNetServiceMonitorType recordType,
   CFStreamError *error
);
```

Parameters

monitor

CFNetServiceMonitor, created by calling `CFNetServiceMonitorCreate` (page 22), that is to be started.

recordType

CFNetServiceMonitorType that specified the type of record to monitor. For possible values, see CFNetServiceMonitorType Constants (page 40).

error

Pointer to a `CFStreamError` structure. If an error occurs, on output, the structure's `domain` field will be set to the error code's domain and the `error` field will be set to an appropriate error code. Set this parameter to `NULL` if you don't want to receive the error code and its domain.

Return Value

`TRUE` if an asynchronous monitor was started successfully. `FALSE` if an error occurred when starting an asynchronous or synchronous monitor, or if `CFNetServiceMonitorStop` (page 26) was called for an synchronous monitor.

Discussion
This function starts monitoring for changes to records of the type specified by `recordType`. If a monitor is already running for the service associated with the specified CFNetServiceMonitorRef, this function returns `FALSE`.

For synchronous monitors, this function blocks until the monitor is stopped by calling `CFNetServiceMonitorStop` (page 26), in which case, this function returns `FALSE`.

For asynchronous monitors, this function returns `TRUE` or `FALSE`, depending on whether monitoring starts successfully.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceMonitorStop

Stops a CFNetServiceMonitor.

```
void CFNetServiceMonitorStop (
   CFNetServiceMonitorRef monitor,
   CFStreamError *error
);
```

Parameters

monitor

CFNetServiceMonitor, started by calling `CFNetServiceMonitorStart` (page 25), that is to be stopped.

error

Pointer to a `CFStreamError` structure or `NULL`. For synchronous monitors, set the `error` field of this structure to the non-zero value you want to be set in the `CFStreamError` structure when `CFNetServiceMonitorStart` (page 25) returns. Note that when it returns, `CFNetServiceMonitorStart` returns `FALSE`. If the monitor was started asynchronously, set the `error` field to the non-zero value you want the monitor's callback to receive when it is called. If this parameter is `NULL`, default values for the `CFStreamError` structure are used: the domain is set to `kCFStreamErrorDomainNetServices` and the error code is set to `kCFNetServicesErrorCancel`.

Discussion
This function stops the specified monitor. Call `CFNetServiceMonitorStart` (page 25) if you want to start monitoring again.

If you want to stop monitoring and no longer need to monitor record changes, call `CFNetServiceMonitorInvalidate` (page 24) instead of this function.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceMonitorUnscheduleFromRunLoop

Unschedules a CFNetServiceMonitor from a run loop.

```
void CFNetServiceMonitorUnscheduleFromRunLoop (
   CFNetServiceMonitorRef monitor,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

monitor
The CFNetServiceMonitor that is to be unscheduled; cannot be `NULL`.

runLoop
The run loop; cannot be `NULL`.

runLoopMode
The mode from which the monitor is to be unscheduled; cannot be `NULL`.

Discussion
Unschedules the specified monitor from the specified run loop and mode. Call this function to shut down a monitor that is running asynchronously.

To change a monitor so that it cannot be scheduled and so that its callback will never be called, call `CFNetServiceMonitorInvalidate` (page 24).

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceRegister

Makes a CFNetService available on the network. (**Deprecated.** Use `CFNetServiceRegisterWithOptions` (page 28) instead.)

```
Boolean CFNetServiceRegister (
   CFNetServiceRef theService,
   CFStreamError *error
);
```

Parameters

theService
The CFNetService to register; cannot be `NULL`. The registration will fail if the service doesn't have a domain, a type, a name, and an IP address.

error

A pointer to a `CFStreamError` structure that will be set to an error code and the error code's domain if an error occurs; or `NULL` if you don't want to receive the error code and its domain.

Return Value

`TRUE` if an asynchronous service registration was started; `FALSE` if an asynchronous or synchronous registration failed or if a synchronous registration was canceled.

Discussion

If the service is to run in asynchronous mode, you must call `CFNetServiceSetClient` (page 32) to associate a callback function with this CFNetService before calling this function.

