

---

# System Configuration Framework Reference

Networking, Internet, & Web



2009-07-30



Apple Inc.  
© 2009 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, AirPort, AppleTalk, FireWire, Keychain, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY**

**DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

**Introduction**      **Introduction** 7

---

**Part I**              **Other References** 9

---

**Chapter 1**        **SCDynamicStore Reference** 11

---

Overview 11  
Functions by Task 11  
Functions 13  
Data Types 22  
Constants 23

**Chapter 2**        **SCDynamicStoreCopyDHCPInfo Reference** 25

---

Overview 25  
Functions 25

**Chapter 3**        **SCDynamicStoreCopySpecific Reference** 27

---

Overview 27  
Functions 27

**Chapter 4**        **SCDynamicStoreKey Reference** 31

---

Overview 31  
Functions 31

**Chapter 5**        **SCNetwork Reference** 37

---

Overview 37  
Functions 37  
Constants 39

**Chapter 6**        **SCNetworkConfiguration Reference** 41

---

Overview 41  
Functions by Task 41  
Functions 46  
Data Types 82  
Constants 83

**Chapter 7**      **SCNetworkConnection Reference 89**

---

Overview 89  
Functions by Task 89  
Functions 90  
Data Types 98  
Constants 100

**Chapter 8**      **SCNetworkReachability Reference 105**

---

Overview 105  
Functions by Task 105  
Functions 106  
Data Types 110  
Constants 112

**Chapter 9**      **SCPreferences Reference 115**

---

Overview 115  
Functions by Task 115  
Functions 117  
Data Types 126  
Constants 128

**Chapter 10**      **SCPreferencesPath Reference 129**

---

Overview 129  
Functions by Task 129  
Functions 130

**Chapter 11**      **SCPreferencesSetSpecific Reference 135**

---

Overview 135  
Functions 135

**Chapter 12**      **SCSchemaDefinitions Reference 137**

---

Overview 137  
Constants 137

**Chapter 13**      **SystemConfiguration Reference 169**

---

Overview 169  
Functions 169  
Constants 170

**Document Revision History 175**

---

**Index 177**

---



# Introduction

---

**Companion guide**

System Configuration Programming Guidelines

**Declared in**

SCDynamicStore.h  
SCDynamicStoreCopyDHCPInfo.h  
SCDynamicStoreCopySpecific.h  
SCDynamicStoreKey.h  
SCNetwork.h  
SCNetworkConfiguration.h  
SCNetworkConnection.h  
SCNetworkReachability.h  
SCPreferences.h  
SCPreferencesPath.h  
SCPreferencesSetSpecific.h  
SCSchemaDefinitions.h  
SystemConfiguration.h

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



# Other References

---



# SCDynamicStore Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCDynamicStore.h

## Overview

The `SCDynamicStore` programming interface provides access to the key-value pairs in the dynamic store of a running system. The dynamic store contains, among other items, a copy of the configuration settings for the currently active set (which is sometimes referred to as the location) and information about the current network state.

The functions in the `SCDynamicStore` programming interface allow you to find key-value pairs, add or remove key-value pairs, add or change values, and request notifications. Note that these functions follow Core Foundation function-name conventions. A function that has "Create" or "Copy" in its name returns a reference you must release with the `CFRelease` function.

To use these functions, you must first establish a dynamic store session using the [SCDynamicStoreCreate](#) (page 16) function. When you are finished with the session, use `CFRelease` to close it.

## Functions by Task

### Creating a Dynamic Store Session

[SCDynamicStoreCreateWithOptions](#) (page 17)

Creates a new session used to interact with the dynamic store maintained by the System Configuration server.

[SCDynamicStoreCreate](#) (page 16)

Creates a new session used to interact with the dynamic store maintained by the System Configuration server.

### Adding or Updating Keys and Values

[SCDynamicStoreAddTemporaryValue](#) (page 13)

Temporarily adds the specified key-value pair to the dynamic store, if no such key already exists.

[SCDynamicStoreAddValue](#) (page 13)

Adds the specified key-value pair to the dynamic store, if no such key already exists.

[SCDynamicStoreSetMultiple](#) (page 20)

Updates multiple values in the dynamic store.

[SCDynamicStoreSetValue](#) (page 21)

Adds or replaces a value in the dynamic store for the specified key.

## Getting Keys and Values

[SCDynamicStoreCopyKeyList](#) (page 14)

Returns the keys that represent the current dynamic store entries that match the specified pattern.

[SCDynamicStoreCopyMultiple](#) (page 14)

Returns the key-value pairs that match the specified keys and key patterns.

[SCDynamicStoreCopyNotifiedKeys](#) (page 15)

Returns the keys that have changed since the last call to this function.

[SCDynamicStoreCopyValue](#) (page 15)

Returns the value associated with the specified key.

## Monitoring Keys and Values

[SCDynamicStoreNotifyValue](#) (page 19)

Causes a notification to be delivered for the specified key in the dynamic store.

[SCDynamicStoreSetNotificationKeys](#) (page 21)

Specifies a set of keys and key patterns that should be monitored for changes.

[SCDynamicStoreSetDispatchQueue](#) (page 20)

Initiates notifications for the notification keys, using the specified dispatch queue for the callback.

## Removing Keys and Values

[SCDynamicStoreRemoveValue](#) (page 19)

Removes the value of the specified key from the dynamic store.

## Creating a Run Loop Source

[SCDynamicStoreCreateRunLoopSource](#) (page 17)

Creates a run loop source object that can be added to the application's run loop.

## Getting Information About the Dynamic Store

[SCDynamicStoreGetTypeID](#) (page 18)

Returns the type identifier of all `SCDynamicStore` instances.

## Functions

### SCDynamicStoreAddTemporaryValue

Temporarily adds the specified key-value pair to the dynamic store, if no such key already exists.

```
Boolean SCDynamicStoreAddTemporaryValue (
    SCDynamicStoreRef store,
    CFStringRef key,
    CFPropertyListRef value
);
```

#### Parameters

*store*

The dynamic store session.

*key*

The key of the value to add to the dynamic store.

*value*

The value to add to the dynamic store.

#### Return Value

TRUE if the key was added; FALSE if the key was already present in the dynamic store or if an error occurred.

#### Discussion

Unless the key is updated by another session, the key-value pair added by this function is removed automatically when the session is closed.

#### Availability

Available in Mac OS X v10.1 and later.

#### Declared In

SCDynamicStore.h

### SCDynamicStoreAddValue

Adds the specified key-value pair to the dynamic store, if no such key already exists.

```
Boolean SCDynamicStoreAddValue (
    SCDynamicStoreRef store,
    CFStringRef key,
    CFPropertyListRef value
);
```

#### Parameters

*store*

The dynamic store session.

*key*

The key of the value to add to the dynamic store.

*value*

The value to add to the dynamic store.

**Return Value**

TRUE if the key was added; FALSE if the key was already present in the dynamic store or if an error occurred.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCDynamicStore.h

**SCDynamicStoreCopyKeyList**

Returns the keys that represent the current dynamic store entries that match the specified pattern.

```
CFArrayRef SCDynamicStoreCopyKeyList (
    SCDynamicStoreRef store,
    CFStringRef pattern
);
```

**Parameters**

*store*

The dynamic store session.

*pattern*

A `regex(3)` regular expression pattern used to match the dynamic store keys.

**Return Value**

An array of matching keys, or NULL if an error occurred. You must release the returned value.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCDynamicStore.h

**SCDynamicStoreCopyMultiple**

Returns the key-value pairs that match the specified keys and key patterns.

```
CFDictionaryRef SCDynamicStoreCopyMultiple (
    SCDynamicStoreRef store,
    CFArrayRef keys,
    CFArrayRef patterns
);
```

**Parameters**

*store*

The dynamic store session.

*keys*

The keys associated with the desired values or NULL if no specific keys are requested.

*patterns*

An array of `regex(3)` pattern strings used to match the keys, or `NULL` if no key patterns are requested.

**Return Value**

A dictionary of key-value pairs that match the specified keys and key patterns, or `NULL` if an error occurred. You must release the returned value.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStore.h`

## SCDynamicStoreCopyNotifiedKeys

Returns the keys that have changed since the last call to this function.

```
CFArrayRef SCDynamicStoreCopyNotifiedKeys (
    SCDynamicStoreRef store
);
```

**Parameters**

*store*

The dynamic store session.

**Return Value**

The keys that have changed, or `NULL` if an error occurred. You must release the returned value.

**Discussion**

If possible, your application should use the notification functions instead of polling for the list of changed keys returned by this function.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStore.h`

## SCDynamicStoreCopyValue

Returns the value associated with the specified key.

```
CFPropertyListRef SCDynamicStoreCopyValue (
    SCDynamicStoreRef store,
    CFStringRef key
);
```

**Parameters**

*store*

The dynamic store session.

*key*

The key associated with the desired value.

**Return Value**

The value associated with the specified key, or NULL if no value was located or if an error occurred. You must release the returned value.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCDynamicStore.h

**SCDynamicStoreCreate**

Creates a new session used to interact with the dynamic store maintained by the System Configuration server.

```
SCDynamicStoreRef SCDynamicStoreCreate (
    CFAllocatorRef allocator,
    CFStringRef name,
    SCDynamicStoreCallback callout,
    SCDynamicStoreContext *context
);
```

**Parameters**

*allocator*

The allocator that should be used to allocate memory for the local dynamic store object. This parameter may be NULL in which case the current default allocator is used. If this value is not a valid CFAllocatorRef, the behavior is undefined.

*name*

The name of the calling process or plug-in of the caller.

*callout*

The function to be called when a watched value in the dynamic store is changed. Pass NULL if no callouts are desired.

*context*

The context associated with the callout. See [SCDynamicStoreContext](#) (page 22) for more information about this value.

**Return Value**

A reference to the new dynamic store session. You must release the returned value.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

ImageClient

MoreSCF

**Declared In**

SCDynamicStore.h

## SCDynamicStoreCreateRunLoopSource

Creates a run loop source object that can be added to the application's run loop.

```
CFRunLoopSourceRef SCDynamicStoreCreateRunLoopSource (  
    CFAllocatorRef allocator,  
    SCDynamicStoreRef store,  
    CFIndex order  
);
```

### Parameters

*allocator*

The allocator that should be used to allocate memory for the run loop source. This parameter may be NULL in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

*store*

The dynamic store session.

*order*

The order in which the sources that are ready to be processed are handled, on platforms that support it and for source versions that support it. A source with a lower order number is processed before a source with a higher order number. It is inadvisable to depend on the order number for any architectural or design aspect of code. In the absence of any reason to do otherwise, pass 0 for this parameter.

### Return Value

The new run loop source object. You must release the returned value.

### Discussion

Note that all dynamic store notifications are delivered using the run loop source this function returns.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

ImageClient

### Declared In

SCDynamicStore.h

## SCDynamicStoreCreateWithOptions

Creates a new session used to interact with the dynamic store maintained by the System Configuration server.

```

SCDynamicStoreRef SCDynamicStoreCreateWithOptions (
    CFAllocatorRef allocator,
    CFStringRef name,
    CFDictionaryRef storeOptions,
    SCDynamicStoreCallback callout,
    SCDynamicStoreContext *context
);

```

**Parameters***allocator*

The allocator that should be used to allocate memory for the local dynamic store object. This parameter may be NULL in which case the current default allocator is used. If this value is not a valid CFAllocatorRef, the behavior is undefined.

*name*

The name of the calling process or plug-in of the caller.

*storeOptions*

A dictionary of options for the dynamic store session (such as whether all keys added or set into the dynamic store should be per-session keys). Pass NULL if no options are desired.

Currently, the available options are:

Key	Value
<a href="#">kSCDynamicStoreUseSessionKeys</a> (page 23)	CFBooleanRef

*callout*

The function to be called when a watched value in the dynamic store is changed. Pass NULL if no callouts are desired.

*context*

The context associated with the callout. See [SCDynamicStoreContext](#) (page 22) for more information about this value.

**Return Value**

A reference to the new dynamic store session. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCDynamicStore.h

**SCDynamicStoreGetTypeID**

Returns the type identifier of all SCDynamicStore instances.

```

CTypeID SCDynamicStoreGetTypeID (
    void
);

```

**Return Value**

The type identifier of all SCDynamicStore instances.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCDynamicStore.h

**SCDynamicStoreNotifyValue**

Causes a notification to be delivered for the specified key in the dynamic store.

```
Boolean SCDynamicStoreNotifyValue (  
    SCDynamicStoreRef store,  
    CFStringRef key  
);
```

**Parameters***store*

The dynamic store session.

*key*

The key that should be flagged as changed. All dynamic store sessions that are monitoring this key will receive a notification. Note that the key's value is not updated.

**Return Value**

TRUE if the notification was processed; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCDynamicStore.h

**SCDynamicStoreRemoveValue**

Removes the value of the specified key from the dynamic store.

```
Boolean SCDynamicStoreRemoveValue (  
    SCDynamicStoreRef store,  
    CFStringRef key  
);
```

**Parameters***store*

The dynamic store session.

*key*

The key of the value to remove.

**Return Value**

TRUE if the key was removed; FALSE if no value was located or an error occurred.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCDynamicStore.h

## SCDynamicStoreSetDispatchQueue

Initiates notifications for the notification keys, using the specified dispatch queue for the callback.

```

Boolean SCDynamicStoreSetDispatchQueue (
    SCDynamicStoreRef store,
    dispatch_queue_t queue
);

```

### Parameters

*store*

The dynamic store session.

*queue*

The dispatch queue on which to run the callback function. Pass `NULL` to disable notifications and release the queue.

### Return Value

TRUE if notifications were successfully initiated; otherwise, FALSE.

### Availability

Available in Mac OS X v10.6 and later.

### Declared In

SCDynamicStore.h

## SCDynamicStoreSetMultiple

Updates multiple values in the dynamic store.

```

Boolean SCDynamicStoreSetMultiple (
    SCDynamicStoreRef store,
    CFDictionaryRef keysToSet,
    CFArrayRef keysToRemove,
    CFArrayRef keysToNotify
);

```

### Parameters

*store*

The dynamic store session.

*keysToSet*

A dictionary of key-value pairs to add to the dynamic store.

*keysToRemove*

An array of keys to remove from the dynamic store.

*keysToNotify*

An array of keys to flag as changed (without changing their values).

### Return Value

TRUE if the dynamic store updates were successful; otherwise, FALSE.

### Availability

Available in Mac OS X v10.1 and later.

### Declared In

SCDynamicStore.h

## SCDynamicStoreSetNotificationKeys

Specifies a set of keys and key patterns that should be monitored for changes.

```
Boolean SCDynamicStoreSetNotificationKeys (
    SCDynamicStoreRef store,
    CFArrayRef keys,
    CFArrayRef patterns
);
```

### Parameters

*store*

The dynamic store session being watched.

*keys*

An array of keys to be monitored or NULL if no specific keys are to be monitored.

*patterns*

An array of `regex(3)` pattern strings used to match keys to be monitored or NULL if no key patterns are to be monitored.

### Return Value

TRUE if the set of notification keys and patterns was successfully updated; otherwise, FALSE.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

ImageClient

### Declared In

SCDynamicStore.h

## SCDynamicStoreSetValue

Adds or replaces a value in the dynamic store for the specified key.

```
Boolean SCDynamicStoreSetValue (
    SCDynamicStoreRef store,
    CFStringRef key,
    CFPropertyListRef value
);
```

### Parameters

*store*

The dynamic store session.

*key*

The key associated with the value.

*value*

The value to add to or replace in the dynamic store.

### Return Value

TRUE if the key was updated; otherwise, FALSE.

### Availability

Available in Mac OS X v10.1 and later.

**Declared In**

SCDynamicStore.h

## Data Types

### SCDynamicStoreCallback

Callback used when notification of changes made to the dynamic store is delivered.

```
typedef void (*SCDynamicStoreCallback) (
    SCDynamicStoreRef store,
    CFArrayRef changedKeys,
    void *info
);
```

**Fields**

store

The dynamic store session.

changedKeys

The list of changed keys.

info

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

### SCDynamicStoreContext

Structure containing user-specified data and callbacks for a dynamic store session.

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

**Fields**

version

The version number of the structure type being passed in as a parameter to the `SCDynamicStore` creation function (such as `SCDynamicStoreCreate` (page 16)). This structure is version 0.

info

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

retain

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

release

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

`copyDescription`

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

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStore.h`

**SCDynamicStoreRef**

The handle to an open dynamic store session with the system configuration daemon.

```
typedef const struct __SCDynamicStore * SCDynamicStoreRef;
```

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStore.h`

## Constants

**Dynamic Store Options Keys**

Keys that indicate the options for a dynamic store session.

```
const CFStringRef kSCDynamicStoreUseSessionKeys;
```

**Constants**

`kSCDynamicStoreUseSessionKeys`

All keys added or set into the dynamic store should be per-session keys.

Available in Mac OS X v10.4 and later.

Declared in `SCDynamicStore.h`.



# SCDynamicStoreCopyDHCPInfo Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCDynamicStoreCopyDHCPInfo.h

## Overview

The functions of the `SCDynamicStoreCopyDHCPInfo` programming interface provide access to information returned by the DHCP or BootP server.

## Functions

### DHCPInfoGetLeaseStartTime

Returns the lease start time data.

```
CFDateRef DHCPInfoGetLeaseStartTime (
    CFDictionaryRef info
);
```

#### Parameters

*info*

The DHCP information dictionary returned by `SCDynamicStoreCopyDHCPInfo` (page 26). Do not pass in a NULL dictionary.

#### Return Value

Data that corresponds to the lease start time, if this information is present, or NULL if the information is not present or if the configuration method is not DHCP.

#### Availability

Available in Mac OS X v10.2 and later.

#### Declared In

SCDynamicStoreCopyDHCPInfo.h

### DHCPInfoGetOptionData

Returns DHCP option data, if present.

```
CFDataRef DHCPInfoGetOptionData (
    CFDictionaryRef info,
    UInt8 code
);
```

**Parameters***info*

The DHCP information dictionary returned by [SCDynamicStoreCopyDHCPInfo](#) (page 26). Do not pass in a NULL dictionary.

*code*

The DHCP option code to get data for. (See RFC 2132 for more information on this code.)

**Return Value**

The DHCP option data if present, or NULL if the data is not present. You must not release the return value.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

SCDynamicStoreCopyDHCPInfo.h

**SCDynamicStoreCopyDHCPInfo**

Returns the DHCP information for the specified service.

```
CFDictionaryRef SCDynamicStoreCopyDHCPInfo (
    SCDynamicStoreRef store,
    CFStringRef serviceID
);
```

**Parameters***store*

The dynamic store session that should be used for communication with the server. If this is NULL, a temporary session is used.

*serviceID*

The service ID. Pass NULL to retrieve information for the primary service.

**Return Value**

A dictionary containing DHCP information if successful, or NULL if unsuccessful. You must use the `CFRelease` function to release return values other than NULL.

**Discussion**

Use [DHCPInfoGetOptionData](#) (page 25) to extract individual options from the dictionary returned by this function.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

SCDynamicStoreCopyDHCPInfo.h

# SCDynamicStoreCopySpecific Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCDynamicStoreCopySpecific.h

## Overview

The functions of the `SCDynamicStoreCopySpecific` programming interface allow an application to determine specific configuration information about the current system (for example, the computer or sharing name or the currently logged-in user). Note that these functions follow Core Foundation function-name conventions. A function that has "Create" or "Copy" in its name returns a reference you must release with the `CFRelease` function.

## Functions

### SCDynamicStoreCopyComputerName

Returns the current computer name.

```
CFStringRef SCDynamicStoreCopyComputerName (
    SCDynamicStoreRef store,
    CFStringEncoding *nameEncoding
);
```

#### Parameters

*store*

The dynamic store session that should be used for communication with the server. Pass `NULL` to use a temporary session.

*nameEncoding*

A pointer to memory that, on output, is filled with the encoding associated with the computer or host name, if it is non-`NULL`.

#### Return Value

The current computer name, or `NULL` if the name has not been set or if an error occurred. You must release the return value.

#### Availability

Available in Mac OS X v10.1 and later.

#### Declared In

`SCDynamicStoreCopySpecific.h`

## SCDynamicStoreCopyConsoleUser

Returns information about the user currently logged into the system.

```
CFStringRef SCDynamicStoreCopyConsoleUser (
    SCDynamicStoreRef store,
    uid_t *uid,
    gid_t *gid
);
```

### Parameters

*store*

The dynamic store session that should be used for communication with the server. Pass `NULL` to use a temporary session.

*uid*

A pointer to memory that, on output, is filled with the user ID of the currently logged-in user. If `NULL`, this value is not returned.

*gid*

A pointer to memory that, on output, is filled with the group ID of the currently logged-in user. If `NULL`, this value is not returned.

### Return Value

Returns the name, user ID, and group ID of the user currently logged into the system, or `NULL` if no user is logged in or if an error occurred. You must release the returned values.

### Discussion

Note that this function only provides information about the primary console. It does not provide any details about console sessions that have fast user switched out or about other consoles.

### Availability

Available in Mac OS X v10.1 and later.

### Declared In

`SCDynamicStoreCopySpecific.h`

## SCDynamicStoreCopyLocalHostName

Returns the current local host name.

```
CFStringRef SCDynamicStoreCopyLocalHostName (
    SCDynamicStoreRef store
);
```

### Parameters

*store*

The dynamic store session that should be used for communication with the server. Pass `NULL` to use a temporary session.

### Return Value

Returns the current local host name, or `NULL` if the name has not been set or if an error occurred. You must release the return value.

### Availability

Available in Mac OS X v10.2 and later.

**Declared In**

SCDynamicStoreCopySpecific.h

**SCDynamicStoreCopyLocation**

Returns the current location identifier.

```
CFStringRef SCDynamicStoreCopyLocation (
    SCDynamicStoreRef store
);
```

**Parameters***store*

The dynamic store session that should be used for communication with the server. Pass `NULL` to use a temporary session.

**Return Value**

Returns the current location identifier, or `NULL` if no location identifier has been defined or if an error occurred. You must release the returned value.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

SCDynamicStoreCopySpecific.h

**SCDynamicStoreCopyProxies**

Returns the key-value pairs that represent the current internet proxy settings.

```
CFDictionaryRef SCDynamicStoreCopyProxies (
    SCDynamicStoreRef store
);
```

**Parameters***store*

The dynamic store session that should be used for communication with the server. Pass `NULL` to use a temporary session.

**Return Value**

A dictionary of key-value pairs that represent the current internet proxy settings, or `NULL` if no proxy settings have been defined or if an error occurred. You must release the returned value.

**Discussion**

The returned proxy settings dictionary can include the following key-value pairs:

Key	Type	Description
kSCPropNetProxies-ExceptionsList	A CFArray of CFString objects	Host name patterns that should bypass the proxy
kSCPropNetProxiesHTTPEnable	A CFNumber with the value 0 or 1	Enables or disables the use of an HTTP proxy

Key	Type	Description
kSCPropNetProxiesHTTPProxy	CFString	The proxy host
kSCPropNetProxiesHTTPPort	CFNumber	The proxy port number
kSCPropNetProxiesHTTPSEnable	A CFNumber with the value 0 or 1	Enables or disables the use of an HTTPS proxy
kSCPropNetProxiesHTTPSPProxy	CFString	The proxy host
kSCPropNetProxiesHTTPSPort	CFNumber	The proxy port number
kSCPropNetProxiesFTPEnable	A CFNumber with the value 0 or 1	Enables or disables the use of an FTP proxy
kSCPropNetProxiesFTPProxy	CFString	The proxy host
kSCPropNetProxiesFTPPort	CFNumber	The proxy port number
kSCPropNetProxiesFTPPassive	A CFNumber with the value 0 or 1	Enables or disables passive mode operation for use behind connection filtering firewalls

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

CFFTPSample

ImageClient

**Declared In**

SCDynamicStoreCopySpecific.h

# SCDynamicStoreKey Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCDynamicStoreKey.h

## Overview

The `SCDynamicStoreKey` programming interface provides convenience functions that an application can use to create a correctly formatted dynamic store key for accessing specific items in the dynamic store. An application can then use the resulting string in any function that requires a dynamic store key.

## Functions

### SCDynamicStoreKeyCreate

Creates a dynamic store key using the specified format.

```
CFStringRef SCDynamicStoreKeyCreate (
    CFAllocatorRef allocator,
    CFStringRef fmt,
    ...
);
```

#### Parameters

*allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

*fmt*

The description of the format for this key.

#### Return Value

A string containing the formatted key.

#### Availability

Available in Mac OS X v10.1 and later.

