NSNetService Class Reference

Cocoa > Networking

2009-04-08

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

Contents

NSNetService Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.2 and later.
Declared in	NSNetServices.h
Companion guides	Bonjour Overview NSNetServices and CFNetServices Programming Guide
Related sample code	CocoaEcho CocoaSOAP GridCalendar PictureSharing PictureSharingBrowser

Overview

The `NSNetService` class represents a network service that your application publishes or uses as a client. This class and the `NSNetServiceBrowser` class use multicast DNS to convey information about network services to and from your application. The API of `NSNetService` provides a convenient way to publish the services offered by your application and to resolve the socket address for a service.

The types of services you access using `NSNetService` are the same types that you access directly using BSD sockets. HTTP and FTP are two services commonly provided by systems. (For a list of common services and the ports used by those services, see the file `/etc/services`.) Applications can also define their own custom services to provide specific data to clients.

You can use the `NSNetService` class as either a publisher of a service or as a client of a service. If your application publishes a service, your code must acquire a port and prepare a socket to communicate with clients. Once your socket is ready, you use the `NSNetService` class to notify clients that your service is ready. If your application is the client of a network service, you can either create an `NSNetService` object directly (if you know the exact host and port information) or you can use an `NSNetServiceBrowser` object to browse for services.

To publish a service, you must initialize your `NSNetService` object with the service name, domain, type, and port information. All of this information must be valid for the socket created by your application. Once initialized, you call the `publish` (page 14) method to broadcast your service information out to the network.

When connecting to a service, you would normally use the `NSNetServiceBrowser` class to locate the service on the network and obtain the corresponding `NSNetService` object. Once you have the object, you proceed to call the `resolveWithTimeout:` (page 15) method to verify that the service is available and ready for your application. If it is, the `addresses` (page 9) method returns the socket information you can use to connect to the service.

The methods of `NSNetService` operate asynchronously so that your application is not impacted by the speed of the network. All information about a service is returned to your application through the `NSNetService` object's delegate. You must provide a delegate object to respond to messages and to handle errors appropriately.

Tasks

Creating Network Services

– `initWithDomain:type:name:` (page 11)
: Returns the receiver, initialized as a network service of a given type and sets the initial host information.

– `initWithDomain:type:name:port:` (page 12)
: Initializes the receiver as a network service of type *`type`* at the socket location specified by *`domain`*, *`name`*, and *`port`*.

Configuring Network Services

\+ `dataFromTXTRecordDictionary:` (page 8)
: Returns an `NSData` object representing a TXT record formed from a given dictionary.

\+ `dictionaryFromTXTRecordData:` (page 8)
: Returns a dictionary representing a TXT record given as an `NSData` object.

– `addresses` (page 9)
: Returns an array containing `NSData` objects, each of which contains a socket address for the service.

– `domain` (page 10)
: Returns the domain name of the service.

– `getInputStream:outputStream:` (page 10)
: Retrieves by reference the input and output streams for the receiver and returns a Boolean value that indicates whether they were retrieved successfully.

– `hostName` (page 11)
: Returns the host name of the computer providing the service.

– `name` (page 13)
: Returns the name of the service.

– `type` (page 19)
: Returns the type of the service.

– `TXTRecordData` (page 18)
: Returns the TXT record for the receiver.

- `setTXTRecordData:` (page 17)
  Sets the TXT record for the receiver, and returns a Boolean value that indicates whether the operation was successful.
- `delegate` (page 10)
  Returns the delegate for the receiver.
- `setDelegate:` (page 16)
  Sets the delegate for the receiver.
- `protocolSpecificInformation` (page 25) Deprecated in Mac OS X v10.4
  Returns protocol specific information for legacy ZeroConf-style clients. (Deprecated. Use `TXTRecordData` (page 18) instead.)
- `setProtocolSpecificInformation:` (page 25) Deprecated in Mac OS X v10.4
  Sets protocol specific information for legacy ZeroConf-style clients. (Deprecated. Use `setTXTRecordData:` (page 17) instead.)

Managing Run Loops

- `scheduleInRunLoop:forMode:` (page 16)
  Adds the service to the specified run loop.
- `removeFromRunLoop:forMode:` (page 14)
  Removes the service from the given run loop for a given mode.