When registering a service that runs in asynchronous mode, this function returns `TRUE` if the service contains all of the required attributes and the registration process can start. If the registration process completes successfully, the service is available on the network until you shut down the service by calling `CFNetServiceUnscheduleFromRunLoop` (page 34), `CFNetServiceSetClient` (page 32), and `CFNetServiceCancel` (page 14). If the service does not contain all of the required attributes or if the registration process does not complete successfully, this function returns `FALSE`.

When registering a service that runs in synchronous mode, this function blocks until an error occurs, in which case this function returns `FALSE`. Until this function returns `FALSE`, the service is available on the network. To force this function to return `FALSE`, thereby shutting down the service, call `CFNetServiceCancel` (page 14) from another thread.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.2 and later.

Deprecated in Mac OS X version 10.4.

Declared In

`CFNetServices.h`

CFNetServiceRegisterWithOptions

Makes a CFNetService available on the network.

```
Boolean CFNetServiceRegisterWithOptions (
   CFNetServiceRef theService,
   CFOptionFlags options,
   CFStreamError *error
);
```

Parameters

theService

Network service to register; cannot be `NULL`. The registration will fail if the service doesn't have a domain, a type, a name, and an IP address.

options

Bit flags for specifying registration options. Currently, the only registration option is `kCFNetServiceFlagNoAutoRename`. For details, see CFNetService Registration Options (page 39).

error

Pointer to a `CFStreamError` structure that will be set to an error code and the error code's domain if an error occurs; or `NULL` if you don't want to receive the error code and its domain.

Return Value
`TRUE` if an asynchronous service registration was started; `FALSE` if an asynchronous or synchronous registration failed or if a synchronous registration was canceled.

Discussion
If the service is to run in asynchronous mode, you must call `CFNetServiceSetClient` (page 32) to associate a callback function with this CFNetService before calling this function.

When registering a service that runs in asynchronous mode, this function returns `TRUE` if the service contains all of the required attributes and the registration process can start. If the registration process completes successfully, the service is available on the network until you shut down the service by calling `CFNetServiceUnscheduleFromRunLoop` (page 34), `CFNetServiceSetClient` (page 32), and `CFNetServiceCancel` (page 14). If the service does not contain all of the required attributes or if the registration process does not complete successfully, this function returns `FALSE`.

When registering a service that runs in synchronous mode, this function blocks until an error occurs, in which case this function returns `FALSE`. Until this function returns `FALSE`, the service is available on the network. To force this function to return `FALSE`, thereby shutting down the service, call `CFNetServiceCancel` (page 14) from another thread.

The `options` parameter is a bit flag for specifying service registration options. Currently, `kCFNetServiceFlagNoAutoRename` is the only supported registration option. If this bit is set and a service of the same name is running, the registration will fail. If this bit is not set and a service of the same name is running, the service that is being registered will be renamed automatically by appending `(n)` to the service name, where *n* is a number that is incremented until the service can be registered with a unique name.

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceResolve

This function updates the specified CFNetService with the IP address or addresses associated with the service. Call `CFNetServiceGetAddressing` (page 18) to get the addresses. (**Deprecated.** Use `CFNetServiceResolveWithTimeout` (page 30) instead.)

```
Boolean CFNetServiceResolve (
   CFNetServiceRef theService,
   CFStreamError *error
);
```

Parameters

theService

The CFNetService to resolve; cannot be `NULL`. The resolution will fail if the service doesn't have a domain, a type, and a name.

error

A pointer to a `CFStreamError` structure that will be set to an error code and the error code's domain if an error occurs; or `NULL` if you don't want to receive the error code and its domain.

Return Value

`TRUE` if an asynchronous service resolution was started or if a synchronous service resolution updated the CFNetService; `FALSE` if an asynchronous or synchronous resolution failed or if a synchronous resolution was canceled.

Discussion

When resolving a service that runs in asynchronous mode, this function returns `TRUE` if the CFNetService has a domain, type, and name, and the underlying resolution process was started. Otherwise, this function returns `FALSE`. Once started, the resolution continues until it is canceled by calling `CFNetServiceCancel` (page 14).