#### Declared In

`SCDynamicStoreKey.h`

**SCDynamicStoreKeyCreateComputerName**

Creates a key that can be used to receive notifications when the current computer name changes.

```
CFStringRef SCDynamicStoreKeyCreateComputerName (
    CFAllocatorRef allocator
);
```

**Parameters**

*allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

**Return Value**

A notification string for the current computer or host name.

**Discussion**

Use this key with the [SCDynamicStoreSetNotificationKeys](#) (page 21) function.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStoreKey.h`

**SCDynamicStoreKeyCreateConsoleUser**

Creates a key that can be used to receive notifications when the current console user changes.

```
CFStringRef SCDynamicStoreKeyCreateConsoleUser (
    CFAllocatorRef allocator
);
```

**Parameters**

*allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

**Return Value**

A notification string for the current console user.

**Discussion**

Use this key with the [SCDynamicStoreSetNotificationKeys](#) (page 21) function.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStoreKey.h`

**SCDynamicStoreKeyCreateHostNames**

Creates a key that can be used to receive notifications when the `HostNames` entity changes.

```
CFStringRef SCDynamicStoreKeyCreateHostNames (
    CFAllocatorRef allocator
);
```

**Parameters***allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

**Return Value**

A notification string for the `HostNames` entity.

**Discussion**

Use this key with the [SCDynamicStoreSetNotificationKeys](#) (page 21) function. Note that the `HostNames` entity includes the local host name.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

`SCDynamicStoreKey.h`

**SCDynamicStoreKeyCreateLocation**

Creates a key that can be used to receive notifications when the location identifier changes.

```
CFStringRef SCDynamicStoreKeyCreateLocation (
    CFAllocatorRef allocator
);
```

**Parameters***allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

**Return Value**

A notification string for the current location identifier.

**Discussion**

Use this key with the [SCDynamicStoreSetNotificationKeys](#) (page 21) function.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

`SCDynamicStoreKey.h`

**SCDynamicStoreKeyCreateNetworkGlobalEntity**

Creates a dynamic store key that can be used to access a specific global (as opposed to a per-service or per-interface) network configuration entity.

```
CFStringRef SCDynamicStoreKeyCreateNetworkGlobalEntity (
    CFAllocatorRef allocator,
    CFStringRef domain,
    CFStringRef entity
);
```

**Parameters***allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

*domain*

The desired domain, such as the requested configuration or the current state.

*entity*

The specific global entity, such as IPv4 or DNS.

**Return Value**

A string containing the formatted key.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStoreKey.h`

**SCDynamicStoreKeyCreateNetworkInterface**

Creates a dynamic store key that can be used to access the network interface configuration information in the dynamic store.

```
CFStringRef SCDynamicStoreKeyCreateNetworkInterface (
    CFAllocatorRef allocator,
    CFStringRef domain
);
```

**Parameters***allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

*domain*

The desired domain, such as the requested configuration or the current state.

**Return Value**

A string containing the formatted key.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

`SCDynamicStoreKey.h`

## SCDynamicStoreKeyCreateNetworkInterfaceEntity

Creates a dynamic store key that can be used to access the per-interface network configuration information in the dynamic store.

```
CFStringRef SCDynamicStoreKeyCreateNetworkInterfaceEntity (
    CFAllocatorRef allocator,
    CFStringRef domain,
    CFStringRef ifname,
    CFStringRef entity
);
```

### Parameters

*allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

*domain*

The desired domain, such as the requested configuration or the current state.

*ifname*

The interface name or a regular expression pattern.

*entity*

The specific global entity, such as IPv4 or DNS.

### Return Value

A string containing the formatted key.

### Availability

Available in Mac OS X v10.1 and later.

### Declared In

SCDynamicStoreKey.h

## SCDynamicStoreKeyCreateNetworkServiceEntity

Creates a dynamic store key that can be used to access the per-service network configuration information.

```
CFStringRef SCDynamicStoreKeyCreateNetworkServiceEntity (
    CFAllocatorRef allocator,
    CFStringRef domain,
    CFStringRef serviceID,
    CFStringRef entity
);
```

### Parameters

*allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

*domain*

The desired domain, such as the requested configuration or the current state.

*serviceID*

The service ID or a regular expression pattern.

*entity*

The specific global entity, such as IPv4 or DNS.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCDynamicStoreKey.h

**SCDynamicStoreKeyCreateProxies**

Creates a key that can be used to receive notifications when the current network proxy settings are changed.

```
CFStringRef SCDynamicStoreKeyCreateProxies (
    CFAllocatorRef allocator
);
```

**Parameters**

*allocator*

The allocator that should be used to allocate memory for this key. This parameter may be `NULL` in which case the current default allocator is used. If this value is not a valid `CFAllocatorRef`, the behavior is undefined.

**Return Value**

A notification string for the current proxy settings.

**Discussion**

Use this key with the [SCDynamicStoreSetNotificationKeys](#) (page 21) function.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

ImageClient

**Declared In**

SCDynamicStoreKey.h

# SCNetwork Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCNetwork.h

## Overview

The `SCNetwork` programming interface contains functions an application can use to determine whether that application can reach a remote host and to notify the system of configuration changes.

A remote host is considered reachable when a data packet, sent by an application into the network stack, can leave the local computer. Note that reachability does not guarantee that the data packet will actually be received by the host.

## Functions

### SCNetworkCheckReachabilityByAddress

Determines whether the specified network address is reachable using the current network configuration. (Deprecated in Mac OS X v10.6. See the discussion for a coding alternative.)

```
Boolean SCNetworkCheckReachabilityByAddress (
    const struct sockaddr *address,
    socklen_t addrLen,
    SCNetworkConnectionFlags *flags
);
```

#### Parameters

*address*

The network address of the desired host.

*addrLen*

The length, in bytes, of the address.

*flags*

A pointer to memory that, on output, will be filled with a set of “[SCNetworkConnectionFlags](#)” (page 39) values detailing the reachability of the specified address

#### Return Value

TRUE if the network connection flags are valid; FALSE if the status could not be determined.

#### Discussion

This function is deprecated, but you can get equivalent results using the following code:

```
SCNetworkReachabilityRef target;
SCNetworkConnectionFlags flags = 0;
Boolean ok;
target = SCNetworkReachabilityCreateWithAddress(NULL, address);
ok = SCNetworkReachabilityGetFlags(target, &flags);
CFRelease(target);
```

**Availability**

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.6.

**Declared In**

SCNetwork.h

**SCNetworkCheckReachabilityByName**

Determines whether the specified network host or node name is reachable using the current network configuration. (Deprecated in Mac OS X v10.6. See the discussion for a coding alternative.)

```
Boolean SCNetworkCheckReachabilityByName (
    const char *nodename,
    SCNetworkConnectionFlags *flags
);
```

**Parameters**

*nodename*

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

*flags*

A pointer to memory that, on output, will be filled with a set of “[SCNetworkConnectionFlags](#)” (page 39) values detailing the reachability of the specified address

**Return Value**

TRUE if the network connection flags are valid; FALSE if the status could not be determined.

**Discussion**

This function is deprecated, but you can get equivalent results using the following code:

```
SCNetworkReachabilityRef target;
SCNetworkConnectionFlags flags = 0;
Boolean ok;
target = SCNetworkReachabilityCreateWithName(NULL, name);
ok = SCNetworkReachabilityGetFlags(target, &flags);
CFRelease(target);
```

**Availability**

Available in Mac OS X v10.1 and later.

Deprecated in Mac OS X v10.6.

**Declared In**

SCNetwork.h

## SCNetworkInterfaceRefreshConfiguration

Sends a notification to interested configuration agents to have them immediately retry their configuration over a particular network interface. (**Deprecated in Mac OS X v10.5.** Use the [SCNetworkInterfaceForceConfigurationRefresh](#) (page 54) function instead.)

```
Boolean SCNetworkInterfaceRefreshConfiguration (
    CFStringRef ifName
);
```

### Parameters

*ifName*

The BSD name of the network interface, such as CFSTR("en0").

### Return Value

TRUE if the notification was sent; otherwise, FALSE.

### Discussion

This function must be invoked by root (in other words, the user with uid equal to 0).

### Availability

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.5.

### Declared In

SCNetwork.h

## Constants

### Network Connection Flags

Flags that indicate whether the specified network node name or address is reachable, whether a connection is required, and whether some user intervention may be required when establishing a connection.

```
enum {
    kSCNetworkFlagsTransientConnection    = 1<<0,
    kSCNetworkFlagsReachable              = 1<<1,
    kSCNetworkFlagsConnectionRequired     = 1<<2,
    kSCNetworkFlagsConnectionAutomatic    = 1<<3,
    kSCNetworkFlagsInterventionRequired   = 1<<4,
    kSCNetworkFlagsIsLocalAddress         = 1<<16,
    kSCNetworkFlagsIsDirect                = 1<<17,
};
typedef    uint32_t    SCNetworkConnectionFlags;
```

### Constants

kSCNetworkFlagsTransientConnection

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

Available in Mac OS X v10.1 and later.

Declared in SCNetwork.h.

`kSCNetworkFlagsReachable`

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

Available in Mac OS X v10.1 and later.

Declared in `SCNetwork.h`.

`kSCNetworkFlagsConnectionRequired`

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

For example, this status would be returned for a dialup connection that was not currently active, but could handle network traffic for the target system.

Available in Mac OS X v10.1 and later.

Declared in `SCNetwork.h`.

`kSCNetworkFlagsConnectionAutomatic`

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

Any traffic directed to the specified name or address will initiate the connection.

Available in Mac OS X v10.1 and later.

Declared in `SCNetwork.h`.

`kSCNetworkFlagsInterventionRequired`

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

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

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

Available in Mac OS X v10.1 and later.

Declared in `SCNetwork.h`.

`kSCNetworkFlagsIsLocalAddress`

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

Available in Mac OS X v10.3 and later.

Declared in `SCNetwork.h`.

`kSCNetworkFlagsIsDirect`

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

Available in Mac OS X v10.3 and later.

Declared in `SCNetwork.h`.

# SCNetworkConfiguration Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCNetworkConfiguration.h

## Overview

The `SCNetworkConfiguration` programming interface provides access to the stored network configuration. The functions include providing access to the network-capable devices on the system, the network sets, network services, and network protocols. Note that these functions follow Core Foundation function-name conventions. A function that has "Create" or "Copy" in its name returns a reference you must release with the `CFRelease` function.

Note that when using the functions in this programming interface, you must call the [SCPreferencesCommitChanges](#) (page 118) function to ensure that your changes are committed to permanent storage.

## Functions by Task

### Configuring Ethernet Bond Interfaces

[SCBondInterfaceCopyAll](#) (page 46)

Returns all Ethernet bond interfaces on the system.

[SCBondInterfaceCopyAvailableMemberInterfaces](#) (page 46)

Returns all network capable devices on the system that can be added to an Ethernet bond interface.

[SCBondInterfaceCopyStatus](#) (page 46)

Returns the status of the specified Ethernet bond interface.

[SCBondInterfaceCreate](#) (page 47)

Creates a new Ethernet bond interface.

[SCBondInterfaceGetMemberInterfaces](#) (page 47)

Returns the member interfaces for the specified Ethernet bond interface.

[SCBondInterfaceGetOptions](#) (page 48)

Returns the configuration settings associated with the specified Ethernet bond interface.

[SCBondInterfaceRemove](#) (page 48)

Removes the Ethernet bond interface from the configuration.

[SCBondInterfaceSetLocalizedDisplayName](#) (page 48)

Sets the localized display name for the specified Ethernet bond interface.

[SCBondInterfaceSetMemberInterfaces](#) (page 49)

Sets the member interfaces for the specified Ethernet bond interface.

[SCBondInterfaceSetOptions](#) (page 49)

Sets the configuration settings for the specified Ethernet bond interface.

[SCBondStatusGetInterfaceStatus](#) (page 50)

Returns the status of the specified member interface of an Ethernet bond or the status of the bond as a whole.

[SCBondStatusGetMemberInterfaces](#) (page 50)

Returns the member interfaces that are represented with the Ethernet bond interface.

[SCBondStatusGetTypeID](#) (page 51)

Returns the type identifier of all `SCBondStatusRef` instances.

## Configuring Network Interfaces

[SCNetworkInterfaceCopyAll](#) (page 51)

Returns all network-capable interfaces on the system.

[SCNetworkInterfaceCopyMTU](#) (page 53)

Returns the current MTU setting and the range of allowable values for the specified network interface.

[SCNetworkInterfaceCopyMediaOptions](#) (page 52)

Returns information media options for the specified network interface.

[SCNetworkInterfaceCopyMediaSubTypeOptions](#) (page 52)

Returns a list of available media options for the specified interface configuration options and subtype.

[SCNetworkInterfaceCopyMediaSubTypes](#) (page 53)

Returns a list of available media subtypes for the specified interface configuration options.

[SCNetworkInterfaceCreateWithInterface](#) (page 54)

Creates a new network interface layered on top of the specified interface.

[SCNetworkInterfaceForceConfigurationRefresh](#) (page 54)

Sends a notification to interested network configuration agents to immediately retry their configuration.

[SCNetworkInterfaceGetBSDName](#) (page 55)

Returns the BSD interface or device name for the specified interface.

[SCNetworkInterfaceGetConfiguration](#) (page 55)

Returns the configuration settings associated with the specified interface.

[SCNetworkInterfaceGetExtendedConfiguration](#) (page 56)

Returns the extended configuration settings associated with the specified interface.

[SCNetworkInterfaceGetHardwareAddressString](#) (page 56)

Returns a displayable link layer address for the specified interface.

[SCNetworkInterfaceGetInterface](#) (page 57)

Returns the underlying interface, for layered network interfaces.

[SCNetworkInterfaceGetInterfaceType](#) (page 57)

Returns the network interface type of the specified interface.

[SCNetworkInterfaceGetLocalizedDisplayName](#) (page 58)

Returns the localized display name, such as "Ethernet" or "FireWire," for the specified interface.

[SCNetworkInterfaceGetSupportedInterfaceTypes](#) (page 58)

Identifies all of the network interface types, such as PPP, that can be layered on top of the specified interface.

[SCNetworkInterfaceGetSupportedProtocolTypes](#) (page 58)

Identifies all of the network protocol types, such as IPv4 and IPv6, that can be layered on top of the specified interface.

[SCNetworkInterfaceGetTypeID](#) (page 59)

Returns the type identifier of all `SCNetworkInterface` instances.

[SCNetworkInterfaceSetConfiguration](#) (page 59)

Stores the configuration settings for the specified interface.

[SCNetworkInterfaceSetExtendedConfiguration](#) (page 60)

Stores the extended configuration settings for the specified interface.

[SCNetworkInterfaceSetMTU](#) (page 61)

Sets the requested MTU setting for the specified network interface.

[SCNetworkInterfaceSetMediaOptions](#) (page 60)

Sets the requested media subtype and options for the specified network interface.

## Configuring Network Protocols

[SCNetworkProtocolGetConfiguration](#) (page 61)

Returns the configuration settings associated with the specified protocol.

[SCNetworkProtocolGetEnabled](#) (page 62)

Returns a Boolean value indicating whether the specified protocol is enabled.

[SCNetworkProtocolGetProtocolType](#) (page 62)

Returns the type of the specified network protocol.

[SCNetworkProtocolGetTypeID](#) (page 62)

Returns the type identifier of all `SCNetworkProtocol` instances.

[SCNetworkProtocolSetConfiguration](#) (page 63)

Stores the configuration settings for the specified network protocol.

[SCNetworkProtocolSetEnabled](#) (page 63)

Enables or disables the specified protocol.

## Configuring Network Services

[SCNetworkServiceAddProtocolType](#) (page 64)

Adds the network protocol of the specified type to the specified service.

[SCNetworkServiceCopy](#) (page 64)

Returns the network service with the specified identifier.

[SCNetworkServiceCopyAll](#) (page 65)

Returns all available network services for the specified preferences.

[SCNetworkServiceCopyProtocol](#) (page 65)

Returns the network protocol of the specified type for the specified service.

[SCNetworkServiceCopyProtocols](#) (page 65)

Returns all network protocols associated with the specified service.

- [SCNetworkServiceCreate](#) (page 66)  
Creates a new network service for the specified interface in the configuration.
- [SCNetworkServiceEstablishDefaultConfiguration](#) (page 66)  
Establishes the default configuration for the specified network service.
- [SCNetworkServiceGetEnabled](#) (page 67)  
Returns a Boolean value indicating whether the specified service is enabled.
- [SCNetworkServiceGetInterface](#) (page 67)  
Returns the network interface associated with the specified service.
- [SCNetworkServiceGetName](#) (page 68)  
Returns the user-specified name associated with the specified service.
- [SCNetworkServiceGetServiceID](#) (page 68)  
Returns the identifier for the specified service.
- [SCNetworkServiceGetTypeID](#) (page 68)  
Returns the type identifier of all `SCNetworkService` instances.
- [SCNetworkServiceRemove](#) (page 69)  
Removes the specified network service from the configuration.
- [SCNetworkServiceRemoveProtocolType](#) (page 69)  
Removes the network protocol of the specified type from the specified service.
- [SCNetworkServiceSetEnabled](#) (page 69)  
Enables or disables the specified service.
- [SCNetworkServiceSetName](#) (page 70)  
Stores the user-specified name for the specified service.

## Configuring Network Sets

- [SCNetworkSetAddService](#) (page 70)  
Adds the specified network service to the specified set.
- [SCNetworkSetContainsInterface](#) (page 71)  
Returns a Boolean value indicating whether the specified interface is represented by at least one network service in the specified set.
- [SCNetworkSetCopy](#) (page 71)  
Returns the set with the specified identifier.
- [SCNetworkSetCopyAll](#) (page 72)  
Returns all available sets for the specified preferences session.
- [SCNetworkSetCopyCurrent](#) (page 72)  
Returns the current set.
- [SCNetworkSetCopyServices](#) (page 73)  
Returns all network services associated with the specified set.
- [SCNetworkSetCreate](#) (page 73)  
Creates a new set in the configuration.
- [SCNetworkSetGetName](#) (page 74)  
Returns the user-specified name associated with the specified set.
- [SCNetworkSetGetServiceOrder](#) (page 74)  
Returns the user-specified ordering of network services within the specified set.

[SCNetworkSetGetSetID](#) (page 74)

Returns the identifier for the specified set.

[SCNetworkSetGetTypeID](#) (page 75)

Returns the type identifier of all `SCNetworkSet` instances.

[SCNetworkSetRemove](#) (page 75)

Removes the specified set from the configuration.

[SCNetworkSetRemoveService](#) (page 75)

Removes the specified network service from the specified set.

[SCNetworkSetSetCurrent](#) (page 76)

Specifies the set that should be the current set.

[SCNetworkSetSetName](#) (page 76)

Stores the user-specified name for the specified set.

[SCNetworkSetSetServiceOrder](#) (page 77)

Stores the user-specified ordering of network services for the specified set.

## Configuring VLAN Interfaces

[SCVLANInterfaceCopyAll](#) (page 77)

Returns all virtual LAN (VLAN) interfaces on the system.

[SCVLANInterfaceCopyAvailablePhysicalInterfaces](#) (page 78)

Returns the network capable devices on the system that can be associated with a virtual LAN (VLAN) interface.

[SCVLANInterfaceCreate](#) (page 78)

Creates a new virtual LAN (VLAN) interface.

[SCVLANInterfaceGetOptions](#) (page 79)

Returns the configuration settings associated with the virtual LAN (VLAN) interface.

[SCVLANInterfaceGetPhysicalInterface](#) (page 79)

Returns the physical interface for the specified virtual LAN (VLAN) interface.

[SCVLANInterfaceGetTag](#) (page 79)

Returns the tag for the specified virtual LAN (VLAN) interface.

[SCVLANInterfaceRemove](#) (page 80)

Removes the virtual LAN (VLAN) interface from the configuration.

[SCVLANInterfaceSetLocalizedDisplayName](#) (page 80)

Sets the localized display name for the specified virtual LAN (VLAN) interface.

[SCVLANInterfaceSetOptions](#) (page 81)

Sets the specified configuration settings for the specified virtual LAN (VLAN) interface.

[SCVLANInterfaceSetPhysicalInterfaceAndTag](#) (page 81)

Updates the specified virtual LAN (VLAN) interface with the specified information.

## Functions

### SCBondInterfaceCopyAll

Returns all Ethernet bond interfaces on the system.

```
CFArrayRef SCBondInterfaceCopyAll (  
    SCPreferencesRef prefs  
);
```

#### Parameters

*prefs*  
The preferences session.

#### Return Value

The list of Ethernet bond interfaces on the system. You must release the returned value.

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

SCNetworkConfiguration.h

### SCBondInterfaceCopyAvailableMemberInterfaces

Returns all network capable devices on the system that can be added to an Ethernet bond interface.

```
CFArrayRef SCBondInterfaceCopyAvailableMemberInterfaces (  
    SCPreferencesRef prefs  
);
```

#### Parameters

*prefs*  
The preferences session.

#### Return Value

The list of interfaces. You must release the returned value.

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

SCNetworkConfiguration.h

### SCBondInterfaceCopyStatus

Returns the status of the specified Ethernet bond interface.

```
SCBondStatusRef SCBondInterfaceCopyStatus (
    SCBondInterfaceRef bond
);
```

**Parameters***bond*

The Ethernet bond interface.

**Return Value**

The status associated with the interface. You must release the returned value.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceCreate**

Creates a new Ethernet bond interface.

```
SCBondInterfaceRef SCBondInterfaceCreate (
    SCPreferencesRef prefs
);
```

**Parameters***prefs*

The preferences session.

**Return Value**The new Ethernet bond interface, represented by an [SCBondInterfaceRef](#) (page 82) object. You must release the returned value.**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceGetMemberInterfaces**

Returns the member interfaces for the specified Ethernet bond interface.

```
CFArrayRef SCBondInterfaceGetMemberInterfaces (
    SCBondInterfaceRef bond
);
```

**Parameters***bond*

The Ethernet bond interface.

**Return Value**

The list of interfaces.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceGetOptions**

Returns the configuration settings associated with the specified Ethernet bond interface.

```
CFDictionaryRef SCBondInterfaceGetOptions (  
    SCBondInterfaceRef bond  
);
```

**Parameters**

*bond*

The Ethernet bond interface.

**Return Value**

The configuration settings associated with the Ethernet bond interface, or `NULL` if no changes to the default configuration have been saved.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceRemove**

Removes the Ethernet bond interface from the configuration.

```
Boolean SCBondInterfaceRemove (  
    SCBondInterfaceRef bond  
);
```

**Parameters**

*bond*

The Ethernet bond interface.

**Return Value**

TRUE if the interface was removed; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceSetLocalizedDisplayName**

Sets the localized display name for the specified Ethernet bond interface.

```
Boolean SCBondInterfaceSetLocalizedDisplayName (
    SCBondInterfaceRef bond,
    CFStringRef newName
);
```

**Parameters***bond*

The Ethernet bond interface.

*newName*

The new display name.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceSetMemberInterfaces**

Sets the member interfaces for the specified Ethernet bond interface.

```
Boolean SCBondInterfaceSetMemberInterfaces (
    SCBondInterfaceRef bond,
    CFArrayRef members
);
```

**Parameters***bond*

The Ethernet bond interface

*members*

The desired member interfaces.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondInterfaceSetOptions**

Sets the configuration settings for the specified Ethernet bond interface.

```
Boolean SCBondInterfaceSetOptions (
    SCBondInterfaceRef bond,
    CFDictionaryRef newOptions
);
```

**Parameters***bond*

The Ethernet bond interface.

*newOptions*

The new configuration settings.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondStatusGetInterfaceStatus**

Returns the status of the specified member interface of an Ethernet bond or the status of the bond as a whole.

```
CFDictionaryRef SCBondStatusGetInterfaceStatus (
    SCBondStatusRef bondStatus,
    SCNetworkInterfaceRef interface
);
```

**Parameters***bondStatus*

The Ethernet bond status.

*interface*

The member interface whose status is needed. Pass NULL to get the status of the Ethernet bond.

**Return Value**

The interface status.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondStatusGetMemberInterfaces**

Returns the member interfaces that are represented with the Ethernet bond interface.

```
CFArrayRef SCBondStatusGetMemberInterfaces (
    SCBondStatusRef bondStatus
);
```

**Parameters**

*bondStatus*

The Ethernet bond status.