Using Network Services

- `publish` (page 14)
  Attempts to advertise the receiver's on the network.
- `publishWithOptions:` (page 14)
  Attempts to advertise the receiver on the network, with the given options.
- `netServiceWillPublish:` (page 21) *delegate method*
  Notifies the delegate that the network is ready to publish the service.
- `netService:didNotPublish:` (page 19) *delegate method*
  Notifies the delegate that a service could not be published.
- `netServiceDidPublish:` (page 20) *delegate method*
  Notifies the delegate that a service was successfully published.
- `resolve` (page 15)
  Starts a resolve process for the receiver. (Deprecated. Use `resolveWithTimeout:` (page 15) instead.)
- `resolveWithTimeout:` (page 15)
  Starts a resolve process of a finite duration for the receiver.
- `netServiceWillResolve:` (page 22) *delegate method*
  Notifies the delegate that the network is ready to resolve the service.
- `netService:didNotResolve:` (page 19) *delegate method*
  Informs the delegate that an error occurred during resolution of a given service.
- `netServiceDidResolveAddress:` (page 21) *delegate method*
  Informs the delegate that the address for a given service was resolved.

Class Methods

dataFromTXTRecordDictionary:

Returns an `NSData` object representing a TXT record formed from a given dictionary.

```
+ (NSData *)dataFromTXTRecordDictionary:(NSDictionary *)txtDictionary
```

Parameters

txtDictionary
  A dictionary containing a TXT record.

Return Value
An `NSData` object representing TXT data formed from *txtDictionary*. Fails an assertion if *txtDictionary* cannot be represented as an `NSData` object.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `TXTRecordData` (page 18)
+ `dictionaryFromTXTRecordData:` (page 8)

Declared In
`NSNetServices.h`

dictionaryFromTXTRecordData:

Returns a dictionary representing a TXT record given as an `NSData` object.

```
+ (NSDictionary *)dictionaryFromTXTRecordData:(NSData *)txtData
```

Parameters

txtData

A data object encoding a TXT record.

Return Value

A dictionary representing *txtData*. The dictionary's keys are `NSString` objects using UTF8 encoding. The values associated with all the dictionary's keys are `NSData` objects that encapsulate strings or data.

Fails an assertion if *txtData* cannot be represented as an `NSDictionary` object.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `TXTRecordData` (page 18)

+ `dataFromTXTRecordDictionary:` (page 8)

Declared In

`NSNetServices.h`

Instance Methods

addresses

Returns an array containing `NSData` objects, each of which contains a socket address for the service.

```
- (NSArray *)addresses
```

Return Value

An array containing `NSData` objects, each of which contains a socket address for the service. Each `NSData` object in the returned array contains an appropriate `sockaddr` structure that you can use to connect to the socket. The exact type of this structure depends on the service to which you are connecting. If no addresses were resolved for the service, the returned array contains zero elements.

Discussion

It is possible for a single service to resolve to more than one address or not resolve to any addresses. A service might resolve to multiple addresses if the computer publishing the service is currently multihoming.

Availability

Available in Mac OS X v10.2 and later.

See Also

- `resolve` (page ?)

Related Sample Code

CocoaSOAP

PictureSharingBrowser

Declared In

`NSNetServices.h`

delegate

Returns the delegate for the receiver.

```
- (id)delegate
```

Return Value
The delegate for the receiver.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- setDelegate:` (page 16)

Declared In
`NSNetServices.h`

domain

Returns the domain name of the service.

```
- (NSString *)domain
```

Return Value
The domain name of the service.

This can be an explicit domain name or it can contain the generic local domain name, `@"local."` (note the trailing period, which indicates an absolute name).

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
GridCalendar

Declared In
`NSNetServices.h`

getInputStream:outputStream:

Retrieves by reference the input and output streams for the receiver and returns a Boolean value that indicates whether they were retrieved successfully.

```
- (BOOL)getInputStream:(NSInputStream **)inputStream outputStream:(NSOutputStream
    **)outputStream
```

Parameters

inputStream
: Upon return, the input stream for the receiver.

outputStream
: Upon return, the output stream for the receiver.