When resolving a service that runs in synchronous mode, this function blocks until the CFNetService is updated with at least one IP address, until an error occurs, or until `CFNetServiceCancel` (page 14) is called.

Special Considerations

This function is thread safe.

If the service will be used in asynchronous mode, you must call `CFNetServiceSetClient` (page 32) before calling this function.

Availability

Available in Mac OS X version 10.2 and later.

Deprecated in Mac OS X version 10.4.

Declared In

`CFNetServices.h`

CFNetServiceResolveWithTimeout

Gets the IP address or addresses for a CFNetService.

```
Boolean CFNetServiceResolveWithTimeout (
   CFNetServiceRef theService,
   CFTimeInterval timeout,
   CFStreamError *error
);
```

Parameters

theService

The CFNetService to resolve; cannot be `NULL`. The resolution will fail if the service doesn't have a domain, a type, and a name.

timeout

Value of type `CFTimeInterval` specifying the maximum amount of time allowed to perform the resolution. If the resolution is not performed within the specified amount of time, a timeout error will be returned. If `timeout` is less than or equal to zero, an infinite amount of time is allowed.

error

Pointer to a `CFStreamError` structure that will be set to an error code and the error code's domain if an error occurs; or `NULL` if you don't want to receive the error code and its domain.

Return Value
`TRUE` if an asynchronous service resolution was started or if a synchronous service resolution updated the CFNetService; `FALSE` if an asynchronous or synchronous resolution failed or timed out, or if a synchronous resolution was canceled.

Discussion
This function updates the specified CFNetService with the IP address or addresses associated with the service. Call `CFNetServiceGetAddressing` (page 18) to get the addresses.

When resolving a service that runs in asynchronous mode, this function returns `TRUE` if the CFNetService has a domain, type, and name, and the underlying resolution process was started. Otherwise, this function returns `FALSE`. Once started, the resolution continues until it is canceled by calling `CFNetServiceCancel` (page 14).

When resolving a service that runs in synchronous mode, this function blocks until the CFNetService is updated with at least one IP address, until an error occurs, or until `CFNetServiceCancel` (page 14) is called.

Special Considerations

This function is thread safe.

If the service will be used in asynchronous mode, you must call `CFNetServiceSetClient` (page 32) before calling this function.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceScheduleWithRunLoop

Schedules a CFNetService on a run loop.

```
void CFNetServiceScheduleWithRunLoop (
   CFNetServiceRef theService,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

theService
The CFNetService that is to be scheduled on a run loop; cannot be `NULL`.

runLoop
The run loop on which the service is to be scheduled; cannot be `NULL`.

runLoopMode
The mode on which to schedule the service; cannot be `NULL`.

Discussion
Schedules the specified service on a run loop, which places the service in asynchronous mode. The caller is responsible for ensuring that at least one of the run loops on which the service is scheduled is being run.

Special Considerations

This function is thread safe.

Before calling this function, call `CFNetServiceSetClient` (page 32) to prepare a CFNetService for use in asynchronous mode.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceSetClient

Associates a callback function with a CFNetService or disassociates a callback function from a CFNetService.

```
Boolean CFNetServiceSetClient (
   CFNetServiceRef theService,
   CFNetServiceClientCallBack clientCB,
   CFNetServiceClientContext *clientContext
);
```

Parameters

`theService`
The CFNetService; cannot be `NULL`.

`clientCB`
The callback function that is to be associated with this CFNetService. If you are shutting down the service, set `clientCB` to `NULL` to disassociate from this CFNetService the callback function that was previously associated.

`clientContext`
Context information to be used when `clientCB` is called; cannot be `NULL`.

Return Value
`TRUE` if the client was set; otherwise, `FALSE`.

Discussion
The callback function specified by `clientCB` will be called to report IP addresses (in the case of `CFNetServiceResolve`) or to report registration errors (in the case of `CFNetServiceRegister`).

Special Considerations

This function is thread safe.

For a CFNetService that will operate asynchronously, call this function and then call `CFNetServiceScheduleWithRunLoop` (page 31) to schedule the service on a run loop. Then call `CFNetServiceRegister` (page 27) or `CFNetServiceResolve` (page 29).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceSetProtocolSpecificInformation

Sets protocol-specific information for a CFNetService. (**Deprecated.** Use `CFNetServiceSetTXTData` instead.)