**Return Value**

The list of interfaces.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCBondStatusGetTypeID**

Returns the type identifier of all `SCBondStatusRef` instances.

```
CTypeID SCBondStatusGetTypeID (
    void
);
```

**Return Value**

The type identifier of all `SCBondStatusRef` (page 82) instances.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceCopyAll**

Returns all network-capable interfaces on the system.

```
CFArrayRef SCNetworkInterfaceCopyAll (
    void
);
```

**Return Value**

The list of interfaces on the system. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCNetworkConfiguration.h

## SCNetworkInterfaceCopyMediaOptions

Returns information media options for the specified network interface.

```
Boolean SCNetworkInterfaceCopyMediaOptions (
    SCNetworkInterfaceRef interface,
    CFDictionaryRef *current,
    CFDictionaryRef *active,
    CFArrayRef *available,
    Boolean filter
);
```

### Parameters

*interface*

The network interface.

*current*

On output, a dictionary representing the currently requested media options (subtype, options). If NULL, the current options are not returned.

*active*

On output, a dictionary representing the active media options (subtype, options). If NULL, the active options are not returned.

*available*

On output, an array representing the possible media options (subtype, options). If NULL, the current options are not returned.

*filter*

A Boolean value indicating whether the available options should be filtered to exclude those options which would not normally be requested by a user/admin (for example, hw-loopback).

### Return Value

TRUE if requested information has been returned.

### Availability

Available in Mac OS X v10.5 and later.

### Declared In

SCNetworkConfiguration.h

## SCNetworkInterfaceCopyMediaSubTypeOptions

Returns a list of available media options for the specified interface configuration options and subtype.

```
CFArrayRef SCNetworkInterfaceCopyMediaSubTypeOptions (
    CFArrayRef available,
    CFStringRef subType
);
```

### Parameters

*available*

The available options as returned by the [SCNetworkInterfaceCopyMediaOptions](#) (page 52) function.

*subType*

The subtype.

**Return Value**

An array of available media options, or NULL if no options are available. Each of the available options is returned as an array of strings.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceCopyMediaSubTypes**

Returns a list of available media subtypes for the specified interface configuration options.

```
CFArrayRef SCNetworkInterfaceCopyMediaSubTypes (
    CFArrayRef available
);
```

**Parameters**

*available*

The available options as returned by the [SCNetworkInterfaceCopyMediaOptions](#) (page 52) function.

**Return Value**

An array of available media subtypes (for example, 10BaseT/UTP, 100baseTX, etc), or NULL if no subtypes are available

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceCopyMTU**

Returns the current MTU setting and the range of allowable values for the specified network interface.

```
Boolean SCNetworkInterfaceCopyMTU (
    SCNetworkInterfaceRef interface,
    int *mtu_cur,
    int *mtu_min,
    int *mtu_max
);
```

**Parameters**

*interface*

The network interface.

*mtu\_cur*

On output, the current MTU setting for the interface.

*mtu\_min*

On output, the minimum MTU setting for the interface. If negative, the minimum setting could not be determined.

*mtu\_max*

On output, the maximum MTU setting for the interface. If negative, the maximum setting could not be determined.

**Return Value**

TRUE if the requested information has been returned.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

## SCNetworkInterfaceCreateWithInterface

Creates a new network interface layered on top of the specified interface.

```
SCNetworkInterfaceRef SCNetworkInterfaceCreateWithInterface (
    SCNetworkInterfaceRef interface,
    CFStringRef interfaceType
);
```

**Parameters**

*interface*

The network interface.

*interfaceType*

The type of interface to layer on top of the network interface specified in *interface*.

**Return Value**

A reference to the new network interface. You must release the returned value.

**Discussion**

You can use this function to create a "PPP" interface on top of a "modem."

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

## SCNetworkInterfaceForceConfigurationRefresh

Sends a notification to interested network configuration agents to immediately retry their configuration.

```
Boolean SCNetworkInterfaceForceConfigurationRefresh (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The desired network interface.

**Return Value**

TRUE if the notification was sent; otherwise, FALSE.

**Discussion**

Calling this function causes the DHCP client to contact the DHCP server immediately rather than waiting until its timeout has expired. The caller uses this function to inform the system that the network infrastructure or configuration has changed.

Note that this function requires root privilege; alternatively, you can pass in an interface that is derived from a sequence of calls to:

- [SCPreferencesCreateWithAuthorization](#) (page 119)
- The appropriate `SCNetworkSetCopy...` function, such as [SCNetworkSetCopyServices](#) (page 73)
- [SCNetworkServiceGetInterface](#) (page 67)

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`SCNetworkConfiguration.h`

**SCNetworkInterfaceGetBSDName**

Returns the BSD interface or device name for the specified interface.

```
CFStringRef SCNetworkInterfaceGetBSDName (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The network interface.

**Return Value**

The BSD name associated with the interface (for example, `en0`), or `NULL` if no BSD name is available.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

MoreSCF

**Declared In**

`SCNetworkConfiguration.h`

**SCNetworkInterfaceGetConfiguration**

Returns the configuration settings associated with the specified interface.

```
CFDictionaryRef SCNetworkInterfaceGetConfiguration (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The network interface.

**Return Value**

The configuration settings associated with the interface, or `NULL` if no configuration settings are associated with the interface or an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

`SCNetworkConfiguration.h`

**SCNetworkInterfaceGetExtendedConfiguration**

Returns the extended configuration settings associated with the specified interface.

```
CFDictionaryRef SCNetworkInterfaceGetExtendedConfiguration (
    SCNetworkInterfaceRef interface,
    CFStringRef extendedType
);
```

**Parameters**

*interface*

The network interface.

*extendedType*

The type of extended information (for example, EAPOL).

**Return Value**

The configuration settings associated with the interface, or `NULL` if no configuration settings are associated with the interface or an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`SCNetworkConfiguration.h`

**SCNetworkInterfaceGetHardwareAddressString**

Returns a displayable link layer address for the specified interface.

```
CFStringRef SCNetworkInterfaceGetHardwareAddressString (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The network interface.

**Return Value**

The hardware MAC (Media Access Control) address for the interface.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceGetInterface**

Returns the underlying interface, for layered network interfaces.

```
SCNetworkInterfaceRef SCNetworkInterfaceGetInterface (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The network interface.

**Return Value**

The underlying network interface, or NULL if the specified interface is a leaf interface.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceGetInterfaceType**

Returns the network interface type of the specified interface.

```
CFStringRef SCNetworkInterfaceGetInterfaceType (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The network interface.

**Return Value**

The interface type.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceGetLocalizedDisplayName**

Returns the localized display name, such as "Ethernet" or "FireWire," for the specified interface.

```
CFStringRef SCNetworkInterfaceGetLocalizedDisplayName (
    SCNetworkInterfaceRef interface
);
```

**Parameters***interface*

The network interface.

**Return Value**

The localized display name for the interface, or NULL if no name is available.

**Availability**

Available in Mac OS X v10.4 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceGetSupportedInterfaceTypes**

Identifies all of the network interface types, such as PPP, that can be layered on top of the specified interface.

```
CFArrayRef SCNetworkInterfaceGetSupportedInterfaceTypes (
    SCNetworkInterfaceRef interface
);
```

**Parameters***interface*

The network interface.

**Return Value**

The list of network interface types supported by the specified interface, or NULL if no interface types are supported.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceGetSupportedProtocolTypes**

Identifies all of the network protocol types, such as IPv4 and IPv6, that can be layered on top of the specified interface.

```
CFArrayRef SCNetworkInterfaceGetSupportedProtocolTypes (
    SCNetworkInterfaceRef interface
);
```

**Parameters**

*interface*

The network interface.

**Return Value**

The list of network protocol types supported by the specified interface, or NULL if no protocol types are supported.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceGetTypeID**

Returns the type identifier of all SCNetworkInterface instances.

```
CFTypeID SCNetworkInterfaceGetTypeID (
    void
);
```

**Return Value**

The type identifier of all SCNetworkInterface instances.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceSetConfiguration**

Stores the configuration settings for the specified interface.

```
Boolean SCNetworkInterfaceSetConfiguration (
    SCNetworkInterfaceRef interface,
    CFDictionaryRef config
);
```

**Parameters**

*interface*

The network interface.

*config*

The configuration settings to store.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceSetExtendedConfiguration**

Stores the extended configuration settings for the specified interface.

```
Boolean SCNetworkInterfaceSetExtendedConfiguration (
    SCNetworkInterfaceRef interface,
    CFStringRef extendedType,
    CFDictionaryRef config
);
```

**Parameters**

*interface*

The network interface.

*extendedType*

The type of extended information.

*config*

The extended configuration settings to store.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceSetMediaOptions**

Sets the requested media subtype and options for the specified network interface.

```
Boolean SCNetworkInterfaceSetMediaOptions (
    SCNetworkInterfaceRef interface,
    CFStringRef subtype,
    CFArrayRef options
);
```

**Parameters**

*interface*

The network interface.

*subtype*

The media subtype to set (for example, "autoselect" or "100baseTX").

*options*

The media options to set (for example, "half-duplex" or "full-duplex"). If NULL, the active options are not returned.

**Return Value**

TRUE if the configuration was updated; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkInterfaceSetMTU**

Sets the requested MTU setting for the specified network interface.

```
Boolean SCNetworkInterfaceSetMTU (
    SCNetworkInterfaceRef interface,
    int mtu
);
```

**Parameters**

*interface*

The network interface.

*mtu*

The MTU setting.

**Return Value**

TRUE if the configuration was updated; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkProtocolGetConfiguration**

Returns the configuration settings associated with the specified protocol.

```
CFDictionaryRef SCNetworkProtocolGetConfiguration (
    SCNetworkProtocolRef protocol
);
```

**Parameters**

*protocol*

The network protocol.

**Return Value**

The configuration settings associated with the protocol, or NULL if no configuration settings are associated with the protocol or an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkProtocolGetEnabled**

Returns a Boolean value indicating whether the specified protocol is enabled.

```
Boolean SCNetworkProtocolGetEnabled (
    SCNetworkProtocolRef protocol
);
```

**Parameters**

*protocol*

The network protocol.

**Return Value**

TRUE if the protocol is enabled; otherwise, FALSE.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkProtocolGetProtocolType**

Returns the type of the specified network protocol.

```
CFStringRef SCNetworkProtocolGetProtocolType (
    SCNetworkProtocolRef protocol
);
```

**Parameters**

*protocol*

The network protocol.

**Return Value**

The protocol type.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkProtocolGetTypeID**

Returns the type identifier of all `SCNetworkProtocol` instances.

```
CFTypeID SCNetworkProtocolGetTypeID (
    void
);
```

**Return Value**

The type identifier of all `SCNetworkProtocol` instances.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkProtocolSetConfiguration**

Stores the configuration settings for the specified network protocol.

```
Boolean SCNetworkProtocolSetConfiguration (
    SCNetworkProtocolRef protocol,
    CFDictionaryRef config
);
```

**Parameters***protocol*

The network protocol.

*config*

The configuration settings to store.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkProtocolSetEnabled**

Enables or disables the specified protocol.

```
Boolean SCNetworkProtocolSetEnabled (
    SCNetworkProtocolRef protocol,
    Boolean enabled
);
```

**Parameters***protocol*

The network protocol to enable or disable.

*enabled*

TRUE if the protocol should be enabled.

**Return Value**

TRUE if the enabled status was saved; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

## SCNetworkServiceAddProtocolType

Adds the network protocol of the specified type to the specified service.

```
Boolean SCNetworkServiceAddProtocolType (
    SCNetworkServiceRef service,
    CFStringRef protocolType
);
```

### Parameters

*service*

The network service.

*protocolType*

The type of network protocol to add to the service.

### Return Value

TRUE if the protocol was added to the service; FALSE if the protocol was already present or an error occurred.

### Discussion

The protocol configuration is set to default values that are appropriate for the interface associated with the service.

### Availability

Available in Mac OS X v10.4 and later.

### Declared In

SCNetworkConfiguration.h

## SCNetworkServiceCopy

Returns the network service with the specified identifier.

```
SCNetworkServiceRef SCNetworkServiceCopy (
    SCPreferencesRef prefs,
    CFStringRef serviceID
);
```

### Parameters

*prefs*

The preferences session.

*serviceID*

The unique identifier of the network service.

### Return Value

The network service from the associated preferences, or NULL if the service ID does not exist in the preferences or if an error occurred. You must release the returned value.

### Availability

Available in Mac OS X v10.4 and later.

### Declared In

SCNetworkConfiguration.h

**SCNetworkServiceCopyAll**

Returns all available network services for the specified preferences.

```
CFArrayRef SCNetworkServiceCopyAll (
    SCPreferencesRef prefs
);
```

**Parameters**

*prefs*

The preferences session.

**Return Value**

The list of network services associated with the preferences. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceCopyProtocol**

Returns the network protocol of the specified type for the specified service.

```
SCNetworkProtocolRef SCNetworkServiceCopyProtocol (
    SCNetworkServiceRef service,
    CFStringRef protocolType
);
```

**Parameters**

*service*

The network service.

*protocolType*

The type of network protocol.

**Return Value**

The network protocol associated with the service, or NULL if this protocol has not been added or if an error occurred. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceCopyProtocols**

Returns all network protocols associated with the specified service.

```
CFArrayRef SCNetworkServiceCopyProtocols (
    SCNetworkServiceRef service
);
```

**Parameters***service*

The network service.

**Return Value**

The network protocols associated with the service. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceCreate**

Creates a new network service for the specified interface in the configuration.

```
SCNetworkServiceRef SCNetworkServiceCreate (
    SCPreferencesRef prefs,
    SCNetworkInterfaceRef interface
);
```

**Parameters***prefs*

The preferences session.

*interface*

The network interface for which to create the new service.

**Return Value**

The new network service. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceEstablishDefaultConfiguration**

Establishes the default configuration for the specified network service.

```
Boolean SCNetworkServiceEstablishDefaultConfiguration (
    SCNetworkServiceRef service
);
```

**Parameters***service*

The network service.

**Return Value**

TRUE if the configuration was updated; FALSE if an error occurred.

**Discussion**

The default configuration includes the addition of network protocols for the service (with default configuration options).

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceGetEnabled**

Returns a Boolean value indicating whether the specified service is enabled.

```
Boolean SCNetworkServiceGetEnabled (  
    SCNetworkServiceRef service  
);
```

**Parameters**

*service*

The network service.

**Return Value**

TRUE if the service is enabled; otherwise, FALSE.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceGetInterface**

Returns the network interface associated with the specified service.

```
SCNetworkInterfaceRef SCNetworkServiceGetInterface (  
    SCNetworkServiceRef service  
);
```

**Parameters**

*service*

The network service.

**Return Value**

The network interface associated with the service, or NULL if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceGetName**

Returns the user-specified name associated with the specified service.

```
CFStringRef SCNetworkServiceGetName (
    SCNetworkServiceRef service
);
```

**Parameters**

*service*

The network service.

**Return Value**

The user-specified name associated with the service.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceGetServiceID**

Returns the identifier for the specified service.

```
CFStringRef SCNetworkServiceGetServiceID (
    SCNetworkServiceRef service
);
```

**Parameters**

*service*

The network service.

**Return Value**

The service identifier.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceGetTypeID**

Returns the type identifier of all `SCNetworkService` instances.

```
CFTypeID SCNetworkServiceGetTypeID (
    void
);
```

**Return Value**

The type identifier of all `SCNetworkService` instances.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceRemove**

Removes the specified network service from the configuration.

```
Boolean SCNetworkServiceRemove (
    SCNetworkServiceRef service
);
```

**Parameters***service*

The network service to remove.

**Return Value**

TRUE if the service was removed; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceRemoveProtocolType**

Removes the network protocol of the specified type from the specified service.

```
Boolean SCNetworkServiceRemoveProtocolType (
    SCNetworkServiceRef service,
    CFStringRef protocolType
);
```

**Parameters***service*

The network service.

*protocolType*

The type of network protocol to remove from the service.

**Return Value**

TRUE if the protocol was removed to the service; FALSE if the protocol was not configured or an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceSetEnabled**

Enables or disables the specified service.

```
Boolean SCNetworkServiceSetEnabled (
    SCNetworkServiceRef service,
    Boolean enabled
);
```

**Parameters***service*

The network service to enable or disable.

*enabled*

Pass TRUE if the service should be enabled; FALSE otherwise.

**Return Value**

TRUE if the enabled status was saved; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkServiceSetName**

Stores the user-specified name for the specified service.

```
Boolean SCNetworkServiceSetName (
    SCNetworkServiceRef service,
    CFStringRef name
);
```

**Parameters***service*

The network service.

*name*

The user-defined name to associate with the service.

**Return Value**

TRUE if the name was saved; FALSE if an error occurred.

**Discussion**

Although it is not technically required, the user-specified names for all services within any given set should be unique. For this reason, an error will be returned if you attempt to name two services with the same string.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetAddService**

Adds the specified network service to the specified set.

```
Boolean SCNetworkSetAddService (
    SCNetworkSetRef set,
    SCNetworkServiceRef service
);
```

**Parameters***set*

The set (the complete configuration for a single location).

*service*

The service to add to the set.

**Return Value**

TRUE if the service was added to the set; FALSE if the service was already present or an error occurred.

**Discussion**

Prior to Mac OS X v10.5, the Network Preferences pane did not support having a single service being a member of more than one set. Therefore, an error is returned if you attempt to add a service to more than one set on a pre-10.5 system.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetContainsInterface**

Returns a Boolean value indicating whether the specified interface is represented by at least one network service in the specified set.

```
Boolean SCNetworkSetContainsInterface (
    SCNetworkSetRef set,
    SCNetworkInterfaceRef interface
);
```

**Parameters***set*

The set (the complete configuration for a single location).

*interface*

The network interface.

**Return Value**

TRUE if the interface is represented in the set; otherwise, FALSE.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetCopy**

Returns the set with the specified identifier.

```
SCNetworkSetRef SCNetworkSetCopy (
    SCPreferencesRef prefs,
    CFStringRef setID
);
```

**Parameters***prefs*

The preferences session.

*setID*

The unique identifier for the set.

**Return Value**

The network set from the associated preferences, or `NULL` if the identifier does not exist in the preferences or if an error occurred. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetCopyAll**

Returns all available sets for the specified preferences session.

```
CFArrayRef SCNetworkSetCopyAll (
    SCPreferencesRef prefs
);
```

**Parameters***prefs*

The preferences session.

**Return Value**

The list of network sets associated with the preferences. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetCopyCurrent**

Returns the current set.

```
SCNetworkSetRef SCNetworkSetCopyCurrent (
    SCPreferencesRef prefs
);
```

**Parameters***prefs*

The preferences session.

**Return Value**

The current set, or NULL if no current set has been defined.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetCopyServices**

Returns all network services associated with the specified set.

```
CFArrayRef SCNetworkSetCopyServices (
    SCNetworkSetRef set
);
```

**Parameters**

*set*

The set (the complete configuration for a single location).

**Return Value**

The list of network services associated with the set. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetCreate**

Creates a new set in the configuration.

```
SCNetworkSetRef SCNetworkSetCreate (
    SCPreferencesRef prefs
);
```

**Parameters**

*prefs*

The preferences session.

**Return Value**

The new network set. You must release the returned value.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetGetName**

Returns the user-specified name associated with the specified set.

```
CFStringRef SCNetworkSetGetName (
    SCNetworkSetRef set
);
```

**Parameters**

*set*

The set (the complete configuration for a single location).

**Return Value**

The user-specified name associated with the set.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetGetServiceOrder**

Returns the user-specified ordering of network services within the specified set.

```
CFArrayRef SCNetworkSetGetServiceOrder (
    SCNetworkSetRef set
);
```

**Parameters**

*set*

The set (the complete configuration for a single location).

**Return Value**

The ordered list of service identifiers associated with the set, or `NULL` if no service order has been specified or if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetGetSetID**

Returns the identifier for the specified set.

```
CFStringRef SCNetworkSetGetSetID (
    SCNetworkSetRef set
);
```

**Parameters**

*set*

The set (the complete configuration for a single location).

**Return Value**

The set identifier.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetGetTypeID**

Returns the type identifier of all `SCNetworkSet` instances.

```
CTypeID SCNetworkSetGetTypeID (
    void
);
```

**Return Value**

The type identifier of all `SCNetworkSet` instances.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetRemove**

Removes the specified set from the configuration.

```
Boolean SCNetworkSetRemove (
    SCNetworkSetRef set
);
```

**Parameters**

*set*

The set (the complete configuration for a single location).

**Return Value**

TRUE if the set was removed; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetRemoveService**

Removes the specified network service from the specified set.

```
Boolean SCNetworkSetRemoveService (
    SCNetworkSetRef set,
    SCNetworkServiceRef service
);
```

**Parameters***set*

The set (the complete configuration for a single location).

*service*

The service to remove.

**Return Value**

TRUE if the service was removed from the set; FALSE if the service was not already present or an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetSetCurrent**

Specifies the set that should be the current set.

```
Boolean SCNetworkSetSetCurrent (
    SCNetworkSetRef set
);
```

**Parameters***set*

The set (the complete configuration for a single location).

**Return Value**

TRUE if the current set was updated; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

**SCNetworkSetSetName**

Stores the user-specified name for the specified set.

```
Boolean SCNetworkSetSetName (
    SCNetworkSetRef set,
    CFStringRef name
);
```

**Parameters***set*

The set (the complete configuration for a single location).

*name*

The user-defined name to associate with the set.

**Return Value**

TRUE if the name was saved; FALSE if an error occurred.

**Discussion**

Although it is not technically required, the user-specified names for all sets should be unique. For this reason, an error is returned if you attempt to name two sets with the same string.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

## SCNetworkSetSetServiceOrder

Stores the user-specified ordering of network services for the specified set.

```
Boolean SCNetworkSetSetServiceOrder (
    SCNetworkSetRef set,
    CFArrayRef newOrder
);
```

**Parameters**

*set*

The set (the complete configuration for a single location).

*newOrder*

The ordered list of service identifiers for the set.

**Return Value**

TRUE if the new service order was saved; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCNetworkConfiguration.h

## SCVLANInterfaceCopyAll

Returns all virtual LAN (VLAN) interfaces on the system.

```
CFArrayRef SCVLANInterfaceCopyAll (
    SCPreferencesRef prefs
);
```

**Parameters**

*prefs*

The preferences session.

**Return Value**

The list of VLAN interfaces on the system. You must release the returned value.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceCopyAvailablePhysicalInterfaces**

Returns the network capable devices on the system that can be associated with a virtual LAN (VLAN) interface.

```
CFArrayRef SCVLANInterfaceCopyAvailablePhysicalInterfaces (
    void
);
```

**Return Value**

The list of interfaces. You must release the returned value.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceCreate**

Creates a new virtual LAN (VLAN) interface.

```
SCVLANInterfaceRef SCVLANInterfaceCreate (
    SCPreferencesRef prefs,
    SCNetworkInterfaceRef physical,
    CFNumberRef tag
);
```

**Parameters**

*prefs*

The preferences session.

*physical*

The physical interface to associate with the VLAN.

*tag*

The tag to associate with the VLAN. This value must be between 1 and 4094.

**Return Value**

The new VLAN interface. You must release the returned value.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceGetOptions**

Returns the configuration settings associated with the virtual LAN (VLAN) interface.

```
CFDictionaryRef SCVLANInterfaceGetOptions (
    SCVLANInterfaceRef vlan
);
```

**Parameters**

*vlan*

The VLAN interface.

**Return Value**

The configuration settings associated with the VLAN interface, or NULL if no changes to the default configuration have been saved.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceGetPhysicalInterface**

Returns the physical interface for the specified virtual LAN (VLAN) interface.

```
SCNetworkInterfaceRef SCVLANInterfaceGetPhysicalInterface (
    SCVLANInterfaceRef vlan
);
```

**Parameters**

*vlan*

The VLAN interface.

**Return Value**

The list of interfaces.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceGetTag**

Returns the tag for the specified virtual LAN (VLAN) interface.

```
CFNumberRef SCVLANInterfaceGetTag (
    SCVLANInterfaceRef vlan
);
```

**Parameters**

*vlan*

The VLAN interface.

**Return Value**

The tag for the VLAN interface.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceRemove**

Removes the virtual LAN (VLAN) interface from the configuration.

```
Boolean SCVLANInterfaceRemove (
    SCVLANInterfaceRef vlan
);
```

**Parameters**

*vlan*

The VLAN interface to remove.

**Return Value**

TRUE if the interface was removed; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceSetLocalizedDisplayName**

Sets the localized display name for the specified virtual LAN (VLAN) interface.

```
Boolean SCVLANInterfaceSetLocalizedDisplayName (
    SCVLANInterfaceRef vlan,
    CFStringRef newName
);
```

**Parameters**

*vlan*

The VLAN interface.

*newName*

The new display name.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceSetOptions**

Sets the specified configuration settings for the specified virtual LAN (VLAN) interface.

```
Boolean SCVLANInterfaceSetOptions (
    SCVLANInterfaceRef vlan,
    CFDictionaryRef newOptions
);
```

**Parameters**

*vlan*

The VLAN interface.