Return Value
`YES` if the streams are created successfully, otherwise `NO`.

Discussion
After this method is called, no delegate callbacks are called by the receiver.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSNetServices.h`

hostName

Returns the host name of the computer providing the service.

```
- (NSString *)hostName
```

Return Value
The host name of the computer providing the service. Returns `nil` if a successful resolve has not occurred.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSNetServices.h`

initWithDomain:type:name:

Returns the receiver, initialized as a network service of a given type and sets the initial host information.

```
- (id)initWithDomain:(NSString *)domain type:(NSString *)type name:(NSString *)name
```

Parameters

domain
: The domain for the service. For the local domain, use `@"local."` not `@""`.

type
: The network service type.

  type must contain both the service type and transport layer information. To ensure that the mDNS responder searches for services, as opposed to hosts, prefix both the service name and transport layer name with an underscore character ("_"). For example, to search for an HTTP service on TCP, you would use the type string "`_http._tcp.`". Note that the period character at the end of the string, which indicates that the domain name is an absolute name, is required.

name
: The name of the service to resolve.

Return Value
The receiver, initialized as a network service named *name* of type *type* in the domain *domain*.

Discussion
This method is the appropriate initializer to use to resolve a service—to publish a service, use initWithDomain:type:name:port: (page 12).

If you know the values for *domain*, *type*, and *name* of the service you wish to connect to, you can create an `NSNetService` object using this initializer and call `resolveWithTimeout:` (page 15) on the result.

You cannot use this initializer to publish a service. This initializer passes an invalid port number to the designated initializer, which prevents the service from being registered. Calling `publish` (page 14) on an `NSNetService` object initialized with this method generates a call to your delegate's `netService:didNotPublish:` (page 19) method with an `NSNetServicesBadArgumentError` error.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- initWithDomain:type:name:port:` (page 12)

Related Sample Code
CocoaSOAP
GridCalendar

Declared In
`NSNetServices.h`

initWithDomain:type:name:port:

Initializes the receiver as a network service of type *type* at the socket location specified by *domain*, *name*, and *port*.

```
- (id)initWithDomain:(NSString *)domain type:(NSString *)type name:(NSString *)name
    port:(int)port
```

Parameters

domain
: The domain for the service. For the local domain, use `@"local."` not `@""`.

  It is generally preferred to use a `NSNetServiceBrowser` object to obtain the local registration domain in which to publish your service. To use this default domain, simply pass in an empty string (`@""`).

type
: The network service type.

  type must contain both the service type and transport layer information. To ensure that the mDNS responder searches for services, as opposed to hosts, prefix both the service name and transport layer name with an underscore character ("_"). For example, to search for an HTTP service on TCP, you would use the type string `"_http._tcp."`. Note that the period character at the end of the string, which indicates that the domain name is an absolute name, is required.

name
: The name by which the service is identified to the network. The name must be unique.

port
: The port on which the service is published.

  port must be a port number acquired by your application for the service.

Discussion
You use this method to create a service that you wish to publish on the network. Although you can also use this method to create a service you wish to resolve on the network, it is generally more appropriate to use the `initWithDomain:type:name:` (page 11) method instead.

When publishing a service, you must provide valid arguments in order to advertise your service correctly. If the host computer has access to multiple registration domains, you must create separate `NSNetService` objects for each domain. If you attempt to publish in a domain for which you do not have registration authority, your request may be denied.

It is acceptable to use an empty string for the *domain* argument when publishing or browsing a service, but do not rely on this for resolution.

This method is the designated initializer.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- initWithDomain:type:name:` (page 11)

Related Sample Code
CocoaEcho

CocoaHTTPServer

CocoaSOAP

PictureSharing

Declared In
`NSNetServices.h`

name

Returns the name of the service.

```
- (NSString *)name
```

Return Value
The name of the service.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
GridCalendar

Declared In
`NSNetServices.h`

port

Provides the port of the receiver.

```
- (NSInteger)port
```

Return Value
The receiver's port. `-1` when it has not been resolved.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSNetServices.h`

publish

Attempts to advertise the receiver's on the network.

```
- (void)publish
```

Discussion
This method returns immediately, with success or failure indicated by the callbacks to the delegate.