```
void CFNetServiceSetProtocolSpecificInformation (
   CFNetServiceRef theService,
   CFStringRef theInfo
);
```

Parameters

theService

The CFNetService whose protocol-specific information is to be set; cannot be `NULL`.

theInfo

The protocol-specific information to be set. Pass `NULL` to remove protocol-specific information from the service.

Discussion

The protocol-specific information appears in DNS TXT records for the service. Each TXT record consists of zero or more strings, packed together without any intervening gaps or padding bytes for word alignment. The format of each constituent string is a single length byte, followed by zero to 255 bytes of text data.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.2 and later.

Deprecated in Mac OS X version 10.4.

Declared In

`CFNetServices.h`

CFNetServiceSetTXTData

Sets the TXT record for a CFNetService.

```
Boolean CFNetServiceSetTXTData (
   CFNetServiceRef theService,
   CFDataRef txtRecord
);
```

Parameters

theService

CFNetServiceRef for which a TXT record is to be set; cannot be `NULL`.

txtRecord

Contents of the TXT record that is to be set. The contents must not exceed 1450 bytes.

Return Value

`TRUE` if the TXT record was set; otherwise, `FALSE`.

Discussion

This function sets a TXT record for the specified service. If the service is currently registered on the network, the record is broadcast. Setting a TXT record on a service that is still being resolved is not allowed.

Special Considerations

This function is thread safe.

Availability

Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceUnscheduleFromRunLoop

Unschedules a CFNetService from a run loop.

```
void CFNetServiceUnscheduleFromRunLoop (
   CFNetServiceRef theService,
   CFRunLoopRef runLoop,
   CFStringRef runLoopMode
);
```

Parameters

theService
The CFNetService that is to be unscheduled; cannot be `NULL`.

runLoop
The run loop; cannot be `NULL`.

runLoopMode
The mode from which the service is to be unscheduled; cannot be `NULL`.

Discussion
Unschedules the specified service from the specified run loop and mode. Call this function to shut down a service that is running asynchronously. To complete the shutdown, call `CFNetServiceSetClient` (page 32) and set `clientCB` to `NULL`. Then call `CFNetServiceCancel` (page 14).

Special Considerations

This function is thread safe.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

Callbacks

CFNetServiceBrowserClientCallBack

Defines a pointer to the callback function for a CFNetServiceBrowser.

```
typedef void (*CFNetServiceBrowserClientCallBack) (
   CFNetServiceBrowserRef browser,
   CFOptionFlags flags,
   CFTypeRef domainOrService,
   CFStreamError* error,
   void* info);
```

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

```
void MyNetServiceBrowserClientCallBack (
```

```
    CFNetServiceBrowserRef browser,
    CFOptionFlags flags,
    CFTypeRef domainOrService,
    CFStreamError* error,
    void* info);
```

Parameters

browser

The CFNetServiceBrowser associated with this callback function.

flags

Flags conveying additional information. The `kCFNetServiceFlagIsDomain` bit is set if `domainOrService` contains a domain; if this bit is not set, `domainOrService` contains a CFNetService instance. For additional bit values, see CFNetServiceBrowserClientCallBack Bit Flags (page 39).

domainOrService

A string containing a domain name if this callback function is being called as a result of calling `CFNetServiceBrowserSearchForDomains` (page 10), or a CFNetService instance if this callback function is being called as a result calling `CFNetServiceBrowserSearchForServices` (page 11).

error

A pointer to a `CFStreamError` structure whose `error` field may contain an error code.

info

User-defined context information. The value of `info` is the same as the value of the `info` field of the `CFNetServiceClientContext` (page 37) structure that was provided when `CFNetServiceBrowserCreate` (page 8) was called to create the CFNetServiceBrowser associated with this callback function.

Discussion

The callback function for a CFNetServiceBrowser is called one or more times when domains or services are found as the result of calling `CFNetServiceBrowserSearchForDomains` (page 10) and `CFNetServiceBrowserSearchForServices` (page 11).

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetServices.h`

CFNetServiceClientCallBack

Defines a pointer to the callback function for a CFNetService.