*newOptions*

The new configuration settings for the VLAN interface.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

**SCVLANInterfaceSetPhysicalInterfaceAndTag**

Updates the specified virtual LAN (VLAN) interface with the specified information.

```
Boolean SCVLANInterfaceSetPhysicalInterfaceAndTag (
    SCVLANInterfaceRef vlan,
    SCNetworkInterfaceRef physical,
    CFNumberRef tag
);
```

**Parameters**

*vlan*

The VLAN interface to update.

*physical*

The physical interface to associate with the VLAN interface.

*tag*

The tag to associate with the VLAN interface. This value must be between 1 and 4094.

**Return Value**

TRUE if the configuration was stored; FALSE if an error occurred.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

SCNetworkConfiguration.h

## Data Types

### SCNetworkInterfaceRef

The reference to an object that represents a network interface.

```
typedef const struct __SCNetworkInterface * SCNetworkInterfaceRef;
```

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

SCNetworkConfiguration.h

### SCBondInterfaceRef

The reference to an object that represents an Ethernet bond interface.

```
typedef SCNetworkInterfaceRef SCBondInterfaceRef;
```

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

SCNetworkConfiguration.h

### SCBondStatusRef

The reference to an object that represents the status of an Ethernet bond interface.

```
typedef const struct __SCBondStatus * SCBondStatusRef;
```

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

SCNetworkConfiguration.h

### SCVLANInterfaceRef

The reference to an object that represents a virtual LAN (VLAN) interface.

```
typedef SCNetworkInterfaceRef SCVLANInterfaceRef;
```

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

SCNetworkConfiguration.h

### SCNetworkProtocolRef

The reference to an object that represents a network protocol.

```
typedef const struct __SCNetworkProtocol * SCNetworkProtocolRef;
```

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

SCNetworkConfiguration.h

### SCNetworkServiceRef

The reference to an object that represents a network service.

```
typedef const struct __SCNetworkService * SCNetworkServiceRef;
```

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

SCNetworkConfiguration.h

### SCNetworkSetRef

The reference to an object that represents a network set.

```
typedef const struct __SCNetworkSet * SCNetworkSetRef;
```

#### Availability

Available in Mac OS X v10.4 and later.

#### Declared In

SCNetworkConfiguration.h

## Constants

### Ethernet Bond Aggregation Status

Ethernet bond aggregation status codes.

```
enum {
    kSCBondStatusOK = 0,
    kSCBondStatusLinkInvalid = 1,
    kSCBondStatusNoPartner = 2,
    kSCBondStatusNotInActiveGroup = 3,
    kSCBondStatusUnknown = 999
};
```

**Constants**

kSCBondStatusOK

The status is valid (for example, enabled, active, running, and so forth.)

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

kSCBondStatusLinkInvalid

The link state is not valid (such as down, half-duplex, or wrong speed).

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

kSCBondStatusNoPartner

The port on the switch to which the device is connected doesn't seem to have 802.3ad Link Aggregation enabled.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

kSCBondStatusNotInActiveGroup

Communication with a partner is occurring, but the link aggregation group is different from the one that is active.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

kSCBondStatusUnknown

Nonspecific failure.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

**Ethernet Bond Status Constants**

Ethernet bond status codes.

```
const CFStringRef kSCBondStatusDeviceAggregationStatus;
const CFStringRef kSCBondStatusDeviceCollecting;
const CFStringRef kSCBondStatusDeviceDistributing;
```

**Constants**

kSCBondStatusDeviceAggregationStatus

Device is aggregating. See [“Ethernet Bond Aggregation Status”](#) (page 83) for a list of possible values.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCBondStatusDeviceCollecting`

Can be 0 or 1.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCBondStatusDeviceDistributing`

Can be 0 or 1.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

## Network Interface Types

Keys that identify network interface types.

```
const CFStringRef kSCNetworkInterfaceType6to4;
const CFStringRef kSCNetworkInterfaceTypeBluetooth;
const CFStringRef kSCNetworkInterfaceTypeBond;
const CFStringRef kSCNetworkInterfaceTypeEthernet;
const CFStringRef kSCNetworkInterfaceTypeFireWire;
const CFStringRef kSCNetworkInterfaceTypeIEEE80211;
const CFStringRef kSCNetworkInterfaceTypeIPSec;
const CFStringRef kSCNetworkInterfaceTypeIrDA;
const CFStringRef kSCNetworkInterfaceTypeL2TP;
const CFStringRef kSCNetworkInterfaceTypeModem;
const CFStringRef kSCNetworkInterfaceTypePPP;
const CFStringRef kSCNetworkInterfaceTypePPTP;
const CFStringRef kSCNetworkInterfaceTypeSerial;
const CFStringRef kSCNetworkInterfaceTypeVLAN;
const CFStringRef kSCNetworkInterfaceTypeWWAN;
const CFStringRef kSCNetworkInterfaceTypeIPv4;
const SCNetworkInterfaceRef kSCNetworkInterfaceIPv4;
```

### Constants

`kSCNetworkInterfaceType6to4`

The 6to4 interface.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkInterfaceTypeBluetooth`

The Bluetooth interface.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkInterfaceTypeBond`

The Ethernet bond interface.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkInterfaceTypeEthernet`

The Ethernet interface.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

- `kSCNetworkInterfaceTypeFireWire`  
The FireWire interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeIEEE80211`  
The IEEE 802.11 interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeIPSec`  
The IPSec interface.  
Available in Mac OS X v10.6 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeIrDA`  
The IrDA interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeL2TP`  
The L2TP interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeModem`  
The modem interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypePPP`  
The PPP interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypePPTP`  
The PPTP interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeSerial`  
The serial interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.
- `kSCNetworkInterfaceTypeVLAN`  
The VLAN interface.  
Available in Mac OS X v10.4 and later.  
Declared in `SCNetworkConfiguration.h`.

`kSCNetworkInterfaceTypeWWAN`

The WWAN interface.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkInterfaceTypeIPv4`

The IPv4 interface.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkInterfaceIPv4`

A network interface that can be used for layering other interfaces (for example, 6to4, PPTP, or L2TP) over an existing IPv4 network.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

## Network Protocol Types

Keys that identify network protocol types.

```
const CFStringRef kSCNetworkProtocolTypeAppleTalk;
const CFStringRef kSCNetworkProtocolTypeDNS;
const CFStringRef kSCNetworkProtocolTypeIPv4;
const CFStringRef kSCNetworkProtocolTypeIPv6;
const CFStringRef kSCNetworkProtocolTypeProxies;
const CFStringRef kSCNetworkProtocolTypeSMB;
```

### Constants

`kSCNetworkProtocolTypeAppleTalk`

The AppleTalk protocol.

Available in Mac OS X v10.4 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkProtocolTypeDNS`

The DNS protocol.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkProtocolTypeIPv4`

The IPv4 protocol.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

`kSCNetworkProtocolTypeIPv6`

The IPv6 protocol.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

kSCNetworkProtocolTypeProxies

Protocol proxies.

Available in Mac OS X v10.4 and later.

Declared in `SCNetworkConfiguration.h`.

kSCNetworkProtocolTypeSMB

The SMB protocol.

Available in Mac OS X v10.5 and later.

Declared in `SCNetworkConfiguration.h`.

# SCNetworkConnection Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCNetworkConnection.h

## Overview

The `SCNetworkConnection` programming interface contains functions that allow an application to control connection-oriented services defined in the system and get connection-status information. Note that these functions allow you to control and get information about existing services only. If you need to create, change, or remove services, you should use the `SCNetworkConfiguration` programming interface instead.

**Note:** Currently, only PPP services can be controlled.

## Functions by Task

### Getting Connection-Status Information

[SCNetworkConnectionGetTypeID](#) (page 95)

Returns the type identifier of all `SCNetworkConnection` instances.

[SCNetworkConnectionCopyUserPreferences](#) (page 93)

Provides the default service ID and a dictionary of user options for the specified connection.

[SCNetworkConnectionGetStatus](#) (page 94)

Returns the status of the specified network connection.

[SCNetworkConnectionCopyExtendedStatus](#) (page 90)

Returns the extended status of the connection.

[SCNetworkConnectionCopyStatistics](#) (page 91)

Returns the statistics of the specified connection.

[SCNetworkConnectionCopyUserOptions](#) (page 92)

Gets the user options used to start the specified connection.

[SCNetworkConnectionCopyServiceID](#) (page 91) **Deprecated in Mac OS X v10.6**

Returns the service ID associated with the specified network connection.

## Starting and Stopping a Connection

[SCNetworkConnectionStart](#) (page 96)

Starts the connection process for the specified network connection.

[SCNetworkConnectionStop](#) (page 97)

Stops the connection process for the specified network connection.

## Scheduling a Connection Reference on a Run Loop

[SCNetworkConnectionScheduleWithRunLoop](#) (page 95)

Schedules the specified connection with the specified run loop.

[SCNetworkConnectionUnscheduleFromRunLoop](#) (page 98)

Unschedules the specified connection from the specified run loop.

## Creating a Connection Reference

[SCNetworkConnectionCreateWithServiceID](#) (page 94)

Creates a new connection reference to use for getting the status or for connecting or disconnecting the associated service.

## Specifying a Dispatch Queue and Enabling Notifications

[SCNetworkConnectionSetDispatchQueue](#) (page 96)

Specifies a dispatch queue to use for the connection's callback function and enables notifications.

# Functions

### SCNetworkConnectionCopyExtendedStatus

Returns the extended status of the connection.

```
CFDictionaryRef SCNetworkConnectionCopyExtendedStatus (
    SCNetworkConnectionRef connection
);
```

#### Parameters

*connection*

The network connection.

#### Return Value

The status dictionary, or NULL if an error occurred (use the [SCError](#) (page 169) function to retrieve the specific error).

#### Discussion

An extended status dictionary contains specific dictionaries describing the status for each subcomponent of the service. For example, a status dictionary contains the following sub-dictionaries, keys, and values:

Sub-dictionary	Key	Value
IPv4	Addresses	The assigned IP address
PPP	Status	The PPP-specific status (see “ <a href="#">Network Connection PPP Status Values</a> ” (page 100) for possible values)
	LastCause	Available when the status is “Disconnected” and contains the last error associated with connecting or disconnecting.
	ConnectTime	The time when the connection was established.
Modem	ConnectSpeed	The speed of the modem connection in bits per second.

Other dictionaries can be present for PpOE, PPTP, and L2TP.

The status dictionary may be extended in the future to contain additional information.

#### Availability

Available in Mac OS X v10.3 and later.

#### Related Sample Code

SimpleDial

#### Declared In

SCNetworkConnection.h

### SCNetworkConnectionCopyServiceID

Returns the service ID associated with the specified network connection.

```
CFStringRef SCNetworkConnectionCopyServiceID (
    SCNetworkConnectionRef connection
);
```

#### Parameters

*connection*

The network connection.

#### Return Value

The service ID associated with the network connection.

#### Availability

Available in Mac OS X v10.3 and later.

#### Declared In

SCNetworkConnection.h

### SCNetworkConnectionCopyStatistics

Returns the statistics of the specified connection.

```
CFDictionaryRef SCNetworkConnectionCopyStatistics (
    SCNetworkConnectionRef connection
);
```

**Parameters**

*connection*

The network connection.

**Return Value**

The statistics dictionary, or NULL if an error occurred (use the [SCError](#) (page 169) function to retrieve the specific error).

**Discussion**

A statistics dictionary contains specific dictionaries with statistics for each subcomponent of the service. For example, a statistics dictionary for PPP contains the following sub-dictionaries, keys, and values:

Sub-dictionary	Key	Value
PPP	BytesIn	The number of bytes going up into the network stack for any networking protocol without the PPP headers and trailers.
PPP	BytesOut	The number of bytes coming out of the network stack for any networking protocol without the PPP headers and trailers.
PPP	PacketsIn	The number of packets going up into the network stack for any networking protocol without the PPP headers and trailers.
PPP	PacketsOut	The number of packets coming out of the network stack for any networking protocol without the PPP headers and trailers.
PPP	ErrorsIn	The number of errors going up into the network stack for any networking protocol without the PPP headers and trailers.
PPP	ErrorsOut	The number of errors coming out of the network stack for any networking protocol without the PPP headers and trailers.

See [“Statistics Dictionary Keys”](#) (page 102) for the keys to use in the statistics dictionary.

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionCopyUserOptions**

Gets the user options used to start the specified connection.

```
CFDictionaryRef SCNetworkConnectionCopyUserOptions (
    SCNetworkConnectionRef connection
);
```

**Parameters***connection*

The network connection.

**Return Value**The service dictionary containing the connection options (the dictionary can be empty if no user options were used), or NULL if an error occurred (use the [SCError](#) (page 169) function to retrieve the specific error).**Discussion**A client can call this function to retrieve the user options previously passed to the [SCNetworkConnectionStart](#) (page 96) function.**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionCopyUserPreferences**

Provides the default service ID and a dictionary of user options for the specified connection.

```
Boolean SCNetworkConnectionCopyUserPreferences (
    CFDictionaryRef selectionOptions,
    CFStringRef *serviceID,
    CFDictionaryRef *userOptions
);
```

**Parameters***selectionOptions*

Currently unimplemented. Pass NULL.

*serviceID*

On output, a reference to the default service ID for starting connections.

*userOptions*

On output, a reference to the default user options dictionary for starting connections.

**Return Value**

TRUE if there is a valid service to dial; FALSE if the function was unable to retrieve a service to dial.

**Discussion**

You can use the service ID and user options values returned by this function to open a connection on the fly.

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleDial

**Declared In**

SCNetworkConnection.h

## SCNetworkConnectionCreateWithServiceID

Creates a new connection reference to use for getting the status or for connecting or disconnecting the associated service.

```

SCNetworkConnectionRef SCNetworkConnectionCreateWithServiceID (
    CFAllocatorRef allocator,
    CFStringRef serviceID,
    SCNetworkConnectionCallback callout,
    SCNetworkConnectionContext *context
);

```

### Parameters

*allocator*

The allocator that should be used to allocate memory for the connection structure. This parameter may be NULL, in which case the current default allocator is used. If this reference is not a valid allocator, the behavior is undefined.

*serviceID*

The service identifier of the connection. This value uniquely identifies service in the system configuration database.

*callout*

The function to be called when the status of the connection changes. If this parameter is NULL, the application receives notifications of status change and will need to poll for updates.

*context*

User-specified data associated with the connection.

### Return Value

A reference to a new network connection.

### Availability

Available in Mac OS X v10.3 and later.

### Related Sample Code

SimpleDial

### Declared In

SCNetworkConnection.h

## SCNetworkConnectionGetStatus

Returns the status of the specified network connection.

```

SCNetworkConnectionStatus SCNetworkConnectionGetStatus (
    SCNetworkConnectionRef connection
);

```

### Parameters

*connection*

The network connection.

### Return Value

The status of the network connection. See [“Network Connection Status Values”](#) (page 100) for a list of possible values.

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleDial

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionGetTypeID**

Returns the type identifier of all `SCNetworkConnection` instances.

```
CFTypeID SCNetworkConnectionGetTypeID (
    void
);
```

**Return Value**

The type identifier of all `SCNetworkConnection` instances.

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionScheduleWithRunLoop**

Schedules the specified connection with the specified run loop.

```
Boolean SCNetworkConnectionScheduleWithRunLoop (
    SCNetworkConnectionRef connection,
    CFSRunLoopRef runLoop,
    CFStringRef runLoopMode
);
```

**Parameters**

*connection*

The network connection to schedule.

*runLoop*

The run loop with which to schedule the network connection.

*runLoopMode*

The run loop mode.

**Return Value**

TRUE if the connection is scheduled successfully; FALSE (use the [SCError](#) (page 169) function to retrieve the specific error).

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleDial

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionSetDispatchQueue**

Specifies a dispatch queue to use for the connection's callback function and enables notifications.

```
Boolean SCNetworkConnectionSetDispatchQueue (
    SCNetworkConnectionRef connection,
    dispatch_queue_t queue
);
```

**Parameters***connection*

The network connection to notify.

*queue*

The queue on which to run the connection's callback function. Pass `NULL` to disable notifications and release the queue.

**Return Value**

TRUE if successful; otherwise, FALSE (use the [SCError](#) (page 169) function to retrieve the specific error).

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionStart**

Starts the connection process for the specified network connection.

```
Boolean SCNetworkConnectionStart (
    SCNetworkConnectionRef connection,
    CFDictionaryRef userOptions,
    Boolean linger
);
```

**Parameters***connection*

The network connection to start.

*userOptions*

The options with which to start the connection. If *userOptions* is `NULL`, the default settings are used. If *userOptions* are specified, they must be in the same format as network services stored in the system configuration preferences schema. The options override the default settings defined for the service.

For security reasons, not all options can be overridden; the appropriate merging of all settings is done before the connection is established, and inappropriate options are ignored.

*linger*

A Boolean value indicating whether the connection can persist when the application no longer has interest in it. A typical application should pass `FALSE`, in which case the connection is automatically stopped when the reference is released or the application quits. If the application passes `TRUE`, the application can release the reference or exit and the connection is maintained until a timeout event, until a specific stop request occurs, or until an error occurs.

**Return Value**

`TRUE` if the connection was correctly started (the actual connection is not established yet, and the connection status needs to be periodically checked); `FALSE` if the connection request was not started (use the [SCError](#) (page 169) function to retrieve the specific error).

**Discussion**

The connection process is asynchronous and this function returns immediately. The connection status can be obtained by polling or by callback. The connection is made with the default settings from the administrator. Some of the settings can be overridden for the duration of the connection. These are specified in an options dictionary. The options dictionary uses the same format as a network service defined in the system configuration preferences schema.

**Note:** Starting and stopping of connections is implicitly arbitrated. Calling `SCNetworkConnectionStart` on a connection already started indicates that the application has interest in the connection and it shouldn't be stopped by anyone else.

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleDial

**Declared In**

`SCNetworkConnection.h`

**SCNetworkConnectionStop**

Stops the connection process for the specified network connection.

```
Boolean SCNetworkConnectionStop (
    SCNetworkConnectionRef connection,
    Boolean forceDisconnect
);
```

**Parameters**

*connection*

The network connection to stop.

*forceDisconnect*

Pass `TRUE` to stop the connection regardless of other applications that might have interest in it.

**Return Value**

`TRUE` if the disconnection request succeeded; `FALSE` (use the [SCError](#) (page 169) function to retrieve the specific error).

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleDial

**Declared In**

SCNetworkConnection.h

**SCNetworkConnectionUnscheduleFromRunLoop**

Unschedules the specified connection from the specified run loop.

```
Boolean SCNetworkConnectionUnscheduleFromRunLoop (
    SCNetworkConnectionRef connection,
    CFRunLoopRef runLoop,
    CFStringRef runLoopMode
);
```

**Parameters***connection*

The network connection to unschedule.

*runLoop*

The run loop from which to unschedule the network connection.

*runLoopMode*

The run loop mode.

**Return Value**TRUE if the connection is unscheduled successfully; FALSE (use the [SCError](#) (page 169) function to retrieve the specific error).**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleDial

**Declared In**

SCNetworkConnection.h

## Data Types

**SCNetworkConnectionRef**

The handle to manage a connection-oriented service.

```
typedef const struct __SCNetworkConnection * SCNetworkConnectionRef;
```

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

SCNetworkConnection.h

## SCNetworkConnectionCallback

The type of callback function used when a status event is delivered.

```
typedef void (*SCNetworkConnectionCallback) (
    SCNetworkConnectionRef connection,
    SCNetworkConnectionStatus status,
    void *info
);
```

### Fields

connection

The network connection.

status

The connection status.

info

Application-specific information.

## SCNetworkConnectionContext

A structure containing user-specified data and callbacks for a network connection.

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

### Fields

version

The version number of the structure type being passed in as a parameter to the [SCNetworkConnectionCreateWithServiceID](#) (page 94) function. This structure is version 0.

info

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

retain

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

release

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

copyDescription

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

### Availability

Available in Mac OS X v10.3 and later.

### Declared In

SCNetworkConnection.h

## Constants

### Network Connection Status Values

The current status of the network connection.

```
enum {
    kSCNetworkConnectionInvalid = -1,
    kSCNetworkConnectionDisconnected = 0,
    kSCNetworkConnectionConnecting = 1,
    kSCNetworkConnectionConnected = 2,
    kSCNetworkConnectionDisconnecting = 3
};
typedef int32_t SCNetworkConnectionStatus;
```

#### Constants

`kSCNetworkConnectionInvalid`  
 The network connection refers to an invalid service.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionDisconnected`  
 The network connection is disconnected.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionConnecting`  
 The network connection is connecting.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionConnected`  
 The network connection is connected.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionDisconnecting`  
 The network connection is disconnecting.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.

#### Discussion

This status is intended to be generic and high level. An extended status, specific to the type of network connection, is also available for applications that need additional information.

### Network Connection PPP Status Values

The PPP-specific status of the network connection.

```
enum {
    kSCNetworkConnectionPPPDisconnected = 0,
    kSCNetworkConnectionPPPInitializing = 1,
    kSCNetworkConnectionPPPConnectingLink = 2,
    kSCNetworkConnectionPPPDialOnTraffic = 3,
    kSCNetworkConnectionPPPNegotiatingLink = 4,
    kSCNetworkConnectionPPPAuthenticating = 5,
    kSCNetworkConnectionPPPWaitingForCallBack = 6,
    kSCNetworkConnectionPPPNegotiatingNetwork = 7,
    kSCNetworkConnectionPPPConnected = 8,
    kSCNetworkConnectionPPPTerminating = 9,
    kSCNetworkConnectionPPPDisconnectingLink = 10,
    kSCNetworkConnectionPPP HoldingLinkOff = 11,
    kSCNetworkConnectionPPPSuspended = 12,
    kSCNetworkConnectionPPPWaitingForRedial = 13
};
typedef int32_t SCNetworkConnectionPPPStatus;
```

**Constants**

- `kSCNetworkConnectionPPPDisconnected`  
**PPP is disconnected.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.
- `kSCNetworkConnectionPPPInitializing`  
**PPP is initializing.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.
- `kSCNetworkConnectionPPPConnectingLink`  
**PPP is connecting the lower connection layer (for example, the modem is dialing out).**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.
- `kSCNetworkConnectionPPPDialOnTraffic`  
**PPP is waiting for networking traffic to automatically establish the connection.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.
- `kSCNetworkConnectionPPPNegotiatingLink`  
**The PPP lower layer is connected and PPP is negotiating the link layer (LCP protocol).**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.
- `kSCNetworkConnectionPPPAuthenticating`  
**PPP is authenticating to the server (PAP, CHAP, MS-CHAP, or EAP protocols).**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.
- `kSCNetworkConnectionPPPWaitingForCallBack`  
**PPP is waiting for the server to call back.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPPNegotiatingNetwork`

PPP is now authenticated and negotiating the networking layer (IPCP or IPv6CP protocols).

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPPConnected`

PPP is now fully connected for at least one networking layer. Additional networking protocol might still be negotiating.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPPTerminating`

PPP networking and link protocols are terminating.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPDisconnectingLink`

PPP is disconnecting the lower level (for example, the modem is hanging up).

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPP HoldingLinkOff`

PPP is disconnected and maintaining the link temporarily off.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPPSuspended`

PPP is suspended as a result of the suspend command (for example, when a V.92 Modem is On Hold).

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPPPWaitingForRedial`

PPP has found a busy server and is waiting for redial.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

### Discussion

This status is returned as part of the extended information for a PPP service. Note that additional status might be returned in the future. Therefore, your application should be prepared to receive an unknown value.

## Statistics Dictionary Keys

Keys associated with values in the statistics dictionary.

```
#define kSCNetworkConnectionBytesIn      CFSTR("BytesIn")
#define kSCNetworkConnectionBytesOut    CFSTR("BytesOut")
#define kSCNetworkConnectionPacketsIn   CFSTR("PacketsIn")
#define kSCNetworkConnectionPacketsOut  CFSTR("PacketsOut")
#define kSCNetworkConnectionErrorsIn    CFSTR("ErrorsIn")
#define kSCNetworkConnectionErrorsOut   CFSTR("ErrorsOut")
```

**Constants**

`kSCNetworkConnectionBytesIn`

The key associated with the number of bytes going up into the network stack for any networking protocol without the PPP headers and trailers.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionBytesOut`

The key associated with the number of bytes coming out of the network stack for any networking protocol without the PPP headers and trailers.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPacketsIn`

The key associated with the number of packets going up into the network stack for any networking protocol without the PPP headers and trailers.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionPacketsOut`

The key associated with the number of packets coming out of the network stack for any networking protocol without the PPP headers and trailers.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionErrorsIn`

The key associated with the number of errors going up into the network stack for any networking protocol without the PPP headers and trailers.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

`kSCNetworkConnectionErrorsOut`

The key associated with the number of errors coming out of the network stack for any networking protocol without the PPP headers and trailers.

