# PlayStation®2 EE Library Reference
# Release 2.4.1

# Inet Libraries

Publication date: October 2001

Sony Computer Entertainment Inc.
1-1, Akasaka 7-chome, Minato-ku
Tokyo 107-0052, Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

# Summary Table of Contents

# About This Manual

This is the Runtime Library Release 2.4.1 version of the *PlayStation®2 EE Library Reference - Inet Libraries* manual.

The purpose of this manual is to define all available PlayStation®2 EE inet library structures and functions. The companion *PlayStation®2 EE Library Overview - Inet Libraries* describes the structure and purpose of the libraries.

## Changes Since Last Release

### Chapter 2: Network Socket Library

- Descriptions of the following functions have been added:

  sceInsockAbort()
  sceInsockSetRecvTimeout()
  sceInsockSetSendTimeout()
  sceInsockTerminate()

## Related Documentation

Library specifications for the IOP can be found in the *PlayStation®2 IOP Library Reference* manuals and the *PlayStation®2 IOP Library Overview* manuals.

**Note:** the Developer Support Web site posts current developments regarding the Libraries and also provides notice of future documentation releases and upgrades.

## Typographic Conventions

Certain Typographic Conventions are used throughout this manual to clarify the meaning of the text:

| Convention | Meaning |
|---|---|
| `courier` | Indicates literal program code. |
| *italic* | Indicates names of arguments and structure members (in structure/function definitions only). |
| **medium bold** | Indicates data types and structure/function names (in structure/function definitions only). |
| blue | Indicates a hyperlink. |

## Developer Support

### Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

| Order Information | Developer Support |
|---|---|
| *In North America:* | *In North America:* |
| Attn: Developer Tools Coordinator<br>Sony Computer Entertainment America<br>919 East Hillsdale Blvd.<br>Foster City, CA 94404, U.S.A.<br>Tel: (650) 655-8000 | E-mail: PS2_Support@playstation.sony.com<br>Web: http://www.devnet.scea.com/<br>Developer Support Hotline: (650) 655-5566<br>(Call Monday through Friday,<br>8 a.m. to 5 p.m., PST/PDT) |

### Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

| Order Information | Developer Support |
|---|---|
| *In Europe:* | *In Europe:* |
| Attn: Production Coordinator<br>Sony Computer Entertainment Europe<br>30 Golden Square<br>London W1F 9LD, U.K.<br>Tel: +44 (0) 20 7859-5000 | E-mail: ps2_support@scee.net<br>Web: https://www.ps2-pro.com/<br>Developer Support Hotline:<br>+44 (0) 20 7859-5777<br>(Call Monday through Friday,<br>9 a.m. to 6 p.m., GMT) |

## Chapter 1: dev9 Reference (for networks)
## Table of Contents

# Functions

## sceDevctl

Special operations on device

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| edev9 | 2.3 | July 2, 2001 |

### Syntax

**#include <sifdev.h>**

**int sceDevctl(**

| | |
|---|---|
| **const char** **name,* | Device name (dev9x:). |
| **int** *cmd,* | Operation command. Specify one of the following constants.<br>DDIOC_MODEL<br>DDIOC_OFF |
| **void** **arg,* | Argument assigned to command. Depends on cmd. |
| **unsigned int** *arglen,* | arg size |
| **void** **bufp,* | Arguments received from command. Depends on cmd. |
| **unsigned int** *buflen***)** | bufp size |

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function performs special operations on a device. For descriptions of each *cmd*, see the "devctl Command List." The device name is dev9x, not dev9. Be careful not to use the wrong name.

Example: sceDevctl ("dev9x:",DDIOC_OFF, NULL, 0, NULL, 0);

### Return value

When processing succeeds, a cmd-dependent value is returned.

When an error occurs, -1 times the errno is returned.

Errors that are common to each command are as follows:

| | |
|---|---|
| EMFILE | The maximum number of descriptors that can be opened was reached. |
| ENODEV | The specified device does not exist. |

# devctl Commands

## DDIOC_MODEL

Flush disk cache

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| edev9 | 2.3 | July 2, 2001 |

### Syntax

| | |
|---|---|
| *arg* | Reserved. Specify NULL. |
| *arglen* | arg size. |
| *bufp* | Reserved. Specify NULL. |
| *buflen* | bufp size. |

### Description

This command determines whether the device is a PC Card type or EXPANSION BAY type device. Normally, the application need not perform this operation.

### Return value

When the device is a PC Card type device, 0 is returned. When the device is a hard disk drive (EXPANSION BAY type), 1 is returned.

## DDIOC_OFF

Power off device

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| edev9 | 2.3 | July 2, 2001 |

### Syntax

| | |
|---|---|
| *arg* | Reserved. Specify NULL. |
| *arglen* | arg size. |
| *bufp* | Reserved. Specify NULL. |
| *buflen* | bufp size. |

### Description

This command powers off the entire dev9 device.