Availability
Available in Mac OS X v10.2 and later.

See Also
`- stop` (page 18)

Declared In
`NSNetServices.h`

publishWithOptions:

Attempts to advertise the receiver on the network, with the given options.

```
- (void)publishWithOptions:(NSNetServiceOptions)serviceOptions
```

Parameters

serviceOptions
Options for the receiver.

Discussion
This method returns immediately, with success or failure indicated by the callbacks to the delegate.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSNetServices.h`

removeFromRunLoop:forMode:

Removes the service from the given run loop for a given mode.

```
- (void)removeFromRunLoop:(NSRunLoop *)aRunLoop forMode:(NSString *)mode
```

Parameters

aRunLoop
The run loop from which to remove the receiver.

mode
: The run loop mode from which to remove the receiver. Possible values for *mode* are discussed in the "Constants" section of `NSRunLoop`.

Discussion

You can use this method in conjunction with `scheduleInRunLoop:forMode:` (page 16) to transfer the service to a different run loop. Although it is possible to remove an `NSNetService` object completely from any run loop and then attempt actions on it, it is an error to do so.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- scheduleInRunLoop:forMode:` (page 16)

Declared In

`NSNetServices.h`

resolve

Starts a resolve process for the receiver. (**Deprecated.** Use `resolveWithTimeout:` (page 15) instead.)

```
- (void)resolve
```

Discussion

Attempts to determine at least one address for the receiver. This method returns immediately, with success or failure indicated by the callbacks to the delegate.

In Mac OS X v10.4, this method calls `resolveWithTimeout:` (page 15) with a timeout value of `5`.

Availability

Available in Mac OS X v10.2 and later.

See Also

`- addresses` (page 9)
`- stop` (page 18)
`- resolveWithTimeout:` (page 15)

Declared In

`NSNetServices.h`

resolveWithTimeout:

Starts a resolve process of a finite duration for the receiver.

```
- (void)resolveWithTimeout:(NSTimeInterval)timeout
```

Parameters

timeout
: The maximum number of seconds to attempt a resolve.

Discussion
If the resolve succeeds before the *timeout* period lapses, the receiver sends `netServiceDidResolveAddress:` (page 21) to the delegate. Otherwise, the receiver sends `netService:didNotResolve:` (page 19) to the delegate.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addresses` (page 9)
- `stop` (page 18)

Related Sample Code
CocoaSOAP

Declared In
`NSNetServices.h`

scheduleInRunLoop:forMode:

Adds the service to the specified run loop.

```
- (void)scheduleInRunLoop:(NSRunLoop *)aRunLoop forMode:(NSString *)mode
```

Parameters

aRunLoop
The run loop to which to add the receiver.

mode
The run loop mode to which to add the receiver. Possible values for *mode* are discussed in the "Constants" section of `NSRunLoop`.

Discussion
You can use this method in conjunction with `removeFromRunLoop:forMode:` (page 14) to transfer a service to a different run loop. You should not attempt to run a service on multiple run loops.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `removeFromRunLoop:forMode:` (page 14)

Related Sample Code
CocoaSOAP

Declared In
`NSNetServices.h`

setDelegate:

Sets the delegate for the receiver.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate
The delegate for the receiver.

Discussion
The delegate is not retained.

Availability
Available in Mac OS X v10.2 and later.

See Also
– delegate (page 10)

Declared In
`NSNetServices.h`

setTXTRecordData:

Sets the TXT record for the receiver, and returns a Boolean value that indicates whether the operation was successful.

```
- (BOOL)setTXTRecordData:(NSData *)recordData
```

Parameters

recordData
The TXT record for the receiver.

Return Value
`YES` if *recordData* is successfully set as the TXT record, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
– TXTRecordData (page 18)

Declared In
`NSNetServices.h`

startMonitoring

Starts the monitoring of TXT-record updates for the receiver.

```
- (void)startMonitoring
```

Discussion
The delegate must implement `netService:didUpdateTXTRecordData:` (page 20), which is called when the TXT record for the receiver is updated.

Availability
Available in Mac OS X v10.4 and later.