Available in Mac OS X v10.3 and later.

Declared in `SCNetworkConnection.h`.

**Selection Options Dictionary Keys**

Keys used with the [SCNetworkConnectionCopyUserPreferences](#) (page 93) selection options dictionary.

```
#define kSCNetworkConnectionSelectionOptionOnDemandHostName  
CFSTR("OnDemandHostName")  
#define kSCNetworkConnectionSelectionOptionOnDemandRetry    CFSTR("OnDemandRetry")
```

**Constants**

kSCNetworkConnectionSelectionOptionOnDemandHostName

The key associated with a host name used to select the "best" network connection.

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkConnection.h`.

kSCNetworkConnectionSelectionOptionOnDemandRetry

The key associated with a Boolean value used to indicate whether a DNS query has already been issued for the specified on-demand host name.

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkConnection.h`.

# SCNetworkReachability Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCNetworkReachability.h

## Overview

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

The `SCNetworkReachability` programming interface supports a synchronous and an asynchronous model. In the synchronous model, you get the reachability status by calling the [SCNetworkReachabilityGetFlags](#) (page 108) function. In the asynchronous model, you can schedule the `SCNetworkReachability` object on the run loop of a client object's thread. The client implements a callback function to receive notifications when the reachability status of a given remote host changes. Note that these functions follow Core Foundation naming conventions. A function that has "Create" or "Copy" in its name returns a reference you must release with the `CFRelease` function.

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

## Functions by Task

### Creating a Reachability Reference

- [SCNetworkReachabilityCreateWithAddress](#) (page 106)  
Creates a reachability reference to the specified network address.
- [SCNetworkReachabilityCreateWithAddressPair](#) (page 106)  
Creates a reachability reference to the specified network address.
- [SCNetworkReachabilityCreateWithName](#) (page 107)  
Creates a reachability reference to the specified network host or node name.

### Determining Reachability Status

- [SCNetworkReachabilityGetFlags](#) (page 108)  
Determines if the specified network target is reachable using the current network configuration.

## Preparing to Determine Reachability

[SCNetworkReachabilityGetTypeID](#) (page 108)

Returns the type identifier of all `SCNetworkReachability` instances.

[SCNetworkReachabilitySetCallback](#) (page 109)

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

[SCNetworkReachabilityScheduleWithRunLoop](#) (page 109)

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

[SCNetworkReachabilityUnscheduleFromRunLoop](#) (page 110)

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

## Functions

### SCNetworkReachabilityCreateWithAddress

Creates a reachability reference to the specified network address.

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

#### Parameters

*allocator*

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

*address*

The address of the desired host.

#### Return Value

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

#### Discussion

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

#### Availability

Available in Mac OS X v10.3 and later.

#### Declared In

`SCNetworkReachability.h`

### SCNetworkReachabilityCreateWithAddressPair

Creates a reachability reference to the specified network address.

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

**Parameters***allocator*

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

*localAddress*

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

*remoteAddress*

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

**Return Value**

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

**Discussion**

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

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

`SCNetworkReachability.h`

**SCNetworkReachabilityCreateWithName**

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

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

**Parameters***allocator*

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

*nodename*

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

**Return Value**

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

**Discussion**

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

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleReach

**Declared In**

SCNetworkReachability.h

**SCNetworkReachabilityGetFlags**

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

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

**Parameters***target*

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

*flags*

A pointer to memory that, on output, is filled with flags that describe the reachability of the specified target. (See “[Network Reachability Flags](#)” (page 112) for possible values.)

**Return Value**

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

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleReach

**Declared In**

SCNetworkReachability.h

**SCNetworkReachabilityGetTypeID**

Returns the type identifier of all `SCNetworkReachability` instances.

```
CTypeID SCNetworkReachabilityGetTypeID (
    void
);
```

**Return Value**

The type identifier of all `SCNetworkReachability` instances.

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

SCNetworkReachability.h

## SCNetworkReachabilityScheduleWithRunLoop

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

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

### Parameters

*target*

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

*runLoop*

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

*runLoopMode*

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

### Return Value

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

### Availability

Available in Mac OS X v10.3 and later.

### Related Sample Code

SimpleReach

### Declared In

SCNetworkReachability.h

## SCNetworkReachabilitySetCallback

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

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

### Parameters

*target*

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

*callout*

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

*context*

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

### Return Value

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

### Availability

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleReach

**Declared In**

SCNetworkReachability.h

**SCNetworkReachabilityUnscheduleFromRunLoop**

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

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

**Parameters***target*

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

*runLoop*

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

*runLoopMode*

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

**Return Value**

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

**Availability**

Available in Mac OS X v10.3 and later.

**Related Sample Code**

SimpleReach

**Declared In**

SCNetworkReachability.h

## Data Types

**SCNetworkReachabilityRef**

The handle to a network address or name.

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

**Availability**

Available in Mac OS X v10.3 and later.

**Declared In**

SCNetworkReachability.h

## SCNetworkReachabilityContext

Structure containing user-specified data and callbacks used with [SCNetworkReachabilitySetCallback](#) (page 109).

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

### Fields

version

The version number of the structure type being passed in as a parameter to an `SCDynamicStore` creation function, such as [SCDynamicStoreCreate](#) (page 16). This structure is version 0.

info

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

retain

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

release

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

copyDescription

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

### Availability

Available in Mac OS X v10.3 and later.

### Declared In

`SCNetworkReachability.h`

## SCNetworkReachabilityCallback

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

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

### Fields

target

The network target being monitored for changes.

flags

The flags representing the reachability status of the network address or name (see [“Network Reachability Flags”](#) (page 112) for information about these flags).

info

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

## Constants

### Network Reachability Flags

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

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

#### Constants

`kSCNetworkReachabilityFlagsTransientConnection`

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsReachable`

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsConnectionRequired`

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsConnectionOnTraffic`

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

This flag was previously named `kSCNetworkReachabilityFlagsConnectionAutomatic`.

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsInterventionRequired`

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

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

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsConnectionOnDemand`

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsIsLocalAddress`

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.

`kSCNetworkReachabilityFlagsIsDirect`

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

Available in Mac OS X v10.6 and later.

Declared in `SCNetworkReachability.h`.



# SCPreferences Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCPreferences.h

## Overview

The `SCPreferences` programming interface allows an application to load and store XML configuration data in a controlled manner and provide the necessary notifications to other applications that need to be aware of configuration changes.

To access configuration preferences, you must first establish a preferences session using the `SCPreferencesCreate` (page 119) function. To identify a specific set of preferences to access, you pass a value in the `prefsID` parameter. A `NULL` value indicates that the default system preferences are to be accessed. A string that starts with a leading `"/` character specifies the absolute path to the file containing the preferences to be accessed. A string that does not start with a leading `"/` character specifies a file relative to the default system preferences directory.

When you are finished with the preferences session, use the `CFRelease` function to release it.

## Functions by Task

### Creating a Preferences Session

`SCPreferencesCreate` (page 119)

Initiates access to the per-system set of configuration preferences.

`SCPreferencesCreateWithAuthorization` (page 119)

Initiates access to the per-system set of configuration preferences with the specified authorization.

### Getting Information About a Preferences Session

`SCPreferencesGetTypeID` (page 121)

Returns the type identifier of all `SCPreferences` instances.

`SCPreferencesCopyKeyList` (page 118)

Returns the currently defined preference keys.

`SCPreferencesGetSignature` (page 120)

Returns a value that can be used to determine if the saved configuration preferences have changed.

## Adding, Getting, and Removing Values

[SCPreferencesAddValue](#) (page 117)

Associates the specified value with the specified preference key.

[SCPreferencesGetValue](#) (page 121)

Retrieves the value associated with the specified preference key.

[SCPreferencesSetValue](#) (page 124)

Updates the data associated with the specified preference key with the specified value.

[SCPreferencesRemoveValue](#) (page 122)

Removes the data associated with the specified preference key.

## Applying and Committing Changes

[SCPreferencesApplyChanges](#) (page 117)

Requests that the currently stored configuration preferences be applied to the active configuration.

[SCPreferencesCommitChanges](#) (page 118)

Commits changes made to the configuration preferences to persistent storage.

[SCPreferencesSynchronize](#) (page 125)

Synchronizes accessed preferences with committed changes.

## Managing Notifications and Callbacks

[SCPreferencesSetCallback](#) (page 123)

Assigns the specified callback to the specified preferences session.

[SCPreferencesScheduleWithRunLoop](#) (page 123)

Schedules commit and apply notifications for the specified preferences session using the specified run loop and mode.

[SCPreferencesUnscheduleFromRunLoop](#) (page 126)

Unschedules commit and apply notifications for the specified preferences session from the specified run loop and mode.

[SCPreferencesSetDispatchQueue](#) (page 124)

Schedules commit and apply notifications for the specified preferences session using the specified dispatch queue.

## Managing Access to a Preferences Session

[SCPreferencesLock](#) (page 121)

Locks access to the configuration preferences.

[SCPreferencesUnlock](#) (page 125)

Releases exclusive access to the configuration preferences.

## Functions

### SCPreferencesAddValue

Associates the specified value with the specified preference key.

```
Boolean SCPreferencesAddValue (
    SCPreferencesRef prefs,
    CFStringRef key,
    CFPropertyListRef value
);
```

#### Parameters

*prefs*

The preferences session.

*key*

The preference key.

*value*

The value to associate with the preference key.

#### Return Value

TRUE if the value was added; FALSE if the key already exists or if an error occurred.

#### Discussion

To commit these changes to permanent storage, you must call [SCPreferencesCommitChanges](#) (page 118).

#### Availability

Available in Mac OS X v10.1 and later.

#### Declared In

SCPreferences.h

### SCPreferencesApplyChanges

Requests that the currently stored configuration preferences be applied to the active configuration.

```
Boolean SCPreferencesApplyChanges (
    SCPreferencesRef prefs
);
```

#### Parameters

*prefs*

The preferences session.

#### Return Value

TRUE if the lock was obtained; FALSE if an error occurred.

#### Availability

Available in Mac OS X v10.1 and later.

#### Related Sample Code

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesCommitChanges**

Commits changes made to the configuration preferences to persistent storage.

```
Boolean SCPreferencesCommitChanges (
    SCPreferencesRef prefs
);
```

**Parameters***prefs*

The preferences session.

**Return Value**

TRUE if the lock was obtained; FALSE if an error occurred.

**Discussion**

Implicit calls to the [SCPreferencesLock](#) (page 121) and [SCPreferencesUnlock](#) (page 125) functions are made if exclusive access has not already been established.

**Note:** This function commits changes to persistent storage. To apply the changes to the running system, use the [SCPreferencesApplyChanges](#) (page 117) function.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesCopyKeyList**

Returns the currently defined preference keys.

```
CFArrayRef SCPreferencesCopyKeyList (
    SCPreferencesRef prefs
);
```

**Parameters***prefs*

The preferences session.

**Return Value**

An array of currently defined preference keys. You must release the returned value.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesCreate**

Initiates access to the per-system set of configuration preferences.

```
SCPreferencesRef SCPreferencesCreate (
    CFAllocatorRef allocator,
    CFStringRef name,
    CFStringRef prefsID
);
```

**Parameters***allocator*

The allocator to use to allocate memory for this preferences session. If the value is not a valid `CFAllocator`, the behavior is undefined. Pass `NULL` or `kCFAllocatorDefault` to use the current default `CFAllocator`.

*name*

The name of the calling process.

*prefsID*

The name of the group of preferences to be accessed or updated. A name that starts with a leading "/" character specifies the absolute path to the file containing the preferences to be accessed. A name that does not start with a leading "/" character specifies a file relative to the default system preferences directory.

To access the default system preferences, pass in `NULL`.

**Return Value**

A reference to the new preferences session. You must release the returned value.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesCreateWithAuthorization**

Initiates access to the per-system set of configuration preferences with the specified authorization.

```
SCPreferencesRef SCPreferencesCreateWithAuthorization (
    CFAllocatorRef allocator,
    CFStringRef name,
    CFStringRef prefsID,
    AuthorizationRef authorization
);
```

**Parameters***allocator*

The allocator to use to allocate memory for this preferences session. If the value is not a valid `CFAllocator`, the behavior is undefined. Pass `NULL` or `kCFAllocatorDefault` to use the current default `CFAllocator`.

*name*

The name of the calling process.

*prefsID*

The name of the group of preferences to be accessed or updated. A name that starts with a leading "/" character specifies the absolute path to the file containing the preferences to be accessed. A name that does not start with a leading "/" character specifies a file relative to the default system preferences directory.

To access the default system preferences, pass in `NULL`.

*authorization*

An authorization reference that is used to authorize any access to the enhanced privileges needed to manage the preferences session.

**Return Value**

A reference to the new preferences session. You must release the returned value.

**Availability**

Available in Mac OS X v10.5 and later.

**Declared In**

`SCPreferences.h`

**SCPreferencesGetSignature**

Returns a value that can be used to determine if the saved configuration preferences have changed.

```
CFDataRef SCPreferencesGetSignature (
    SCPreferencesRef prefs
);
```

**Parameters***prefs*

The preferences session.

**Return Value**

Data that reflects the signature of the configuration preferences at the time of the call to the [SCPreferencesCreate](#) (page 119) function.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCPreferences.h

**SCPreferencesGetTypeID**

Returns the type identifier of all SCPreferences instances.

```
CFTypeID SCPreferencesGetTypeID (
    void
);
```

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCPreferences.h

**SCPreferencesGetValue**

Retrieves the value associated with the specified preference key.

```
CFPropertyListRef SCPreferencesGetValue (
    SCPreferencesRef prefs,
    CFStringRef key
);
```

**Parameters***prefs*

The preferences session.

*key*

The preference key.

**Return Value**

The value associated with the specified preference key (can be NULL if no value exists).

**Discussion**

To avoid inadvertently reading stale data, first call [SCPreferencesLock](#) (page 121) before calling this function.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesLock**

Locks access to the configuration preferences.

```
Boolean SCPreferencesLock (
    SCPreferencesRef prefs,
    Boolean wait
);
```

**Parameters***prefs*

The preferences session.

*wait*

A Boolean value indicating whether the calling process should block, waiting for another process to complete its update operation and release its lock.

**Return Value**

TRUE if the lock was obtained; FALSE if an error occurred.

**Discussion**

This function obtains exclusive access to the configuration preferences. Clients attempting to obtain exclusive access to the preferences either receive a `kSCStatusPrefsBusy` error or they block, waiting for the lock to be released.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesRemoveValue**

Removes the data associated with the specified preference key.

```
Boolean SCPreferencesRemoveValue (
    SCPreferencesRef prefs,
    CFStringRef key
);
```

**Parameters***prefs*

The preferences session.

*key*

The preference key.

**Return Value**

TRUE if the value was removed; FALSE if the key does not exist or if an error occurred.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCPreferences.h

**SCPreferencesScheduleWithRunLoop**

Schedules commit and apply notifications for the specified preferences session using the specified run loop and mode.

```
Boolean SCPreferencesScheduleWithRunLoop (
    SCPreferencesRef prefs,
    CFRunLoopRef runLoop,
    CFStringRef runLoopMode
);
```

**Parameters**

*prefs*

The preferences session.

*runLoop*

The run loop on which the notification should be scheduled. Do not pass NULL.

*runLoopMode*

The run loop mode with which to schedule the notification. Do not pass NULL.

**Return Value**

TRUE if the notifications are successfully scheduled; otherwise, FALSE.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCPreferences.h

**SCPreferencesSetCallback**

Assigns the specified callback to the specified preferences session.

```
Boolean SCPreferencesSetCallback (
    SCPreferencesRef prefs,
    SCPreferencesCallback callout,
    SCPreferencesContext *context
);
```

**Parameters**

*prefs*

The preferences session.

*callout*

The function to be called when the preferences have been changed or applied. If NULL, the current callback is removed.

*Term*

The context associated with the callback function. See [SCPreferencesContext](#) (page 127) for more information about this structure.

**Return Value**

TRUE if the callback was successfully associated with the preferences session; otherwise, FALSE.

**Discussion**

This function is called when the changes to the preferences have been committed or applied.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCPreferences.h

**SCPreferencesSetDispatchQueue**

Schedules commit and apply notifications for the specified preferences session using the specified dispatch queue.

```
Boolean SCPreferencesSetDispatchQueue (
    SCPreferencesRef prefs,
    dispatch_queue_t queue
);
```

**Parameters**

*prefs*

The preferences session.

*queue*

The dispatch queue on which to run the callback function.

**Return Value**

TRUE if the notifications are successfully scheduled; otherwise, FALSE.

**Availability**

Available in Mac OS X v10.6 and later.

**Declared In**

SCPreferences.h

**SCPreferencesSetValue**

Updates the data associated with the specified preference key with the specified value.

```
Boolean SCPreferencesSetValue (
    SCPreferencesRef prefs,
    CFStringRef key,
    CFPropertyListRef value
);
```

**Parameters**

*prefs*

The preferences session.

*key*

The preference key.

*value*

The value to associate with the preference key.

**Return Value**

TRUE if the value was set; FALSE if an error occurred.

**Discussion**

This function adds or replaces the value associated with the specified key. To commit these changes to permanent storage you must call [SCPreferencesCommitChanges](#) (page 118).

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesSynchronize**

Synchronizes accessed preferences with committed changes.

```
void SCPreferencesSynchronize (
    SCPreferencesRef prefs
);
```

**Parameters**

*prefs*

The preferences session.

**Discussion**

Any references to preference values returned by calls to [SCPreferencesGetValue](#) (page 121) are no longer valid unless they were explicitly retained or copied. Any preference values that were updated (added, set, or removed), but not committed, are discarded.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCPreferences.h

**SCPreferencesUnlock**

Releases exclusive access to the configuration preferences.

```
Boolean SCPreferencesUnlock (
    SCPreferencesRef prefs
);
```

**Parameters**

*prefs*

The preferences session.

**Return Value**

TRUE if the lock was obtained; FALSE if an error occurred.

**Discussion**

After exclusive access has been released, other clients can establish exclusive access to the preferences.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

**Declared In**

SCPreferences.h

**SCPreferencesUnscheduleFromRunLoop**

Unschedulates commit and apply notifications for the specified preferences session from the specified run loop and mode.

```
Boolean SCPreferencesUnscheduleFromRunLoop (
    SCPreferencesRef prefs,
    CFRunLoopRef runLoop,
    CFStringRef runLoopMode
);
```

**Parameters**

*prefs*

The preferences session.

*runLoop*

The run loop from which the notification should be unscheduled. Do not pass NULL.

*runLoopMode*

The run loop mode associated with the scheduled notification. Do not pass NULL.

**Return Value**

TRUE if the notifications are successfully unscheduled; otherwise, FALSE.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

SCPreferences.h

## Data Types

**SCPreferencesRef**

The handle to an open preferences session for accessing system configuration preferences.

```
typedef const struct __SCPreferences * SCPreferencesRef;
```

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCPreferences.h

## SCPreferencesContext

A structure containing user-specified data and callbacks for accessing system configuration preferences.

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

### Fields

version

The version number of the structure type being passed in as a parameter to [SCPreferencesSetCallback](#) (page 123). This structure is version 0.

info

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

retain

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

release

The callback used to remove a retain previously added for the *info* field. If this parameter is not a pointer to a function of the correct prototype, the behavior is undefined. The value may be NULL.

copyDescription

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

### Availability

Available in Mac OS X v10.4 and later.

### Declared In

SCPreferences.h

## SCPreferencesCallback

Type of the callback function used when the preferences have been updated or applied.

```
typedef void (*SCPreferencesCallback) (
    SCPreferencesRef prefs,
    SCPreferencesNotification notificationType,
    void *info
);
```

### Fields

prefs

The preferences session.

notificationType

The type of notification, such as changes committed or changes applied. See [“Preferences Notification Values”](#) (page 128) for information about possible values.

info

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

## Constants

### Preferences Notification Values

The type of notification (used with the `SCPreferencesCallback` (page 127) callback).

```
enum {  
    kSCPreferencesNotificationCommit = 1<<0,  
    kSCPreferencesNotificationApply = 1<<1  
};  
typedef    uint32_t    SCPreferencesNotification;
```

#### Constants

`kSCPreferencesNotificationCommit`

Indicates when new preferences have been saved.

Available in Mac OS X v10.4 and later.

Declared in `SCPreferences.h`.

`kSCPreferencesNotificationApply`

Indicates when a request has been made to apply the currently saved preferences to the active system configuration.

Available in Mac OS X v10.4 and later.

Declared in `SCPreferences.h`.

# SCPreferencesPath Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCPreferences.h

## Overview

The `SCPreferencesPath` programming interface allows an application to load and store XML configuration data in a controlled manner and provide the necessary notifications to other applications that need to be aware of configuration changes.

The functions in this programming interface view the data as a collection of dictionaries of key-value pairs and an associated path name. The root path ("/") identifies the top-level dictionary. Additional path components specify the keys for subdictionaries.

For example, the following dictionary can be accessed via two paths. The root path ("/") returns a dictionary with all keys and values. The path "/path1" returns only the dictionary with the "key3" and "key4" properties.

```
<dict>
  <key>key1</key>
  <string>val1</string>
  <key>key2</key>
  <string>val2</string>
  <key>path1</key>
  <dict>
    <key>key3</key>
    <string>val3</string>
    <key>key4</key>
    <string>val4</string>
  </dict>
</dict>
```

Each dictionary can also include the `kSCResvLink` ("\_\_LINK\_\_") key. The value associated with this key is interpreted as a link to another path. If this key is present, a call to the `SCPreferencesPathGetValue` (page 131) function returns the dictionary specified by the link.

## Functions by Task

### Creating a New Path

`SCPreferencesPathCreateUniqueChild` (page 130) **Deprecated in Mac OS X v10.6**

Creates a new path component rooted at the specified path in the dictionary hierarchy.

## Associating Information with a Path

[SCPreferencesPathSetLink](#) (page 132)

Associates a link to a second dictionary at the specified path.

[SCPreferencesPathSetValue](#) (page 133) **Deprecated in Mac OS X v10.5**

Associates the specified dictionary with the specified path.

## Getting or Removing Information Associated with a Path

[SCPreferencesPathGetValue](#) (page 131)

Returns the dictionary associated with the specified path.

[SCPreferencesPathGetLink](#) (page 131)

Returns the link associated with the specified path.

[SCPreferencesPathRemoveValue](#) (page 132)

Removes the data associated with the specified path.

## Functions

### SCPreferencesPathCreateUniqueChild

Creates a new path component rooted at the specified path in the dictionary hierarchy.

```
CFStringRef SCPreferencesPathCreateUniqueChild (
    SCPreferencesRef prefs,
    CFStringRef prefix
);
```

#### Parameters

*prefs*

The preferences session.

*prefix*

The parent path.

#### Return Value

A string representing the new (unique) child path, or `NULL` if the specified path does not exist.

#### Availability

Available in Mac OS X v10.1 and later.

#### Related Sample Code

MoreSCF

#### Declared In

SCPreferencesPath.h

## SCPreferencesPathGetLink

Returns the link associated with the specified path.

```
CFStringRef SCPreferencesPathGetLink (  
    SCPreferencesRef prefs,  
    CFStringRef path  
);
```

### Parameters

*prefs*

The preferences session.

*path*

The path.

### Return Value

The link associated with the specified path, or NULL if the path is not a link or does not exist.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

MoreSCF

### Declared In

SCPreferencesPath.h

## SCPreferencesPathGetValue

Returns the dictionary associated with the specified path.

```
CFDictionaryRef SCPreferencesPathGetValue (  
    SCPreferencesRef prefs,  
    CFStringRef path  
);
```

### Parameters

*prefs*

The preferences session.

*path*

The path.

### Return Value

The dictionary associated with the specified path, or NULL if the path does not exist.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

MoreSCF

### Declared In

SCPreferencesPath.h

## SCPreferencesPathRemoveValue

Removes the data associated with the specified path.

```
Boolean SCPreferencesPathRemoveValue (  
    SCPreferencesRef prefs,  
    CFStringRef path  
);
```

### Parameters

*prefs*

The preferences session.

*path*

The path.

### Return Value

TRUE if successful; otherwise, FALSE.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

MoreSCF

### Declared In

SCPreferencesPath.h

## SCPreferencesPathSetLink

Associates a link to a second dictionary at the specified path.

```
Boolean SCPreferencesPathSetLink (  
    SCPreferencesRef prefs,  
    CFStringRef path,  
    CFStringRef link  
);
```

### Parameters

*prefs*

The preferences session.