```
typedef void (*CFNetServiceClientCallBack) (
    CFNetServiceRef theService,
    CFStreamError* error,
    void* info);
```

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

```
void MyNetServiceClientCallBack (
    CFNetServiceRef theService,
    CFStreamError* error,
```

```
   void* info);
```

Parameters

theService

CFNetService associated with this callback function.

error

Pointer to a `CFStreamError` structure whose `error` field contain may contain an error code.

info

User-defined context information. The value of `info` is the same as the value of the `info` field of the `CFNetServiceClientContext` (page 37) structure that was provided when `CFNetServiceSetClient` (page 32) was called for the CFNetService associated with this callback function.

Discussion

Your callback function will be called when there are results of resolving a CFNetService to report or when there are registration errors to report. In the case of resolution, if the service has more than one IP address, your callback will be called once for each address.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetServices.h`

CFNetServiceMonitorClientCallBack

Defines a pointer to the callback function that is to be called when a monitored record type changes.

```
typedef void (*CFNetServiceMonitorClientCallBack) (
   CFNetServiceMonitorRef theMonitor,
   CFNetServiceRef theService,
   CFNetServiceMonitorType typeInfo,
   CFDataRef rdata,
   CFStreamError* error,
   void* info);
```

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

```
void MyNetServiceMonitorClientCallBack (
   CFNetServiceMonitorRef theMonitor,
   CFNetServiceRef theService,
   CFNetserviceMonitorType typeInfo,
   CFDataRef rdata,
   CFStreamError *error,
   void *info);
```

Parameters

theMonitor

CFNetServiceMonitor for which the callback is being called.

theService

CFNetService for which the callback is being called.

typeInfo
Type of record that changed. For possible values, see CFNetServiceMonitorType Constants (page 40).

rdata
Contents of the record that changed.

error
Pointer to `CFStreamError` structure whose `error` field contains an error code if an error occurred.

info
Arbitrary pointer to the user-defined data that was specified in the `info` field of the `CFNetServiceClientContext` structure when the monitor was created by `CFNetServiceMonitorCreate` (page 22).

Discussion
The callback function will be called when the monitored record type changes or when the monitor is stopped by calling `CFNetServiceMonitorStop` (page 26).

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

Data Types

CFNetServiceBrowserRef

An opaque reference representing a CFNetServiceBrowser.

```
typedef struct __CFNetServiceBrowser* CFNetServiceBrowserRef;
```

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceClientContext

A structure provided when a CFNetService is associated with a callback function or when a CFNetServiceBrowser is created.

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

Fields

`version`

Version number for this structure. Currently the only valid value is zero.

`info`

Arbitrary pointer to user-allocated memory containing user-defined data that is associated with the service, browser, or monitor and is passed to their respective callback functions. The data must be valid for as long as the CFNetService, CFNetServiceBrowser, or CFNetServiceMonitor is valid. Set this field to `NULL` if your callback function doesn't want to receive user-defined data.

`retain`

The callback used to add a retain for the service or browser using `info` for the life of the service or browser. This callback may be used for temporary references the service or browser needs to take. This callback returns the actual `info` pointer so it can be stored in the service or browser. This field can be `NULL`.

`release`

Callback that removes a retain previously added for the service or browser on the `info` pointer. This field can be `NULL`, but setting this field to `NULL` may result in memory leaks.

`copyDescription`

Callback used to create a descriptive string representation of the data pointed to by `info`. In implementing this function, return a reference to a CFString object that describes your allocator and some characteristics of your user-defined data, which is used by `CFCopyDescription()`. You can set this field to `NULL`, in which case Core Foundation will provide a rudimentary description.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

CFNetServiceMonitorRef

An opaque reference for a service monitor.

```
typedef struct __CFNetServiceMonitor* CFNetServiceMonitorRef;
```

Discussion
Service monitor references are used to monitor record changes on a CFNetServiceRef.

Availability
Available in Mac OS X version 10.4 and later.

Declared In
`CFNetServices.h`

CFNetServiceRef

An opaque reference representing a CFNetService.