See Also
– stopMonitoring (page 18)

Declared In
`NSNetServices.h`

stop

Halts a currently running attempt to publish or resolve a service.

```
- (void)stop
```

Discussion
This method results in the sending of a `netServiceDidStop:` (page 21) message to the delegate.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
CocoaSOAP

PictureSharingBrowser

Declared In
`NSNetServices.h`

stopMonitoring

Stops the monitoring of TXT-record updates for the receiver.

```
- (void)stopMonitoring
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- startMonitoring` (page 17)

Declared In
`NSNetServices.h`

TXTRecordData

Returns the TXT record for the receiver.

```
- (NSData *)TXTRecordData
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setTXTRecordData:` (page 17)

`+ dictionaryFromTXTRecordData:` (page 8)

`+ dataFromTXTRecordDictionary:` (page 8)

Declared In
`NSNetServices.h`

type

Returns the type of the service.

```
- (NSString *)type
```

Return Value
The type of the service.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
GridCalendar

Declared In
`NSNetServices.h`

Delegate Methods

netService:didNotPublish:

Notifies the delegate that a service could not be published.

```
- (void)netService:(NSNetService *)sender didNotPublish:(NSDictionary *)errorDict
```

Parameters

sender
: The service that could not be published.

errorDict
: A dictionary containing information about the problem. The dictionary contains the keys `NSNetServicesErrorCode` and `NSNetServicesErrorDomain`.

Discussion
This method may be called long after a `netServiceWillPublish:` (page 21) message has been delivered to the delegate.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`NSNetServices.h`

netService:didNotResolve:

Informs the delegate that an error occurred during resolution of a given service.

```
- (void)netService:(NSNetService *)sender didNotResolve:(NSDictionary *)errorDict
```

Parameters

sender

The service that did not resolve.

errorDict

A dictionary containing information about the problem. The dictionary contains the keys `NSNetServicesErrorCode` and `NSNetServicesErrorDomain`.

Discussion

Clients may try to resolve again upon receiving this error. For example, a DNS rotary may yield different IP addresses on different resolution requests.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSNetServices.h`

netService:didUpdateTXTRecordData:

Notifies the delegate that the TXT record for a given service has been updated.

```
- (void)netService:(NSNetService *)sender didUpdateTXTRecordData:(NSData *)data
```

Parameters

sender

The service whose TXT record was updated.

data

The new TXT record.

Availability

Available in Mac OS X v10.4 and later.

See Also

`– startMonitoring` (page 17)

Declared In

`NSNetServices.h`

netServiceDidPublish:

Notifies the delegate that a service was successfully published.

```
- (void)netServiceDidPublish:(NSNetService *)sender
```

Parameters

sender

The service that was published.

Availability

Available in Mac OS X v10.4 and later.

Declared In
`NSNetServices.h`

netServiceDidResolveAddress:

Informs the delegate that the address for a given service was resolved.

```
- (void)netServiceDidResolveAddress:(NSNetService *)sender
```

Parameters

sender
The service that was resolved.

Discussion
The delegate can use the `addresses` (page 9) method to retrieve the service's address.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `addresses` (page 9)

Declared In
`NSNetServices.h`

netServiceDidStop:

Informs the delegate that a `publish` (page 14) or `resolveWithTimeout:` (page 15) request was stopped.

```
- (void)netServiceDidStop:(NSNetService *)sender
```

Parameters

sender
The service that stopped.

Availability
Available in Mac OS X v10.2 and later.

See Also
- `stop` (page 18)

Declared In
`NSNetServices.h`

netServiceWillPublish:

Notifies the delegate that the network is ready to publish the service.

```
- (void)netServiceWillPublish:(NSNetService *)sender
```

Parameters

sender

The service that is ready to publish.

Discussion

Publication of the service proceeds asynchronously and may still generate a call to the delegate's `netService:didNotPublish:` (page 19) method if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSNetServices.h`

netServiceWillResolve:

Notifies the delegate that the network is ready to resolve the service.

```
- (void)netServiceWillResolve:(NSNetService *)sender
```

Parameters

sender

The service that the network is ready to resolve.