*path*

The path.

*link*

The link to be stored at the path.

### Return Value

TRUE if successful; otherwise, FALSE.

### Availability

Available in Mac OS X v10.1 and later.

### Declared In

SCPreferencesPath.h

## SCPreferencesPathSetValue

Associates the specified dictionary with the specified path.

```
Boolean SCPreferencesPathSetValue (  
    SCPreferencesRef prefs,  
    CFStringRef path,  
    CFDictionaryRef value  
);
```

### Parameters

*prefs*

The preferences session.

*path*

The path.

*value*

The dictionary of data to be stored at the path.

### Return Value

TRUE if successful; otherwise, FALSE.

### Availability

Available in Mac OS X v10.1 and later.

### Related Sample Code

MoreSCF

### Declared In

SCPreferencesPath.h



# SCPreferencesSetSpecific Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SCPreferencesSetSpecific.h

## Overview

The functions in the `SCPreferencesSetSpecific` programming interface allow an application to set specific configuration information about the current system (for example, the computer or sharing name). Note that to access configuration preferences, you must first establish a preferences session using the [SCPreferencesCreate](#) (page 119) function.

## Functions

### SCPreferencesSetComputerName

Sets the computer name preference to the specified name.

```
Boolean SCPreferencesSetComputerName (
    SCPreferencesRef prefs,
    CFStringRef name,
    CFStringEncoding nameEncoding
);
```

#### Parameters

*prefs*

The preferences session.

*name*

The computer name.

*nameEncoding*

The encoding associated with the computer name.

#### Return Value

TRUE if successful; otherwise, FALSE.

#### Discussion

To commit these changes to permanent storage you must call the [SCPreferencesCommitChanges](#) (page 118) function. In addition, you must call the [SCPreferencesApplyChanges](#) (page 117) function for the new name to become active.

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SCPreferencesSetSpecific.h

**SCPreferencesSetLocalHostName**

Sets the local host name to the specified name.

```
Boolean SCPreferencesSetLocalHostName (  
    SCPreferencesRef prefs,  
    CFStringRef name  
);
```

**Parameters**

*prefs*

The preferences session.

*name*

The local host name. This string must conform to the naming conventions of a DNS host name as specified in RFC 1034 (section 3.5).

**Return Value**

TRUE if successful; otherwise, FALSE.

**Discussion**

To commit these changes to permanent storage you must call the [SCPreferencesCommitChanges](#) (page 118) function. In addition, you must call the [SCPreferencesApplyChanges](#) (page 117) function for the new name to become active.

**Availability**

Available in Mac OS X v10.2 and later.

**Declared In**

SCPreferencesSetSpecific.h

# SCSchemaDefinitions Reference

---

Declared in

## Overview

This document describes keys and values used to access elements in the System Configuration persistent store.

## Constants

### Generic Keys

General-purpose keys that apply to multiple dictionaries in the persistent store.

kSCPropInterfaceName  
 kSCPropMACAddress  
 kSCPropUserDefinedName  
 kSCPropVersion

### Constants

kSCPropInterfaceName  
 The generic key `InterfaceName`, whose value is of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

kSCPropMACAddress  
 The generic key `MACAddress`, whose value is of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

kSCPropUserDefinedName  
 The generic key `UserDefinedName`, whose value is of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

kSCPropVersion  
 The generic key `Version`, whose value is of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

## Preference Keys

Keys that correspond to preferences in the persistent store.

kSCPrefCurrentSet  
kSCPrefNetworkServices  
kSCPrefSets  
kSCPrefSystem

### Constants

kSCPrefCurrentSet  
The preference key `CurrentSet`, whose value is of type `CFString`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

kSCPrefNetworkServices  
The preference key for the `NetworkServices` dictionary.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

kSCPrefSets  
The preference key for the `Sets` dictionary.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

kSCPrefSystem  
The preference key for the `System` dictionary.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

## Component Keys

Keys that correspond to components in the persistent store.

kSCCompNetwork  
kSCCompService  
kSCCompGlobal  
kSCCompHostNames  
kSCCompInterface  
kSCCompSystem  
kSCCompUsers

### Constants

kSCCompNetwork  
The Component key `Network`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

kSCCompService  
The Component key `Service`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

kSCCompGlobal

The Component key Global.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCCompHostNames

The Component key HostNames.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCCompInterface

The Component key Interface.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCCompSystem

The Component key System.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCCompUsers

The Network key Users.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## Network Dictionary Keys

Keys that correspond to values in the kSCCompNetwork dictionary.

kSCPropNetOverridePrimary

kSCPropNetServiceOrder

kSCPropNetPPPOverridePrimary

### Constants

kSCPropNetOverridePrimary

The Network key OverridePrimary, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetServiceOrder

The Network key ServiceOrder, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPOverridePrimary

The Network key PPPOverridePrimary, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## Interface Dictionary Keys

Keys that correspond to values in the `kSCCompInterface` dictionary.

`kSCPropNetInterfaces`

### Constants

`kSCPropNetInterfaces`

The Network key `Interfaces`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

## Hostnames Dictionary Keys

Keys that correspond to values in the `kSCCompHostnames` dictionary.

`kSCPropNetLocalHostName`

### Constants

`kSCPropNetLocalHostName`

The Network key `LocalHostName`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

## Network Entity Keys

Keys that correspond to network entity dictionaries in the persistent store.

kSCEntNetAirPort  
kSCEntNetAppleTalk  
kSCEntNetDHCP  
kSCEntNetDNS  
kSCEntNetEthernet  
kSCEntNetFireWire  
kSCEntNetInterface  
kSCEntNetIPSec  
kSCEntNetIPv4  
kSCEntNetIPv6  
kSCEntNetL2TP  
kSCEntNetLink  
kSCEntNetModem  
kSCEntNetPPP  
kSCEntNetPPPoE  
kSCEntNetPPPSerial  
kSCEntNetPPTP  
kSCEntNetProxies  
kSCEntNetSMB  
kSCEntNet6to4

### Constants

kSCEntNetAirPort

**The network entity key for the AirPort dictionary.**

**Available in Mac OS X v10.3 and later.**

**Declared in SCSchemaDefinitions.h.**

kSCEntNetAppleTalk

**The network entity key for the AppleTalk dictionary.**

**Available in Mac OS X v10.3 and later.**

**Deprecated in Mac OS X v10.6.**

**Declared in SCSchemaDefinitions.h.**

kSCEntNetDHCP

**The network entity key for the DHCP dictionary.**

**Available in Mac OS X v10.3 and later.**

**Declared in SCSchemaDefinitions.h.**

kSCEntNetDNS

**The network entity key for the DNS dictionary.**

**Available in Mac OS X v10.3 and later.**

**Declared in SCSchemaDefinitions.h.**

kSCEntNetEthernet

**The network entity key for the Ethernet dictionary.**

**Available in Mac OS X v10.3 and later.**

**Declared in SCSchemaDefinitions.h.**

kSCEntNetFireWire

**The network entity key for the FireWire dictionary.**

**Available in Mac OS X v10.3 and later.**

**Declared in SCSchemaDefinitions.h.**

`kSCEntNetInterface`

The network entity key for the `Interface` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetIPSec`

The network entity key for the `IPSec` dictionary.

Available in Mac OS X v10.5 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetIPv4`

The network entity key for the `IPv4` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetIPv6`

The network entity key for the `IPv6` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetL2TP`

The network entity key for the `L2TP` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetLink`

The network entity key for the `Link` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetModem`

The network entity key for the `Modem` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetPPP`

The network entity key for the `PPP` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetPPPoE`

The network entity key for the `PPPoE` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetPPPSerial`

The network entity key for the `PPPSerial` dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetPPTP`

The network entity key for the PPTP dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetProxies`

The network entity key for the Proxies dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNetSMB`

The network entity key for the SMB dictionary.

Available in Mac OS X v10.5 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCEntNet6to4`

The network entity key for the 6to4 dictionary.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

## AirPort Dictionary Keys

Keys that correspond to values in the `kSCEntNetAirPort` dictionary.

`kSCPropNetAirPortAllowNetCreation`

`kSCPropNetAirPortAuthPassword`

`kSCPropNetAirPortAuthPasswordEncryption`

### Constants

`kSCPropNetAirPortAllowNetCreation`

The AirPort key `AllowNetCreation`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetAirPortAuthPassword`

The AirPort key `AuthPassword`, whose value is of type `CFData`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetAirPortAuthPasswordEncryption`

The AirPort key `AuthPasswordEncryption`, whose value is of type `CFString`.

This key can be passed the following constants:

- `kSCValNetAirPortAuthPasswordEncryptionKeychain`, which has the value `Keychain`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAirPortJoinMode

The **AirPort** key `JoinMode`, whose value is of type `CFString`.

This key can be passed the following constants:

- `kSCValNetAirPortJoinModeAutomatic`, which has the value `Automatic`
- `kSCValNetAirPortJoinModePreferred`, which has the value `Preferred`
- `kSCValNetAirPortJoinModeRanked`, which has the value `Ranked`
- `kSCValNetAirPortJoinModeRecent`, which has the value `Recent`
- `kSCValNetAirPortJoinModeStrongest`, which has the value `Strongest`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAirPortPowerEnabled

The **AirPort** key `PowerEnabled`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAirPortPreferredNetwork

The **AirPort** key `PreferredNetwork`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAirPortSavePasswords

The **AirPort** key `SavePasswords`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

## AppleTalk Entity Keys

Keys that correspond to values in the `kSCEntNetAppleTalk` dictionary.

kSCPropNetAppleTalkComputerName  
 kSCPropNetAppleTalkComputerNameEncoding  
 kSCPropNetAppleTalkConfigMethod  
 kSCPropNetAppleTalkDefaultZone  
 kSCPropNetAppleTalkNetworkID  
 kSCPropNetAppleTalkNetworkRange  
 kSCPropNetAppleTalkNodeID  
 kSCPropNetAppleTalkSeedNetworkRange  
 kSCPropNetAppleTalkSeedZones

### Constants

kSCPropNetAppleTalkComputerName

The **AppleTalk** key `ComputerName`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkComputerNameEncoding

The AppleTalk key `ComputerNameEncoding`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkConfigMethod

The AppleTalk key `ConfigMethod`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkDefaultZone

The AppleTalk key `DefaultZone`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkNetworkID

The AppleTalk key `NetworkID`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkNetworkRange

The AppleTalk key `NetworkRange`, whose value is of type `CFArray`, containing elements of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkNodeID

The AppleTalk key `NodeID`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkSeedNetworkRange

The AppleTalk key `SeedNetworkRange`, whose value is of type `CFArray`, containing elements of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetAppleTalkSeedZones

The AppleTalk key `SeedZones`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Deprecated in Mac OS X v10.6.

Declared in `SCSchemaDefinitions.h`.

## DNS Entity Keys

Keys that correspond to values in the `kSCEntNetDNS` dictionary.

`kSCPropNetDNSDomainName`  
`kSCPropNetDNSOptions`  
`kSCPropNetDNSSearchDomains`  
`kSCPropNetDNSSearchOrder`  
`kSCPropNetDNSServerAddresses`  
`kSCPropNetDNSServerPort`  
`kSCPropNetDNSServerTimeout`  
`kSCPropNetDNSSortList`  
`kSCPropNetDNSSupplementalMatchDomains`  
`kSCPropNetDNSSupplementalMatchOrders`

### Constants

`kSCPropNetDNSDomainName`  
 The DNS key `DomainName`, whose value is of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropNetDNSOptions`  
 The DNS key `Options`, whose value is of type `CFString`.  
 Available in Mac OS X v10.4 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropNetDNSSearchDomains`  
 The DNS key `SearchDomains`, whose value is of type `CFArray`, containing elements of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropNetDNSSearchOrder`  
 The DNS key `SearchOrder`, whose value is of type `CFNumber`.  
 Available in Mac OS X v10.4 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropNetDNSServerAddresses`  
 The DNS key `ServerAddresses`, whose value is of type `CFArray`, containing elements of type `CFString`.  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropNetDNSServerPort`  
 The DNS key `ServerPort`, whose value is of type `CFNumber`.  
 Available in Mac OS X v10.4 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropNetDNSServerTimeout`  
 The DNS key `ServerTimeout`, whose value is of type `CFNumber`.  
 Available in Mac OS X v10.4 and later.  
 Declared in `SCSchemaDefinitions.h`.

kSCPropNetDNSSortList

The DNS key `SortList`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetDNSSupplementalMatchDomains

The DNS key `SupplementalMatchDomains`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.4 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetDNSSupplementalMatchOrders

The DNS key `SupplementalMatchOrders`, whose value is of type `CFArray`, containing elements of type `CFNumber`.

Available in Mac OS X v10.4 and later.

Declared in `SCSchemaDefinitions.h`.

## Ethernet Entity Keys

Keys that correspond to values in the `kSCEntNetEthernet` dictionary.

kSCPropNetEthernetMediaSubType

kSCPropNetEthernetMediaOptions

kSCPropNetEthernetMTU

### Constants

kSCPropNetEthernetMediaSubType

The Ethernet key `MediaSubType`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetEthernetMediaOptions

The Ethernet key `MediaOptions`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetEthernetMTU

The Ethernet key `MTU`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

## Interface Entity Keys

Keys that correspond to values in the `kSCEntNetInterface` dictionary.

kSCPropNetInterfaceDeviceName  
 kSCPropNetInterfaceHardware  
 kSCPropNetInterfaceType  
 kSCPropNetInterfaceSubType  
 kSCPropNetInterfaceSupportsModemOnHold

**Constants**

kSCPropNetInterfaceDeviceName

The **Interface key** DeviceName, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetInterfaceHardware

The **Interface key** Hardware, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetInterfaceType

The **Interface key** Type, whose value is of type CFString.

This key can be passed the following constants:

- kSCValNetInterfaceTypeEthernet, which has the value Ethernet
- kSCValNetInterfaceTypeFireWire, which has the value FireWire
- kSCValNetInterfaceTypePPP, which has the value PPP
- kSCValNetInterfaceType6to4, which has the value 6to4

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetInterfaceSubType

The **Interface key** SubType, whose value is of type CFString.

This key can be passed the following constants when the Type key has the value PPP:

- kSCValNetInterfaceSubTypePPPoE, which has the value PPPoE
- kSCValNetInterfaceSubTypePPPSerial, which has the value PPPSerial
- kSCValNetInterfaceSubTypePPTP, which has the value PPTP
- kSCValNetInterfaceSubTypeL2TP, which has the value L2TP

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetInterfaceSupportsModemOnHold

The **Interface key** SupportsModemOnHold, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

**IPSec Entity Keys**

Keys that correspond to values in the kSCEntNetIPSec dictionary.

kSCPropNetIPSecLocalIdentifier  
 kSCPropNetIPSecLocalIdentifierType  
 kSCPropNetIPSecAuthenticationMethod  
 kSCPropNetIPSecSharedSecret  
 kSCPropNetIPSecSharedSecretEncryption  
 kSCPropNetIPSecLocalCertificate

### Constants

kSCPropNetIPSecLocalIdentifier

The IPsec key LocalIdentifier, whose value is of type CFString.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPSecLocalIdentifierType

The IPsec key LocalIdentifierType, whose value is of type CFString.

This key can be passed the following constant:

- kSCValNetIPSecLocalIdentifierTypeKeyID, which has the value KeyID

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPSecAuthenticationMethod

The IPsec key AuthenticationMethod, whose value is of type CFString.

This key can be passed the following constants:

- kSCValNetIPSecAuthenticationMethodSharedSecret, which has the value SharedSecret
- kSCValNetIPSecAuthenticationMethodCertificate, which has the value Certificate

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPSecSharedSecret

The IPsec key SharedSecret, whose value is of type CFString.

This key can be passed the following constant:

- kSCValNetIPSecSharedSecretEncryptionKeychain, which has the value KeyChain

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPSecSharedSecretEncryption

The IPsec key SharedSecretEncryption, whose value is of type CFString.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPSecLocalCertificate

The IPsec key LocalCertificate, whose value is of type CFData.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

## IPv4 Entity Keys

Keys that correspond to values in the kSCEntNetIPv4 dictionary.

kSCPropNetIPv4Addresses  
 kSCPropNetIPv4ConfigMethod  
 kSCPropNetIPv4DHCPClientID  
 kSCPropNetIPv4Router  
 kSCPropNetIPv4SubnetMasks  
 kSCPropNetIPv4DestAddresses  
 kSCPropNetIPv4BroadcastAddresses

### Constants

kSCPropNetIPv4Addresses

The IPv4 key `Addresses`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetIPv4ConfigMethod

The IPv4 key `ConfigMethod`, whose value is of type `CFString`.

This key can be passed the following constants:

- `kSCValNetIPv4ConfigMethodBOOTP`, which has the value `BOOTP`
- `kSCValNetIPv4ConfigMethodDHCP`, which has the value `DHCP`
- `kSCValNetIPv4ConfigMethodINFORM`, which has the value `INFORM`
- `kSCValNetIPv4ConfigMethodLinkLocal`, which has the value `LinkLocal`
- `kSCValNetIPv4ConfigMethodManual`, which has the value `Manual`
- `kSCValNetIPv4ConfigMethodPPP`, which has the value `PPP`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetIPv4DHCPClientID

The IPv4 key `DHCPClientID`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetIPv4Router

The IPv4 key `Router`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetIPv4SubnetMasks

The IPv4 key `SubnetMasks`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetIPv4DestAddresses

The IPv4 key `DestAddresses`, whose value is of type `CFArray`, containing elements of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetIPv4BroadcastAddresses

The **IPv4 key** BroadcastAddresses, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## IPv6 Entity Keys

Keys that correspond to values in the kSCEntNetIPv6 dictionary.

kSCPropNetIPv6Addresses

kSCPropNetIPv6ConfigMethod

kSCPropNetIPv6DestAddresses

kSCPropNetIPv6Flags

kSCPropNetIPv6PrefixLength

kSCPropNetIPv6Router

### Constants

kSCPropNetIPv6Addresses

The **IPv6 key** Addresses, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPv6ConfigMethod

The **IPv6 key** ConfigMethod, whose value is of type CFString.

This key can be passed the following constants:

- kSCValNetIPv6ConfigMethodAutomatic, which has the value Automatic
- kSCValNetIPv6ConfigMethodManual, which has the value Manual
- kSCValNetIPv6ConfigMethodRouterAdvertisement, which has the value RouterAdvertisement
- kSCValNetIPv6ConfigMethod6to4, which has the value 6to4

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPv6DestAddresses

The **IPv6 key** DestAddresses, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPv6Flags

The **IPv6 key** Flags, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPv6PrefixLength

The **IPv6 key** PrefixLength, whose value is of type CFArray, containing elements of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetIPv6Router

The IPv6 key Router, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## 6to4 Entity Keys

Keys that correspond to values in the kSCEntNet6to4 dictionary.

kSCPropNet6to4Relay

### Constants

kSCPropNet6to4Relay

The 6to4 key Relay, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## Link Entity Keys

Keys that correspond to values in the kSCEntNetLink dictionary.

kSCPropNetLinkActive

kSCPropNetLinkDetaching

### Constants

kSCPropNetLinkActive

The Link key Active, whose value is of type CFBoolean.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetLinkDetaching

The Link key Detaching, whose value is of type CFBoolean.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## Modem Entity Keys

Keys that correspond to values in the kSCEntNetModem dictionary.

kSCPropNetModemAccessPointName  
 kSCPropNetModemConnectionPersonality  
 kSCPropNetModemConnectionScript  
 kSCPropNetModemConnectSpeed  
 kSCPropNetModemDataCompression  
 kSCPropNetModemDeviceContextID  
 kSCPropNetModemDeviceModel  
 kSCPropNetModemDeviceVendor  
 kSCPropNetModemDialMode  
 kSCPropNetModemErrorCorrection  
 kSCPropNetModemHoldCallWaitingAudibleAlert  
 kSCPropNetModemHoldDisconnectOnAnswer  
 kSCPropNetModemHoldEnabled  
 kSCPropNetModemHoldReminder  
 kSCPropNetModemHoldReminderTime  
 kSCPropNetModemNote  
 kSCPropNetModemPulseDial  
 kSCPropNetModemSpeaker  
 kSCPropNetModemSpeed

**Constants**

kSCPropNetModemAccessPointName  
**The Modem key** AccessPointName, whose value is of type CFString.  
**Available in Mac OS X v10.5 and later.**  
**Declared in** SCSchemaDefinitions.h.

kSCPropNetModemConnectionPersonality  
**The Modem key** ConnectionPersonality, whose value is of type CFString.  
**Available in Mac OS X v10.5 and later.**  
**Declared in** SCSchemaDefinitions.h.

kSCPropNetModemConnectionScript  
**The Modem key** ConnectionScript, whose value is of type CFString.  
**Available in Mac OS X v10.3 and later.**  
**Declared in** SCSchemaDefinitions.h.

kSCPropNetModemConnectSpeed  
**The Modem key** ConnectSpeed, whose value is of type CFNumber.  
**Available in Mac OS X v10.3 and later.**  
**Declared in** SCSchemaDefinitions.h.

kSCPropNetModemDataCompression  
**The Modem key** DataCompression, whose value is of type CFNumber and is equal to 0 or 1.  
**Available in Mac OS X v10.3 and later.**  
**Declared in** SCSchemaDefinitions.h.

kSCPropNetModemDeviceContextID  
**The Modem key** DeviceContextID, whose value is of type CFString.  
**Available in Mac OS X v10.5 and later.**  
**Declared in** SCSchemaDefinitions.h.

`kSCPropNetModemDeviceModel`

The Modem key `DeviceModel`, whose value is of type `CFString`.

Available in Mac OS X v10.5 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemDeviceVendor`

The Modem key `DeviceVendor`, whose value is of type `CFString`.

Available in Mac OS X v10.5 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemDialMode`

The Modem key `DialMode`, whose value is of type `CFString`.

This key can be passed the following constants:

- `kSCValNetModemDialModeIgnoreDialTone`, which has the value `IgnoreDialTone`
- `kSCValNetModemDialModeManual`, which has the value `Manual`
- `kSCValNetModemDialModeWaitForDialTone`, which has the value `WaitForDialTone`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemErrorCorrection`

The Modem key `ErrorCorrection`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemHoldCallWaitingAudibleAlert`

The Modem key `HoldCallWaitingAudibleAlert`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemHoldDisconnectOnAnswer`

The Modem key `HoldDisconnectOnAnswer`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemHoldEnabled`

The Modem key `HoldEnabled`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemHoldReminder`

The Modem key `HoldReminder`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetModemHoldReminderTime`

The Modem key `HoldReminderTime`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetModemNote

The Modem key Note, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetModemPulseDial

The Modem key PulseDial, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetModemSpeaker

The Modem key Speaker, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetModemSpeed

The Modem key Speed, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## PPP Entity Keys

Keys that correspond to values in the kSCEntNetPPP dictionary.

kSCPropNetPPPACSPEnabled  
 kSCPropNetPPPConnectTime  
 kSCPropNetPPPDeviceLastCause  
 kSCPropNetPPPDialOnDemand  
 kSCPropNetPPPDisconnectOnFastUserSwitch  
 kSCPropNetPPPDisconnectOnIdle  
 kSCPropNetPPPDisconnectOnIdleTimer  
 kSCPropNetPPPDisconnectOnLogout  
 kSCPropNetPPPDisconnectOnSleep  
 kSCPropNetPPPDisconnectTime  
 kSCPropNetPPPIdleReminderTimer  
 kSCPropNetPPPIdleReminder  
 kSCPropNetPPPLastCause  
 kSCPropNetPPPLogFile  
 kSCPropNetPPPPlugins  
 kSCPropNetPPPRetryConnectTime  
 kSCPropNetPPPSessionTimer  
 kSCPropNetPPPStatus  
 kSCPropNetPPPUseSessionTimer  
 kSCPropNetPPPVerboseLogging  
 kSCPropNetPPPAuthEAPPlugins  
 kSCPropNetPPPAuthName  
 kSCPropNetPPPAuthPassword  
 kSCPropNetPPPAuthPasswordEncryption  
 kSCPropNetPPPAuthPrompt  
 kSCPropNetPPPAuthProtocol  
 kSCPropNetPPPCommAlternateRemoteAddress  
 kSCPropNetPPPCommConnectDelay  
 kSCPropNetPPPCommDisplayTerminalWindow  
 kSCPropNetPPPCommRedialCount  
 kSCPropNetPPPCommRedialEnabled  
 kSCPropNetPPPCommRedialInterval  
 kSCPropNetPPPCommRemoteAddress  
 kSCPropNetPPPCommTerminalScript  
 kSCPropNetPPPCommUseTerminalScript  
 kSCPropNetPPPCCPEnabled  
 kSCPropNetPPPCCPMPPE40Enabled  
 kSCPropNetPPPCCPMPPE128Enabled  
 kSCPropNetPPPIPCPCompressionVJ  
 kSCPropNetPPPIPCPUsePeerDNS  
 kSCPropNetPPPLCPEchoEnabled  
 kSCPropNetPPPLCPEchoFailure  
 kSCPropNetPPPLCPEchoInterval  
 kSCPropNetPPPLCPCCompressionACField  
 kSCPropNetPPPLCPCCompressionPField  
 kSCPropNetPPPLCPMRU  
 kSCPropNetPPPLCPMTU  
 kSCPropNetPPPLCPReceiveACCM  
 kSCPropNetPPPLCPTransmitACCM

### Constants

kSCPropNetPPPACSPEnabled

The PPP key ACSPEnabled, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPConnectTime

The PPP key `ConnectTime`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDeviceLastCause

The PPP key `DeviceLastCause`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDialOnDemand

The PPP key `DialOnDemand`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDisconnectOnFastUserSwitch

The PPP key `DisconnectOnFastUserSwitch`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.4 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDisconnectOnIdle

The PPP key `DisconnectOnIdle`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDisconnectOnIdleTimer

The PPP key `DisconnectOnIdleTimer`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDisconnectOnLogout

The PPP key `DisconnectOnLogout`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDisconnectOnSleep

The PPP key `DisconnectOnSleep`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPDisconnectTime

The PPP key `DisconnectTime`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPIdleReminderTimer

The PPP key `IdleReminderTimer`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPIIdleReminder

The PPP key IdleReminder, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLastCause

The PPP key LastCause, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLogfile

The PPP key Logfile, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPPlugins

The PPP key Plugins, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPRetryConnectTime

The PPP key RetryConnectTime, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPSessionTimer

The PPP key SessionTimer, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPStatus

The PPP key Status, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPUseSessionTimer

The PPP key UseSessionTimer, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPVerboseLogging

The PPP key VerboseLogging, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPAuthEAPPlugins

The PPP key AuthEAPPlugins, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

`kSCPropNetPPPAuthName`

The PPP key `AuthName`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetPPPAuthPassword`

The PPP key `AuthPassword`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetPPPAuthPasswordEncryption`

The PPP key `AuthPasswordEncryption`, whose value is of type `CFString`.