```
typedef struct __CFNetService* CFNetServiceRef;
```

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetServices.h`

Constants

CFNetService Registration Options

Bit flags used when registering a service.

```
enum {
    kCFNetServiceFlagNoAutoRename = 1
};
```

Constants
`kCFNetServiceFlagNoAutoRename`
Causes registrations to fail if a name conflict occurs.
Available in Mac OS X v10.4 and later.
Declared in `CFNetServices.h`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetwork/CFNetServices.h`

CFNetServiceBrowserClientCallBack Bit Flags

Bit flags providing additional information about the result returned when a client's `CFNetServiceBrowserClientCallBack` function is called.

```
enum {
kCFNetServiceFlagMoreComing = 1,
kCFNetServiceFlagIsDomain = 2,
kCFNetServiceFlagIsDefault = 4,
kCFNetServiceFlagIsRegistrationDomain = 4, /* For compatibility  */
kCFNetServiceFlagRemove = 8
};
```

Constants

`kCFNetServiceFlagMoreComing`

If set, a hint that the client's callback function will be called again soon; therefore, the client should not do anything time-consuming, such as updating the screen.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServiceFlagIsDomain`

If set, the results pertain to a search for domains. If not set, the results pertain to a search for services.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServiceFlagIsDefault`

If set, the resulting domain is the default registration or browse domain, depending on the context. For this version of the CFNetServices API, the default registration domain is the local domain. In previous versions of this API, this constant was `kCFNetServiceFlagIsRegistrationDomain`, which is retained for backward compatibility.

Available in Mac OS X v10.4 and later.

Declared in `CFNetServices.h`.

`kCFNetServiceFlagRemove`

If set, the client should remove the result item instead of adding it.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

Discussion

See `CFNetServiceBrowserClientCallBack` for additional information.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetwork/CFNetServices.h`

CFNetServiceMonitorType Constants

Record type specifier used to tell a service monitor the type of record changes to watch for.

```
enum {
kCFNetServiceMonitorTXT = 1
} typedef enum CFNetServiceMonitorType CFNetServiceMonitorType;
```

Constants

`kCFNetServiceMonitorTXT`

Watch for TXT record changes.

Available in Mac OS X v10.4 and later.

Declared in `CFNetServices.h`.

Availability

Available in Mac OS X version 10.2 and later.

Declared In

`CFNetwork/CFNetServices.h`

CFNetService Error Constants

Error codes that may be returned by CFNetServices functions or passed to CFNetServices callback functions.

```
typedef enum {
    kCFNetServicesErrorUnknown = -72000,
    kCFNetServicesErrorCollision = -72001,
    kCFNetServicesErrorNotFound = -72002,
    kCFNetServicesErrorInProgress = -72003,
    kCFNetServicesErrorBadArgument = -72004,
    kCFNetServicesErrorCancel = -72005,
    kCFNetServicesErrorInvalid = -72006,
    kCFNetServicesErrorTimeout = -72007
} CFNetServicesError;
```

Constants

`kCFNetServicesErrorUnknown`

An unknown CFNetService error occurred.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorCollision`

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

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorNotFound`

Not used.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorInProgress`

A search is already in progress.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorBadArgument`
A required argument was not provided.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorCancel`
The search or service was canceled.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorInvalid`
Invalid data was passed to a CFNetServices function.

Available in Mac OS X v10.2 and later.

Declared in `CFNetServices.h`.

`kCFNetServicesErrorTimeout`
Resolution failed because the timeout was reached.

Available in Mac OS X v10.4 and later.

Declared in `CFNetServices.h`.

Availability
Available in Mac OS X version 10.2 and later.

Declared In
`CFNetwork/CFNetServices.h`

Error Domains

Error domains.