Discussion

Resolution of the service proceeds asynchronously and may still generate a call to the delegate's `netService:didNotResolve:` (page 19) method if an error occurs.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSNetServices.h`

Constants

NSNetServices Errors

If an error occurs, the delegate error-handling methods return a dictionary with the following keys.

```
extern NSString *NSNetServicesErrorCode;
extern NSString *NSNetServicesErrorDomain;
```

Constants

`NSNetServicesErrorCode`

This key identifies the error that occurred during the most recent operation.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

`NSNetServicesErrorDomain`
This key identifies the originator of the error, which is either the `NSNetService` object or the mach network layer. For most errors, you should not need the value provided by this key.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

Declared In
`NSNetServices.h`

NSNetServicesError

These constants identify errors that can occur when accessing net services.

```
typedef enum {
    NSNetServicesUnknownError = -72000,
    NSNetServicesCollisionError = -72001,
    NSNetServicesNotFoundError    = -72002,
    NSNetServicesActivityInProgress = -72003,
    NSNetServicesBadArgumentError = -72004,
    NSNetServicesCancelledError = -72005,
    NSNetServicesInvalidError = -72006,
    NSNetServicesTimeoutError = -72007,
} NSNetServicesError;
```

Constants

`NSNetServicesUnknownError`
An unknown error occurred.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

`NSNetServicesCollisionError`
The service could not be published because the name is already in use. The name could be in use locally or on another system.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

`NSNetServicesNotFoundError`
The service could not be found on the network.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

`NSNetServicesActivityInProgress`
The net service cannot process the request at this time. No additional information about the network state is known.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

`NSNetServicesBadArgumentError`
An invalid argument was used when creating the `NSNetService` object.

Available in Mac OS X v10.2 and later.

Declared in `NSNetServices.h`.

`NSNetServicesCancelledError`
: The client canceled the action.

  Available in Mac OS X v10.2 and later.

  Declared in `NSNetServices.h`.

`NSNetServicesInvalidError`
: The net service was improperly configured.

  Available in Mac OS X v10.2 and later.

  Declared in `NSNetServices.h`.

`NSNetServicesTimeoutError`
: The net service has timed out.

  Available in Mac OS X v10.4 and later.

  Declared in `NSNetServices.h`.

Declared In
`NSNetServices.h`

NSNetServiceOptions

These constants specify options for a network service.

```
enum {
   NSNetServiceNoAutoRename = 1 << 0
};
typedef NSUInteger NSNetServiceOptions;
```

Constants

`NSNetServiceNoAutoRename`
: Specifies that the network service not rename itself in the event of a name collision.

  Available in Mac OS X v10.5 and later.

  Declared in `NSNetServices.h`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSNetServices.h`

Deprecated NSNetService Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

protocolSpecificInformation

Returns protocol specific information for legacy ZeroConf-style clients. (Deprecated in Mac OS X v10.4. Use `TXTRecordData` (page 18) instead.)

```
- (NSString *)protocolSpecificInformation
```

Return Value
Any protocol-specific data associated with the service.

Discussion
This method is provided for legacy support of older ZeroConf-style clients and its use is discouraged.

Availability
Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.4.

See Also
`- setProtocolSpecificInformation:` (page 25)

Declared In
`NSNetServices.h`

setProtocolSpecificInformation:

Sets protocol specific information for legacy ZeroConf-style clients. (Deprecated in Mac OS X v10.4. Use `setTXTRecordData:` (page 17) instead.)

```
- (void)setProtocolSpecificInformation:(NSString *)specificInformation
```

Parameters

`specificInformation`
Information for the protocol.

Discussion
Attaches protocol-specific data to the service.

This method is provided for legacy support of older ZeroConf-style clients and its use is discouraged.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.4.

See Also
`- protocolSpecificInformation` (page 25)

Declared In
`NSNetServices.h`

Document Revision History

This table describes the changes to *NSNetService Class Reference*.

Date	Notes
2009-04-08	Miscellaneous edits.
2008-02-08	Corrected typographical errors.
2007-04-24	Updated for Mac OS X v10.5.
2006-05-23	Added "NSNetServices and CFNetServices Programming Guide" as a companion document.
	First publication of this content as a separate document.

Document Revision History

Index

S

T