This key can be passed the following constants:

- `kSCValNetPPPAuthPasswordEncryptionKeychain`, which has the value `Keychain`
- `kSCValNetPPPAuthPasswordEncryptionToken`, which has the value `Token`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetPPPAuthPrompt`

The PPP key `AuthPrompt`, whose value is of type `CFString`.

This key can be passed the following constants:

- `kSCValNetPPPAuthPromptBefore`, which has the value `Before`
- `kSCValNetPPPAuthPromptAfter`, which has the value `After`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetPPPAuthProtocol`

The PPP key `AuthProtocol`, whose value is of type `CFArray`, containing elements of type `CFString`.

This key can be passed the following constants:

- `kSCValNetPPPAuthProtocolCHAP`, which has the value `CHAP`
- `kSCValNetPPPAuthProtocolEAP`, which has the value `EAP`
- `kSCValNetPPPAuthProtocolMSCHAP1`, which has the value `MSCHAP1`
- `kSCValNetPPPAuthProtocolMSCHAP2`, which has the value `MSCHAP2`
- `kSCValNetPPPAuthProtocolPAP`, which has the value `PAP`

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetPPPCommAlternateRemoteAddress`

The PPP key `CommAlternateRemoteAddress`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetPPPCommConnectDelay`

The PPP key `CommConnectDelay`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommDisplayTerminalWindow

The PPP key `CommDisplayTerminalWindow`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommRedialCount

The PPP key `CommRedialCount`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommRedialEnabled

The PPP key `CommRedialEnabled`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommRedialInterval

The PPP key `CommRedialInterval`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommRemoteAddress

The PPP key `CommRemoteAddress`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommTerminalScript

The PPP key `CommTerminalScript`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCommUseTerminalScript

The PPP key `CommUseTerminalScript`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCCPEEnabled

The PPP key `CCPEEnabled`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCCPMPPE40Enabled

The PPP key `CCPMPPE40Enabled`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.4 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPCCPMPPE128Enabled

The PPP key `CCPMPPE128Enabled`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.4 and later.

Declared in `SCSchemaDefinitions.h`.

kSCPropNetPPPIPCCompressionVJ

The PPP key IPCCompressionVJ, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPIPCUsePeerDNS

The PPP key IPCUsePeerDNS, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.4 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPEchoEnabled

The PPP key LCPEchoEnabled, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPEchoFailure

The PPP key LCPEchoFailure, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPEchoInterval

The PPP key LCPEchoInterval, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPCCompressionACField

The PPP key LCPCCompressionACField, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPCCompressionPField

The PPP key LCPCCompressionPField, whose value is of type CFNumber and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPMRU

The PPP key LCPMRU, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPMTU

The PPP key LCPMTU, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPReceiveACCM

The PPP key LCPReceiveACCM, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetPPPLCPTransmitACCM

The PPP key LCPTransmitACCM, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## L2TP Entity Keys

Keys that correspond to values in the kSCEntNetL2TP dictionary.

kSCPropNetL2TPIPecSharedSecret

kSCPropNetL2TPIPecSharedSecretEncryption

kSCPropNetL2TPTransport

### Constants

kSCPropNetL2TPIPecSharedSecret

The L2TP key IPecSharedSecret, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetL2TPIPecSharedSecretEncryption

The L2TP key IPecSharedSecretEncryption, whose value is of type CFString.

This key can be passed the following constant:

- kSCValNetL2TPIPecSharedSecretEncryptionKeychain, which has the value Keychain

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetL2TPTransport

The L2TP key Transport, whose value is of type CFString.

This key can be passed the following constants:

- kSCValNetL2TPTransportIP, which has the value IP
- kSCValNetL2TPTransportIPec, which has the value IPec

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## Proxies Entity Keys

Keys that correspond to values in the kSCEntNetProxies dictionary.

kSCPropNetProxiesExceptionsList  
 kSCPropNetProxiesExcludeSimpleHostnames  
 kSCPropNetProxiesFTPEnable  
 kSCPropNetProxiesFTPPassive  
 kSCPropNetProxiesFTPPort  
 kSCPropNetProxiesFTPProxy  
 kSCPropNetProxiesGopherEnable  
 kSCPropNetProxiesGopherPort  
 kSCPropNetProxiesGopherProxy  
 kSCPropNetProxiesHTTPEnable  
 kSCPropNetProxiesHTTPPort  
 kSCPropNetProxiesHTTPProxy  
 kSCPropNetProxiesHTTPSEnable  
 kSCPropNetProxiesHTTPSPort  
 kSCPropNetProxiesHTTPSPProxy  
 kSCPropNetProxiesRTSPEnable  
 kSCPropNetProxiesRTSPPort  
 kSCPropNetProxiesRTSPProxy  
 kSCPropNetProxiesSOCKSEnable  
 kSCPropNetProxiesSOCKSPort  
 kSCPropNetProxiesSOCKSProxy  
 kSCPropNetProxiesProxyAutoConfigEnable  
 kSCPropNetProxiesProxyAutoConfigURLString  
 kSCPropNetProxiesProxyAutoDiscoveryEnable

### Constants

kSCPropNetProxiesExceptionsList

**The Proxies key** ExceptionsList, whose value is of type CFArrary, containing elements of type CFString.

**Available in Mac OS X v10.3 and later.**

**Declared in** SCSchemaDefinitions.h.

kSCPropNetProxiesExcludeSimpleHostnames

**The Proxies key** ExcludeSimpleHostnames, whose value is of type CFNumber and is equal to 0 or 1.

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

**Declared in** SCSchemaDefinitions.h.

kSCPropNetProxiesFTPEnable

**The Proxies key** FTPEnable, whose value is of type CFNumber and is equal to 0 or 1.

**Available in Mac OS X v10.3 and later.**

**Declared in** SCSchemaDefinitions.h.

kSCPropNetProxiesFTPPassive

**The Proxies key** FTPPassive, whose value is of type CFNumber and is equal to 0 or 1.

**Available in Mac OS X v10.3 and later.**

**Declared in** SCSchemaDefinitions.h.

kSCPropNetProxiesFTPPort

**The Proxies key** FTPPort, whose value is of type CFNumber.

**Available in Mac OS X v10.3 and later.**

**Declared in** SCSchemaDefinitions.h.

- `kSCPropNetProxiesFTPProxy`  
The Proxies key `FTPProxy`, whose value is of type `CFString`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesGopherEnable`  
The Proxies key `GopherEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesGopherPort`  
The Proxies key `GopherPort`, whose value is of type `CFNumber`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesGopherProxy`  
The Proxies key `GopherProxy`, whose value is of type `CFString`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesHTTPEnable`  
The Proxies key `HTTPEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesHTTPPort`  
The Proxies key `HTTPPort`, whose value is of type `CFNumber`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesHTTPProxy`  
The Proxies key `HTTPProxy`, whose value is of type `CFString`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesHTTPSEnable`  
The Proxies key `HTTPSEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesHTTPSPort`  
The Proxies key `HTTPSPort`, whose value is of type `CFNumber`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.
- `kSCPropNetProxiesHTTPSPProxy`  
The Proxies key `HTTPSPProxy`, whose value is of type `CFString`.  
Available in Mac OS X v10.3 and later.  
Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesRTSPEnable`

The Proxies key `RTSPEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesRTSPPort`

The Proxies key `RTSPPort`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesRTSPProxy`

The Proxies key `RTSPProxy`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesSOCKSEnable`

The Proxies key `SOCKSEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesSOCKSPort`

The Proxies key `SOCKSPort`, whose value is of type `CFNumber`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesSOCKSPProxy`

The Proxies key `SOCKSPProxy`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesProxyAutoConfigEnable`

The Proxies key `ProxyAutoConfigEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesProxyAutoConfigURLString`

The Proxies key `ProxyAutoConfigURLString`, whose value is of type `CFString`.

Available in Mac OS X v10.3 and later.

Declared in `SCSchemaDefinitions.h`.

`kSCPropNetProxiesProxyAutoDiscoveryEnable`

The Proxies key `ProxyAutoDiscoveryEnable`, whose value is of type `CFNumber` and is equal to 0 or 1.

Available in Mac OS X v10.4 and later.

Declared in `SCSchemaDefinitions.h`.

## SMB Entity Keys

Keys that correspond to values in the `kSCEntNetSMB` dictionary.

kSCPropNetSMBNetBIOSName  
 kSCPropNetSMBNetBIOSNodeType  
 kSCPropNetSMBNetBIOSScope  
 kSCPropNetSMBWINSAddresses  
 kSCPropNetSMBWorkgroup

**Constants**

kSCPropNetSMBNetBIOSName

The SMB key NetBIOSName, whose value is of type CFString.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetSMBNetBIOSNodeType

The SMB key NetBIOSNodeType, whose value is of type CFString.

This key can be passed the following constants:

- kSCValNetSMBNetBIOSNodeTypeBroadcast, which has the value Broadcast
- kSCValNetSMBNetBIOSNodeTypePeer, which has the value Peer
- kSCValNetSMBNetBIOSNodeTypeMixed, which has the value Mixed
- kSCValNetSMBNetBIOSNodeTypeHybrid, which has the value Hybrid

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetSMBNetBIOSScope

The SMB key NetBIOSScope, whose value is of type CFString.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetSMBWINSAddresses

The SMB key WINSAddresses, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

kSCPropNetSMBWorkgroup

The SMB key Workgroup, whose value is of type CFString.

Available in Mac OS X v10.5 and later.

Declared in SCSchemaDefinitions.h.

**CompUsers Entity Keys**

Keys that correspond to values in the kSCCompUsers dictionary.

kSCEntUsersConsoleUser

**Constants**

kSCEntUsersConsoleUser

The CompUsers key ConsoleUser.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## CompSystem Entity Keys

Keys that correspond to values in the `kSCCompSystem` dictionary.

`kSCPropSystemComputerName`  
`kSCPropSystemComputerNameEncoding`

### Constants

`kSCPropSystemComputerName`  
**The `CompSystem` key `ComputerName`, whose value is of type `CFString`.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCPropSystemComputerNameEncoding`  
**The `CompSystem` key `ComputerNameEncoding`, whose value is of type `CFNumber`.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

## Dynamic Store Domain Prefixes

Prefixes used to access information in the dynamic store.

`kSCDynamicStoreDomainFile`  
`kSCDynamicStoreDomainPlugin`  
`kSCDynamicStoreDomainSetup`  
`kSCDynamicStoreDomainState`  
`kSCDynamicStoreDomainPrefs`

### Constants

`kSCDynamicStoreDomainFile`  
**The `File:` prefix.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCDynamicStoreDomainPlugin`  
**The `Plugin:` prefix.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCDynamicStoreDomainSetup`  
**The `Setup:` prefix.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

`kSCDynamicStoreDomainState`  
**The `State:` prefix.**  
 Available in Mac OS X v10.3 and later.  
 Declared in `SCSchemaDefinitions.h`.

kSCDynamicStoreDomainPrefs

The Prefs: prefix.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

## Dynamic Store Entity Keys

Keys that correspond to values in the dynamic store.

kSCDynamicStorePropSetupCurrentSet

kSCDynamicStorePropSetupLastUpdated

kSCDynamicStorePropNetInterfaces

kSCDynamicStorePropNetPrimaryInterface

kSCDynamicStorePropNetPrimaryService

kSCDynamicStorePropNetServiceIDs

### Constants

kSCDynamicStorePropSetupCurrentSet

The dynamic store key CurrentSet, whose value is of type CFNumber.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCDynamicStorePropSetupLastUpdated

The dynamic store key LastUpdated.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCDynamicStorePropNetInterfaces

The dynamic store key Interfaces, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCDynamicStorePropNetPrimaryInterface

The dynamic store key PrimaryInterface, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCDynamicStorePropNetPrimaryService

The dynamic store key PrimaryService, whose value is of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

kSCDynamicStorePropNetServiceIDs

The dynamic store key ServiceIDs, whose value is of type CFArray, containing elements of type CFString.

Available in Mac OS X v10.3 and later.

Declared in SCSchemaDefinitions.h.

# SystemConfiguration Reference

---

<b>Framework:</b>	SystemConfiguration
<b>Declared in</b>	SystemConfiguration.h

## Overview

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

## Functions

### SCCopyLastError

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

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

#### Return Value

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

#### Discussion

Call the `CFErrorGetCode` function on the returned object to get the underlying error-code integer. See [“Status and Error Codes”](#) (page 170) for descriptions of these codes. For more on `CFErrorRef` objects, see *CFError Reference*.

#### Availability

Available in Mac OS X v10.5 and later.

#### Declared In

`SystemConfiguration.h`

### SCError

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

```
int SError (
    void
);
```

**Return Value**

The most recent status or error code generated as the result of calling a function defined by the System Configuration framework. See “[Status and Error Codes](#)” (page 170) for descriptions of these codes.

**Availability**

Available in Mac OS X v10.1 and later.

**Related Sample Code**

MoreSCF

SimpleDial

SimpleReach

**Declared In**

SystemConfiguration.h

**SErrorString**

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

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

**Parameters**

*status*

A status or error code described in “[Status and Error Codes](#)” (page 170). You typically get this code by calling `SError` (page 169) or `SCCopyLastError` (page 169).

**Return Value**

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

**Availability**

Available in Mac OS X v10.1 and later.

**Declared In**

SystemConfiguration.h

## Constants

### Status and Error Codes

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

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

**Constants**`kSCStatusOK`**The call was successful.****Available in Mac OS X v10.1 and later.****Declared in** `SystemConfiguration.h`.`kSCStatusFailed`**A nonspecific failure occurred.****Available in Mac OS X v10.1 and later.****Declared in** `SystemConfiguration.h`.`kSCStatusInvalidArgument`**An invalid argument was specified.****Available in Mac OS X v10.1 and later.****Declared in** `SystemConfiguration.h`.`kSCStatusAccessError`**Permission is denied; you must be root to obtain a lock. As a result, the function could not create or access preferences.****Available in Mac OS X v10.1 and later.****Declared in** `SystemConfiguration.h`.`kSCStatusNoKey`**No such key.****Available in Mac OS X v10.1 and later.****Declared in** `SystemConfiguration.h`.`kSCStatusKeyExists`**The key is already defined.****Available in Mac OS X v10.1 and later.****Declared in** `SystemConfiguration.h`.

`kSCStatusLocked`

A lock is already held.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNeedLock`

A lock is required for this operation.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoStoreSession`

The configuration daemon session is not active.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoStoreServer`

The configuration daemon is not available or no longer available.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNotifierActive`

Notifier is currently active.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoPrefsSession`

The preferences session is not active.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusPrefsBusy`

A preferences update is currently in progress.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoConfigFile`

The configuration file cannot be found.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusNoLink`

No such link exists.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusStale`

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

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusMaxLink`

The maximum link count is exceeded.

Available in Mac OS X v10.2 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusReachabilityUnknown`

Network reachability cannot be determined.

Available in Mac OS X v10.1 and later.

Declared in `SystemConfiguration.h`.

`kSCStatusConnectionNoService`

Network service for the connection is not available.

Available in Mac OS X v10.6 and later.

Declared in `SystemConfiguration.h`.

## Error Domain

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

```
const CFStringRef    kCFErrorDomainSystemConfiguration;
```

### Constants

`kCFErrorDomainSystemConfiguration`

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

Available in Mac OS X v10.5 and later.

Declared in `SystemConfiguration.h`.



# Document Revision History

---

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

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

## REVISION HISTORY

### Document Revision History

# Index

---

## Numerals

---

6to4 Entity Keys [152](#)

## A

---

AirPort Dictionary Keys [143](#)

AppleTalk Entity Keys [144](#)

## C

---

Component Keys [138](#)

CompSystem Entity Keys [167](#)

CompUsers Entity Keys [166](#)

## D

---

DHCPInfoGetLeaseStartTime **function** [25](#)

DHCPInfoGetOptionData **function** [25](#)

DNS Entity Keys [146](#)

Dynamic Store Domain Prefixes [167](#)

Dynamic Store Entity Keys [168](#)

Dynamic Store Options Keys [23](#)

## E

---

Error Domain [173](#)

Ethernet Bond Aggregation Status [83](#)

Ethernet Bond Status Constants [84](#)

Ethernet Entity Keys [147](#)

## G

---

Generic Keys [137](#)

## H

---

Hostnames Dictionary Keys [140](#)

## I

---

Interface Dictionary Keys [140](#)

Interface Entity Keys [147](#)

IPSec Entity Keys [148](#)

IPv4 Entity Keys [149](#)

IPv6 Entity Keys [151](#)

## K

---

kCFErrorDomainSystemConfiguration **constant** [173](#)

kSCBondStatusDeviceAggregationStatus **constant** [84](#)

kSCBondStatusDeviceCollecting **constant** [85](#)

kSCBondStatusDeviceDistributing **constant** [85](#)

kSCBondStatusLinkInvalid **constant** [84](#)

kSCBondStatusNoPartner **constant** [84](#)

kSCBondStatusNotInActiveGroup **constant** [84](#)

kSCBondStatusOK **constant** [84](#)

kSCBondStatusUnknown **constant** [84](#)

kSCCompGlobal **constant** [139](#)

kSCCompHostNames **constant** [139](#)

kSCCompInterface **constant** [139](#)

kSCCompNetwork **constant** [138](#)

kSCCompService **constant** [138](#)

kSCCompSystem **constant** [139](#)

kSCCompUsers **constant** [139](#)

kSCDynamicStoreDomainFile **constant** [167](#)

kSCDynamicStoreDomainPlugin **constant** [167](#)

- kSCDynamicStoreDomainPrefs **constant** 168
- kSCDynamicStoreDomainSetup **constant** 167
- kSCDynamicStoreDomainState **constant** 167
- kSCDynamicStorePropNetInterfaces **constant** 168
- kSCDynamicStorePropNetPrimaryInterface **constant** 168
- kSCDynamicStorePropNetPrimaryService **constant** 168
- kSCDynamicStorePropNetServiceIDs **constant** 168
- kSCDynamicStorePropSetupCurrentSet **constant** 168
- kSCDynamicStorePropSetupLastUpdated **constant** 168
- kSCDynamicStoreUseSessionKeys **constant** 23
- kSCEntNet6to4 **constant** 143
- kSCEntNetAirPort **constant** 141
- kSCEntNetAppleTalk **constant** (Deprecated in Mac OS X v10.6) 141
- kSCEntNetDHCP **constant** 141
- kSCEntNetDNS **constant** 141
- kSCEntNetEthernet **constant** 141
- kSCEntNetFireWire **constant** 141
- kSCEntNetInterface **constant** 142
- kSCEntNetIPSec **constant** 142
- kSCEntNetIPv4 **constant** 142
- kSCEntNetIPv6 **constant** 142
- kSCEntNetL2TP **constant** 142
- kSCEntNetLink **constant** 142
- kSCEntNetModem **constant** 142
- kSCEntNetPPP **constant** 142
- kSCEntNetPPPoE **constant** 142
- kSCEntNetPPPSerial **constant** 142
- kSCEntNetPPTP **constant** 143
- kSCEntNetProxies **constant** 143
- kSCEntNetSMB **constant** 143
- kSCEntUsersConsoleUser **constant** 166
- kSCNetworkConnectionBytesIn **constant** 103
- kSCNetworkConnectionBytesOut **constant** 103
- kSCNetworkConnectionConnected **constant** 100
- kSCNetworkConnectionConnecting **constant** 100
- kSCNetworkConnectionDisconnected **constant** 100
- kSCNetworkConnectionDisconnecting **constant** 100
- kSCNetworkConnectionErrorsIn **constant** 103
- kSCNetworkConnectionErrorsOut **constant** 103
- kSCNetworkConnectionInvalid **constant** 100
- kSCNetworkConnectionPacketsIn **constant** 103
- kSCNetworkConnectionPacketsOut **constant** 103
- kSCNetworkConnectionPPPAuthenticating **constant** 101
- kSCNetworkConnectionPPPConnected **constant** 102
- kSCNetworkConnectionPPPConnectingLink **constant** 101
- kSCNetworkConnectionPPPdialOnTraffic **constant** 101
- kSCNetworkConnectionPPPDisconnected **constant** 101
- kSCNetworkConnectionPPPDisconnectingLink **constant** 102
- kSCNetworkConnectionPPPHoldingLinkOff **constant** 102
- kSCNetworkConnectionPPPInitializing **constant** 101
- kSCNetworkConnectionPPPNegotiatingLink **constant** 101
- kSCNetworkConnectionPPPNegotiatingNetwork **constant** 102
- kSCNetworkConnectionPPPSuspended **constant** 102
- kSCNetworkConnectionPPPTerminating **constant** 102
- kSCNetworkConnectionPPPWaitingForCallback **constant** 101
- kSCNetworkConnectionPPPWaitingForRedial **constant** 102
- kSCNetworkConnectionSelectionOptionOnDemandHostName **constant** 104
- kSCNetworkConnectionSelectionOptionOnDemandRetry **constant** 104
- kSCNetworkFlagsConnectionAutomatic **constant** 40
- kSCNetworkFlagsConnectionRequired **constant** 40
- kSCNetworkFlagsInterventionRequired **constant** 40
- kSCNetworkFlagsIsDirect **constant** 40
- kSCNetworkFlagsIsLocalAddress **constant** 40
- kSCNetworkFlagsReachable **constant** 40
- kSCNetworkFlagsTransientConnection **constant** 39
- kSCNetworkInterfaceIPv4 **constant** 87
- kSCNetworkInterfaceType6to4 **constant** 85
- kSCNetworkInterfaceTypeBluetooth **constant** 85
- kSCNetworkInterfaceTypeBond **constant** 85
- kSCNetworkInterfaceTypeEthernet **constant** 85
- kSCNetworkInterfaceTypeFireWire **constant** 86
- kSCNetworkInterfaceTypeIEEE80211 **constant** 86
- kSCNetworkInterfaceTypeIPSec **constant** 86
- kSCNetworkInterfaceTypeIPv4 **constant** 87
- kSCNetworkInterfaceTypeIrDA **constant** 86
- kSCNetworkInterfaceTypeL2TP **constant** 86
- kSCNetworkInterfaceTypeModem **constant** 86
- kSCNetworkInterfaceTypePPP **constant** 86
- kSCNetworkInterfaceTypePPTP **constant** 86
- kSCNetworkInterfaceTypeSerial **constant** 86
- kSCNetworkInterfaceTypeVLAN **constant** 86
- kSCNetworkInterfaceTypeWWAN **constant** 87
- kSCNetworkProtocolTypeAppleTalk **constant** (Deprecated in Mac OS X v10.6) 87
- kSCNetworkProtocolTypeDNS **constant** 87