```
extern const SInt32 kCFStreamErrorDomainMach;
extern const SInt32 kCFStreamErrorDomainNetServices;
```

Constants
`kCFStreamErrorDomainMach`
Error domain returning errors reported by Mach. For more information, see the header file `/usr/include/mach/error.h`.

Available in Mac OS X version 10.5 and later.

Declared in `CFNetServices.h`.

`kCFStreamErrorDomainNetServices`
Error domain returning errors reported by the service discovery APIs. These errors are only returned if you use the `CFNetServiceBrowser` API or any APIs introduced in Mac OS X v10.4 or later.

Available in Mac OS X version 10.5 and later.

Declared in `CFNetServices.h`.

Document Revision History

This table describes the changes to *CFNetServices Reference*.

Date	Notes
2008-07-08	Added documentation for CFNetServiceGetPortNumber.
2006-07-06	Made minor formatting changes.
2006-07-24	Updated to describe replacements for deprecated functions.
2006-02-07	New document that describes the C API for implementing Bonjour functionality in an application.

Document Revision History

Index

C

CFNetService Error Constants 41
CFNetService Registration Options 39
CFNetServiceBrowserClientCallBack Bit Flags 39
CFNetServiceBrowserClientCallBack callback 34
CFNetServiceBrowserCreate function 8
CFNetServiceBrowserGetTypeID function 9
CFNetServiceBrowserInvalidate function 9
CFNetServiceBrowserRef structure 37
CFNetServiceBrowserScheduleWithRunLoop function 10
CFNetServiceBrowserSearchForDomains function 10
CFNetServiceBrowserSearchForServices function 11
CFNetServiceBrowserStopSearch function 12
CFNetServiceBrowserUnscheduleFromRunLoop function 13
CFNetServiceCancel function 14
CFNetServiceClientCallBack callback 35
CFNetServiceClientContext structure 37
CFNetServiceCreate function 14
CFNetServiceCreateCopy function 16
CFNetServiceCreateDictionaryWithTXTData function 16
CFNetServiceCreateTXTDataWithDictionary function 17
CFNetServiceGetAddressing function 18
CFNetServiceGetDomain function 18
CFNetServiceGetName function 19
CFNetServiceGetPortNumber function (Deprecated in Mac OS X version 10.4) 19
CFNetServiceGetProtocolSpecificInformation function (Deprecated in Mac OS X version 10.4) 20
CFNetServiceGetTargetHost function 20
CFNetServiceGetTXTData function 21
CFNetServiceGetType function 21
CFNetServiceGetTypeID function 22
CFNetServiceMonitorClientCallBack callback 36
CFNetServiceMonitorCreate function 22
CFNetServiceMonitorGetTypeID function 24
CFNetServiceMonitorInvalidate function 24
CFNetServiceMonitorRef structure 38
CFNetServiceMonitorScheduleWithRunLoop function 24
CFNetServiceMonitorStart function 25
CFNetServiceMonitorStop function 26
CFNetServiceMonitorType Constants 40
CFNetServiceMonitorUnscheduleFromRunLoop function 27
CFNetServiceRef structure 39
CFNetServiceRegister function (Deprecated in Mac OS X version 10.4) 27
CFNetServiceRegisterWithOptions function 28
CFNetServiceResolve function (Deprecated in Mac OS X version 10.4) 29
CFNetServiceResolveWithTimeout function 30
CFNetServiceScheduleWithRunLoop function 31
CFNetServiceSetClient function 32
CFNetServiceSetProtocolSpecificInformation function (Deprecated in Mac OS X version 10.4) 32
CFNetServiceSetTXTData function 33
CFNetServiceUnscheduleFromRunLoop function 34

E

Error Domains 42

K

kCFNetServiceFlagIsDefault constant 40
kCFNetServiceFlagIsDomain constant 40
kCFNetServiceFlagMoreComing constant 40
kCFNetServiceFlagNoAutoRename constant 39
kCFNetServiceFlagRemove constant 40
kCFNetServiceMonitorTXT constant 41
kCFNetServicesErrorBadArgument constant 42
kCFNetServicesErrorCancel constant 42
kCFNetServicesErrorCollision constant 41

kCFNetServicesErrorInProgress constant 41
kCFNetServicesErrorInvalid constant 42
kCFNetServicesErrorNotFound constant 41
kCFNetServicesErrorTimeout constant 42
kCFNetServicesErrorUnknown constant 41
kCFStreamErrorDomainMach constant 42
kCFStreamErrorDomainNetServices constant 42