- kSCNetworkProtocolTypeIPv4 **constant** 87
- kSCNetworkProtocolTypeIPv6 **constant** 87
- kSCNetworkProtocolTypeProxies **constant** 88
- kSCNetworkProtocolTypeSMB **constant** 88
- kSCNetworkReachabilityFlagsConnectionOnDemand **constant** 113
- kSCNetworkReachabilityFlagsConnectionOnTraffic **constant** 112
- kSCNetworkReachabilityFlagsConnectionRequired **constant** 112
- kSCNetworkReachabilityFlagsInterventionRequired **constant** 113
- kSCNetworkReachabilityFlagsIsDirect **constant** 113
- kSCNetworkReachabilityFlagsIsLocalAddress **constant** 113
- kSCNetworkReachabilityFlagsReachable **constant** 112
- kSCNetworkReachabilityFlagsTransientConnection **constant** 112
- kSCPrefCurrentSet **constant** 138
- kSCPreferencesNotificationApply **constant** 128
- kSCPreferencesNotificationCommit **constant** 128
- kSCPrefNetworkServices **constant** 138
- kSCPrefSets **constant** 138
- kSCPrefSystem **constant** 138
- kSCPropInterfaceName **constant** 137
- kSCPropMACAddress **constant** 137
- kSCPropNet6to4Relay **constant** 152
- kSCPropNetAirPortAllowNetCreation **constant** 143
- kSCPropNetAirPortAuthPassword **constant** 143
- kSCPropNetAirPortAuthPasswordEncryption **constant** 143
- kSCPropNetAirPortJoinMode **constant** 144
- kSCPropNetAirPortPowerEnabled **constant** 144
- kSCPropNetAirPortPreferredNetwork **constant** 144
- kSCPropNetAirPortSavePasswords **constant** 144
- kSCPropNetAppleTalkComputerName **constant** (Deprecated in Mac OS X v10.6) 144
- kSCPropNetAppleTalkComputerNameEncoding **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkConfigMethod **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkDefaultZone **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkNetworkID **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkNetworkRange **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkNodeID **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkSeedNetworkRange **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetAppleTalkSeedZones **constant** (Deprecated in Mac OS X v10.6) 145
- kSCPropNetDNSDomainName **constant** 146
- kSCPropNetDNSOptions **constant** 146
- kSCPropNetDNSSearchDomains **constant** 146
- kSCPropNetDNSSearchOrder **constant** 146
- kSCPropNetDNSServerAddresses **constant** 146
- kSCPropNetDNSServerPort **constant** 146
- kSCPropNetDNSServerTimeout **constant** 146
- kSCPropNetDNSSortList **constant** 147
- kSCPropNetDNSSupplementalMatchDomains **constant** 147
- kSCPropNetDNSSupplementalMatchOrders **constant** 147
- kSCPropNetEthernetMediaOptions **constant** 147
- kSCPropNetEthernetMediaSubType **constant** 147
- kSCPropNetEthernetMTU **constant** 147
- kSCPropNetInterfaceDeviceName **constant** 148
- kSCPropNetInterfaceHardware **constant** 148
- kSCPropNetInterfaces **constant** 140
- kSCPropNetInterfaceSubType **constant** 148
- kSCPropNetInterfaceSupportsModemOnHold **constant** 148
- kSCPropNetInterfaceType **constant** 148
- kSCPropNetIPSecAuthenticationMethod **constant** 149
- kSCPropNetIPSecLocalCertificate **constant** 149
- kSCPropNetIPSecLocalIdentifier **constant** 149
- kSCPropNetIPSecLocalIdentifierType **constant** 149
- kSCPropNetIPSecSharedSecret **constant** 149
- kSCPropNetIPSecSharedSecretEncryption **constant** 149
- kSCPropNetIPv4Addresses **constant** 150
- kSCPropNetIPv4BroadcastAddresses **constant** 151
- kSCPropNetIPv4ConfigMethod **constant** 150
- kSCPropNetIPv4DestAddresses **constant** 150
- kSCPropNetIPv4DHCPClientID **constant** 150
- kSCPropNetIPv4Router **constant** 150
- kSCPropNetIPv4SubnetMasks **constant** 150
- kSCPropNetIPv6Addresses **constant** 151
- kSCPropNetIPv6ConfigMethod **constant** 151
- kSCPropNetIPv6DestAddresses **constant** 151
- kSCPropNetIPv6Flags **constant** 151
- kSCPropNetIPv6PrefixLength **constant** 151
- kSCPropNetIPv6Router **constant** 152
- kSCPropNetL2TPIPSecSharedSecret **constant** 162
- kSCPropNetL2TPIPSecSharedSecretEncryption **constant** 162
- kSCPropNetL2TPTransport **constant** 162
- kSCPropNetLinkActive **constant** 152
- kSCPropNetLinkDetaching **constant** 152
- kSCPropNetLocalHostName **constant** 140

- kSCPropNetModemAccessPointName **constant** 153
- kSCPropNetModemConnectionPersonality **constant** 153
- kSCPropNetModemConnectionScript **constant** 153
- kSCPropNetModemConnectSpeed **constant** 153
- kSCPropNetModemDataCompression **constant** 153
- kSCPropNetModemDeviceContextID **constant** 153
- kSCPropNetModemDeviceModel **constant** 154
- kSCPropNetModemDeviceVendor **constant** 154
- kSCPropNetModemDialMode **constant** 154
- kSCPropNetModemErrorCorrection **constant** 154
- kSCPropNetModemHoldCallWaitingAudibleAlert **constant** 154
- kSCPropNetModemHoldDisconnectOnAnswer **constant** 154
- kSCPropNetModemHoldEnabled **constant** 154
- kSCPropNetModemHoldReminder **constant** 154
- kSCPropNetModemHoldReminderTime **constant** 154
- kSCPropNetModemNote **constant** 155
- kSCPropNetModemPulseDial **constant** 155
- kSCPropNetModemSpeaker **constant** 155
- kSCPropNetModemSpeed **constant** 155
- kSCPropNetOverridePrimary **constant** 139
- kSCPropNetPPPACSPEnabled **constant** 156
- kSCPropNetPPPAuthEAPPlugins **constant** 158
- kSCPropNetPPPAuthName **constant** 159
- kSCPropNetPPPAuthPassword **constant** 159
- kSCPropNetPPPAuthPasswordEncryption **constant** 159
- kSCPropNetPPPAuthPrompt **constant** 159
- kSCPropNetPPPAuthProtocol **constant** 159
- kSCPropNetPPPCCPEnabled **constant** 160
- kSCPropNetPPPCCPMPPE128Enabled **constant** 160
- kSCPropNetPPPCCPMPPE40Enabled **constant** 160
- kSCPropNetPPPCommAlternateRemoteAddress **constant** 159
- kSCPropNetPPPCommConnectDelay **constant** 159
- kSCPropNetPPPCommDisplayTerminalWindow **constant** 160
- kSCPropNetPPPCommRedialCount **constant** 160
- kSCPropNetPPPCommRedialEnabled **constant** 160
- kSCPropNetPPPCommRedialInterval **constant** 160
- kSCPropNetPPPCommRemoteAddress **constant** 160
- kSCPropNetPPPCommTerminalScript **constant** 160
- kSCPropNetPPPCommUseTerminalScript **constant** 160
- kSCPropNetPPPConnectTime **constant** 157
- kSCPropNetPPPDeviceLastCause **constant** 157
- kSCPropNetPPPDialOnDemand **constant** 157
- kSCPropNetPPPDDisconnectOnFastUserSwitch **constant** 157
- kSCPropNetPPPDDisconnectOnIdle **constant** 157
- kSCPropNetPPPDDisconnectOnLogout **constant** 157
- kSCPropNetPPPDDisconnectOnSleep **constant** 157
- kSCPropNetPPPDDisconnectTime **constant** 157
- kSCPropNetPPPIdleReminder **constant** 158
- kSCPropNetPPPIdleReminderTimer **constant** 157
- kSCPropNetPPPIPCPCCompressionVJ **constant** 161
- kSCPropNetPPPIPCPCUsePeerDNS **constant** 161
- kSCPropNetPPPLastCause **constant** 158
- kSCPropNetPPPLCPCCompressionACField **constant** 161
- kSCPropNetPPPLCPCCompressionPField **constant** 161
- kSCPropNetPPPLCPEchoEnabled **constant** 161
- kSCPropNetPPPLCPEchoFailure **constant** 161
- kSCPropNetPPPLCPEchoInterval **constant** 161
- kSCPropNetPPPLCPMRU **constant** 161
- kSCPropNetPPPLCPMTU **constant** 161
- kSCPropNetPPPLCPReceiveACCM **constant** 161
- kSCPropNetPPPLCPTransmitACCM **constant** 162
- kSCPropNetPPPLogFile **constant** 158
- kSCPropNetPPPOverridePrimary **constant** 139
- kSCPropNetPPPPlugins **constant** 158
- kSCPropNetPPPRetryConnectTime **constant** 158
- kSCPropNetPPPSessionTimer **constant** 158
- kSCPropNetPPPStatus **constant** 158
- kSCPropNetPPPUseSessionTimer **constant** 158
- kSCPropNetPPPVerboseLogging **constant** 158
- kSCPropNetProxiesExceptionsList **constant** 163
- kSCPropNetProxiesExcludeSimpleHostnames **constant** 163
- kSCPropNetProxiesFTPEnable **constant** 163
- kSCPropNetProxiesFTPPassive **constant** 163
- kSCPropNetProxiesFTPPort **constant** 163
- kSCPropNetProxiesFTPProxy **constant** 164
- kSCPropNetProxiesGopherEnable **constant** 164
- kSCPropNetProxiesGopherPort **constant** 164
- kSCPropNetProxiesGopherProxy **constant** 164
- kSCPropNetProxiesHTTPEnable **constant** 164
- kSCPropNetProxiesHTTPPort **constant** 164
- kSCPropNetProxiesHTTPProxy **constant** 164
- kSCPropNetProxiesHTTPSEnable **constant** 164
- kSCPropNetProxiesHTTPSPort **constant** 164
- kSCPropNetProxiesHTTPSProxy **constant** 164
- kSCPropNetProxiesProxyAutoConfigEnable **constant** 165
- kSCPropNetProxiesProxyAutoConfigURLString **constant** 165
- kSCPropNetProxiesProxyAutoDiscoveryEnable **constant** 165
- kSCPropNetProxiesRTSPEnable **constant** 165
- kSCPropNetProxiesRTSPPort **constant** 165
- kSCPropNetProxiesRTSPProxy **constant** 165

[kSCPropNetProxiesSOCKSEnable constant 165](#)  
[kSCPropNetProxiesSOCKSPort constant 165](#)  
[kSCPropNetProxiesSOCKSProxy constant 165](#)  
[kSCPropNetServiceOrder constant 139](#)  
[kSCPropNetSMBNetBIOSName constant 166](#)  
[kSCPropNetSMBNetBIOSNodeType constant 166](#)  
[kSCPropNetSMBNetBIOSScope constant 166](#)  
[kSCPropNetSMBWINSAddresses constant 166](#)  
[kSCPropNetSMBWorkgroup constant 166](#)  
[kSCPropSystemComputerName constant 167](#)  
[kSCPropSystemComputerNameEncoding constant 167](#)  
[kSCPropUserDefinedName constant 137](#)  
[kSCPropVersion constant 137](#)  
[kSCStatusAccessError constant 171](#)  
[kSCStatusConnectionNoService constant 173](#)  
[kSCStatusFailed constant 171](#)  
[kSCStatusInvalidArgument constant 171](#)  
[kSCStatusKeyExists constant 171](#)  
[kSCStatusLocked constant 172](#)  
[kSCStatusMaxLink constant 173](#)  
[kSCStatusNeedLock constant 172](#)  
[kSCStatusNoConfigFile constant 172](#)  
[kSCStatusNoKey constant 171](#)  
[kSCStatusNoLink constant 172](#)  
[kSCStatusNoPrefsSession constant 172](#)  
[kSCStatusNoStoreServer constant 172](#)  
[kSCStatusNoStoreSession constant 172](#)  
[kSCStatusNotifierActive constant 172](#)  
[kSCStatusOK constant 171](#)  
[kSCStatusPrefsBusy constant 172](#)  
[kSCStatusReachabilityUnknown constant 173](#)  
[kSCStatusStale constant 172](#)

## L

---

[L2TP Entity Keys 162](#)  
[Link Entity Keys 152](#)

## M

---

[Modem Entity Keys 152](#)

## N

---

[Network Connection Flags 39](#)  
[Network Connection PPP Status Values 100](#)  
[Network Connection Status Values 100](#)  
[Network Dictionary Keys 139](#)  
[Network Entity Keys 140](#)

[Network Interface Types 85](#)  
[Network Protocol Types 87](#)  
[Network Reachability Flags 112](#)

## P

---

[PPP Entity Keys 155](#)  
[Preference Keys 138](#)  
[Preferences Notification Values 128](#)  
[Proxies Entity Keys 162](#)

## S

---

[SCBondInterfaceCopyAll function 46](#)  
[SCBondInterfaceCopyAvailableMemberInterfaces function 46](#)  
[SCBondInterfaceCopyStatus function 46](#)  
[SCBondInterfaceCreate function 47](#)  
[SCBondInterfaceGetMemberInterfaces function 47](#)  
[SCBondInterfaceGetOptions function 48](#)  
[SCBondInterfaceRef data type 82](#)  
[SCBondInterfaceRemove function 48](#)  
[SCBondInterfaceSetLocalizedDisplayName function 48](#)  
[SCBondInterfaceSetMemberInterfaces function 49](#)  
[SCBondInterfaceSetOptions function 49](#)  
[SCBondStatusGetInterfaceStatus function 50](#)  
[SCBondStatusGetMemberInterfaces function 50](#)  
[SCBondStatusGetTypeID function 51](#)  
[SCBondStatusRef data type 82](#)  
[SCCopyLastError function 169](#)  
[SCDynamicStoreAddTemporaryValue function 13](#)  
[SCDynamicStoreAddValue function 13](#)  
[SCDynamicStoreCallBack data type 22](#)  
[SCDynamicStoreContext data type 22](#)  
[SCDynamicStoreCopyComputerName function 27](#)  
[SCDynamicStoreCopyConsoleUser function 28](#)  
[SCDynamicStoreCopyDHCPInfo function 26](#)  
[SCDynamicStoreCopyKeyList function 14](#)  
[SCDynamicStoreCopyLocalHostName function 28](#)  
[SCDynamicStoreCopyLocation function 29](#)  
[SCDynamicStoreCopyMultiple function 14](#)  
[SCDynamicStoreCopyNotifiedKeys function 15](#)  
[SCDynamicStoreCopyProxies function 29](#)  
[SCDynamicStoreCopyValue function 15](#)  
[SCDynamicStoreCreate function 16](#)  
[SCDynamicStoreCreateRunLoopSource function 17](#)  
[SCDynamicStoreCreateWithOptions function 17](#)  
[SCDynamicStoreGetTypeID function 18](#)  
[SCDynamicStoreKeyCreate function 31](#)

- SCDynamicStoreKeyCreateComputerName **function** [32](#)
- SCDynamicStoreKeyCreateConsoleUser **function** [32](#)
- SCDynamicStoreKeyCreateHostNames **function** [32](#)
- SCDynamicStoreKeyCreateLocation **function** [33](#)
- SCDynamicStoreKeyCreateNetworkGlobalEntity **function** [33](#)
- SCDynamicStoreKeyCreateNetworkInterface **function** [34](#)
- SCDynamicStoreKeyCreateNetworkInterfaceEntity **function** [35](#)
- SCDynamicStoreKeyCreateNetworkServiceEntity **function** [35](#)
- SCDynamicStoreKeyCreateProxies **function** [36](#)
- SCDynamicStoreNotifyValue **function** [19](#)
- SCDynamicStoreRef **data type** [23](#)
- SCDynamicStoreRemoveValue **function** [19](#)
- SCDynamicStoreSetDispatchQueue **function** [20](#)
- SCDynamicStoreSetMultiple **function** [20](#)
- SCDynamicStoreSetNotificationKeys **function** [21](#)
- SCDynamicStoreSetValue **function** [21](#)
- SCError **function** [169](#)
- SCErrorString **function** [170](#)
- SCNetworkCheckReachabilityByAddress **function** [\(Deprecated in Mac OS X v10.6\) 37](#)
- SCNetworkCheckReachabilityByName **function** [\(Deprecated in Mac OS X v10.6\) 38](#)
- SCNetworkConnectionCallback **data type** [99](#)
- SCNetworkConnectionContext **structure** [99](#)
- SCNetworkConnectionCopyExtendedStatus **function** [90](#)
- SCNetworkConnectionCopyServiceID **function** [91](#)
- SCNetworkConnectionCopyStatistics **function** [91](#)
- SCNetworkConnectionCopyUserOptions **function** [92](#)
- SCNetworkConnectionCopyUserPreferences **function** [93](#)
- SCNetworkConnectionCreateWithServiceID **function** [94](#)
- SCNetworkConnectionGetStatus **function** [94](#)
- SCNetworkConnectionGetTypeID **function** [95](#)
- SCNetworkConnectionRef **data type** [98](#)
- SCNetworkConnectionScheduleWithRunLoop **function** [95](#)
- SCNetworkConnectionSetDispatchQueue **function** [96](#)
- SCNetworkConnectionStart **function** [96](#)
- SCNetworkConnectionStop **function** [97](#)
- SCNetworkConnectionUnscheduleFromRunLoop **function** [98](#)
- SCNetworkInterfaceCopyAll **function** [51](#)
- SCNetworkInterfaceCopyMediaOptions **function** [52](#)
- SCNetworkInterfaceCopyMediaSubTypeOptions **function** [52](#)
- SCNetworkInterfaceCopyMediaSubTypes **function** [53](#)
- SCNetworkInterfaceCopyMTU **function** [53](#)
- SCNetworkInterfaceCreateWithInterface **function** [54](#)
- SCNetworkInterfaceForceConfigurationRefresh **function** [54](#)
- SCNetworkInterfaceGetBSDName **function** [55](#)
- SCNetworkInterfaceGetConfiguration **function** [55](#)
- SCNetworkInterfaceGetExtendedConfiguration **function** [56](#)
- SCNetworkInterfaceGetHardwareAddressString **function** [56](#)
- SCNetworkInterfaceGetInterface **function** [57](#)
- SCNetworkInterfaceGetInterfaceType **function** [57](#)
- SCNetworkInterfaceGetLocalizedDisplayName **function** [58](#)
- SCNetworkInterfaceGetSupportedInterfaceTypes **function** [58](#)
- SCNetworkInterfaceGetSupportedProtocolTypes **function** [58](#)
- SCNetworkInterfaceGetTypeID **function** [59](#)
- SCNetworkInterfaceRef **data type** [82](#)
- SCNetworkInterfaceRefreshConfiguration **function** [\(Deprecated in Mac OS X v10.5\) 39](#)
- SCNetworkInterfaceSetConfiguration **function** [59](#)
- SCNetworkInterfaceSetExtendedConfiguration **function** [60](#)
- SCNetworkInterfaceSetMediaOptions **function** [60](#)
- SCNetworkInterfaceSetMTU **function** [61](#)
- SCNetworkProtocolGetConfiguration **function** [61](#)
- SCNetworkProtocolGetEnabled **function** [62](#)
- SCNetworkProtocolGetProtocolType **function** [62](#)
- SCNetworkProtocolGetTypeID **function** [62](#)
- SCNetworkProtocolRef **data type** [83](#)
- SCNetworkProtocolSetConfiguration **function** [63](#)
- SCNetworkProtocolSetEnabled **function** [63](#)
- SCNetworkReachabilityCallback **data type** [111](#)
- SCNetworkReachabilityContext **data type** [111](#)
- SCNetworkReachabilityCreateWithAddress **function** [106](#)
- SCNetworkReachabilityCreateWithAddressPair **function** [106](#)
- SCNetworkReachabilityCreateWithName **function** [107](#)
- SCNetworkReachabilityGetFlags **function** [108](#)
- SCNetworkReachabilityGetTypeID **function** [108](#)
- SCNetworkReachabilityRef **data type** [110](#)
- SCNetworkReachabilityScheduleWithRunLoop **function** [109](#)
- SCNetworkReachabilitySetCallback **function** [109](#)
- SCNetworkReachabilityUnscheduleFromRunLoop **function** [110](#)

- SCNetworkServiceAddProtocolType **function** 64
- SCNetworkServiceCopy **function** 64
- SCNetworkServiceCopyAll **function** 65
- SCNetworkServiceCopyProtocol **function** 65
- SCNetworkServiceCopyProtocols **function** 65
- SCNetworkServiceCreate **function** 66
- SCNetworkServiceEstablishDefaultConfiguration **function** 66
- SCNetworkServiceGetEnabled **function** 67
- SCNetworkServiceGetInterface **function** 67
- SCNetworkServiceGetName **function** 68
- SCNetworkServiceGetServiceID **function** 68
- SCNetworkServiceGetTypeID **function** 68
- SCNetworkServiceRef **data type** 83
- SCNetworkServiceRemove **function** 69
- SCNetworkServiceRemoveProtocolType **function** 69
- SCNetworkServiceSetEnabled **function** 69
- SCNetworkServiceSetName **function** 70
- SCNetworkSetAddService **function** 70
- SCNetworkSetContainsInterface **function** 71
- SCNetworkSetCopy **function** 71
- SCNetworkSetCopyAll **function** 72
- SCNetworkSetCopyCurrent **function** 72
- SCNetworkSetCopyServices **function** 73
- SCNetworkSetCreate **function** 73
- SCNetworkSetGetName **function** 74
- SCNetworkSetGetServiceOrder **function** 74
- SCNetworkSetGetSetID **function** 74
- SCNetworkSetGetTypeID **function** 75
- SCNetworkSetRef **data type** 83
- SCNetworkSetRemove **function** 75
- SCNetworkSetRemoveService **function** 75
- SCNetworkSetSetCurrent **function** 76
- SCNetworkSetSetName **function** 76
- SCNetworkSetSetServiceOrder **function** 77
- SCPreferencesAddValue **function** 117
- SCPreferencesApplyChanges **function** 117
- SCPreferencesCallback **data type** 127
- SCPreferencesCommitChanges **function** 118
- SCPreferencesContext **structure** 127
- SCPreferencesCopyKeyList **function** 118
- SCPreferencesCreate **function** 119
- SCPreferencesCreateWithAuthorization **function** 119
- SCPreferencesGetSignature **function** 120
- SCPreferencesGetTypeID **function** 121
- SCPreferencesGetValue **function** 121
- SCPreferencesLock **function** 121
- SCPreferencesPathCreateUniqueChild **function** 130
- SCPreferencesPathGetLink **function** 131
- SCPreferencesPathGetValue **function** 131
- SCPreferencesPathRemoveValue **function** 132
- SCPreferencesPathSetLink **function** 132
- SCPreferencesPathSetValue **function** 133
- SCPreferencesRef **data type** 126
- SCPreferencesRemoveValue **function** 122
- SCPreferencesScheduleWithRunLoop **function** 123
- SCPreferencesSetCallback **function** 123
- SCPreferencesSetComputerName **function** 135
- SCPreferencesSetDispatchQueue **function** 124
- SCPreferencesSetLocalHostName **function** 136
- SCPreferencesSetValue **function** 124
- SCPreferencesSynchronize **function** 125
- SCPreferencesUnlock **function** 125
- SCPreferencesUnscheduleFromRunLoop **function** 126
- SCVLANInterfaceCopyAll **function** 77
- SCVLANInterfaceCopyAvailablePhysicalInterfaces **function** 78
- SCVLANInterfaceCreate **function** 78
- SCVLANInterfaceGetOptions **function** 79
- SCVLANInterfaceGetPhysicalInterface **function** 79
- SCVLANInterfaceGetTag **function** 79
- SCVLANInterfaceRef **data type** 82
- SCVLANInterfaceRemove **function** 80
- SCVLANInterfaceSetLocalizedDisplayName **function** 80
- SCVLANInterfaceSetOptions **function** 81
- SCVLANInterfaceSetPhysicalInterfaceAndTag **function** 81
- Selection Options Dictionary Keys 103
- SMB Entity Keys 165
- Statistics Dictionary Keys 102
- Status and Error Codes 170