When powering off the system unit, this processing should be performed only if an HDD Ethernet is used. For more information, see "Power Off Processing" in the Network Library Overview (inet). If a hard disk drive s used, the HDIOC_DEV9OFF command of the hard disk library should be used.

### Return value

When processing succeeds, 0 is returned.

## Chapter 2: Network Socket Library
## Table of Contents

# Structures

## sceInsockHostent_t

Internet host structure

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Structure

```
typedef struct sceInsockHostent {
    char *h_name;                    Host name
    char **h_aliases;                Alias (not supported by this library)
    int h_addrtype;                  Address type (AF_INET)
    int h_length;                    Address size (4 bytes)
    char **h_addr_list;              IP address list (this library supports only one address)
#define h_addr h_addr_list[0]
 } sceInsockHostent_t;
#define hostent sceInsockHostent
```

### Description

This structure represents a host on the Internet.

### See also

sceInsockGethostbyaddr(), sceInsockGethostbyname()

## sceInsockInAddr_t

IPv4 address structure

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Structure

```
typedef struct sceInsockInAddr {
    u_int s_addr;                    IPv4 address (4 bytes)
} sceInsockInAddr_t;
#define in_addr sceInsockInAddr
```

### Description

This structure is used for saving an IPv4 address.

### See also

sceInsockSockaddrIn_t, sceInsockInetAton(), sceInsockInetLnaof(), sceInsockInetNetof(), sceInsockInetNtoa()

## sceInsockSockaddr_t

Socket address structure

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Structure

```
typedef u_char sceInsockSaFamily_t;
typedef struct sceInsockSockaddr {
    u_char sa_len;                  Address structure size
    sceInsockSaFamily_t sa_family;  Address family
    char sa_data[14];               Protocol-dependent address
} sceInsockSockaddr_t;
#define sa_family_t sceInsockSaFamily_t
#define sockaddr sceInsockSockaddr
```

### Description

This structure is used to pass a reference of the socket address structure for each protocol family (currently, only the Internet Protocol).

### See also

sceInsockAccept(), sceInsockBind(), sceInsockConnect(), sceInsockGetpeername(), sceInsockGetsockname(), sceInsockRecvfrom(), sceInsockSendto()

## sceInsockSockaddrIn_t

Internet socket address structure

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Structure

**typedef struct sceInsockSockaddrIn {**

| | |
|---|---|
| **u_char** *sin_len;* | Address structure size (16 bytes) |
| **u_char** *sin_family;* | Address family (AF_INET only) |
| **u_short** *sin_port;* | TCP or UDP port number (network byte order) |
| **sceInsockInAddr_t** *sin_addr;* | IPv4 address |
| **char** *sin_zero***[8];** | Unused |

**} sceInsockSockaddrIn_t;**

**#define sockaddr_in sceInsockSockaddrIn**

### Description

This structure is used to specify the socket for a socket API function.

### See also

sceInsockAccept(), sceInsockBind(), sceInsockConnect(), sceInsockGetpeername(), sceInsockGetsockname(), sceInsockRecvfrom(), sceInsockSendto()

# BSD Socket API-compatible Functions

## sceInsockAccept

Get socket for which TCP connection was established

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**typedef u_int sceInsockSocklen_t;**

**int sceInsockAccept(**

| | |
|---|---|
| **int** *s,* | Listening socket (sceInsockBind() and sceInsockListen() completed) |
| **sceInsockSockaddr_t ***_addr,_ | Pointer to area for storing connection destination address structure |
| **sceInsockSocklen_t** **paddrlen***)** | Pointer to area for storing size of addr |

**#define accept sceInsockAccept**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

When operating as a TCP server, this function gets the connection from the client and returns its socket descriptor. Concurrently, the function sets the client's address structure in the addr argument, and returns its size (always 4 bytes) in paddrlen.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

| | |
|---|---|
| New client socket descriptor | Normal termination |
| -1 | Error |

### See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t, sceInsockErrno

## sceInsockBind

Bind address to socket

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | November 5, 2001 |

### Syntax

```
#include < libinsck.h >
typedef u_int sceInsockSocklen_t;
int sceInsockBind(
 int s,                                  Descriptor of socket to which local address is to be bound
 const sceInsockSockaddr_t *addr,        Pointer to local address structure
 sceInsockSocklen_t addrlen);            Local address structure size (always 16 bytes)
#define bind sceInsockBind
#define socklen_t sceInsockSocklen_t
```

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function binds the local address (IP address and port number) indicated by (addr, addrlen) to the socket s. If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

0 Normal termination

-1 Error

### See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t, sceInsockErrno

## sceInsockConnect

Connect to server

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | November 5, 2001 |

### Syntax

```
#include < libinsck.h >
typedef u_int sceInsockSocklen_t;
int sceInsockConnect(
 int s,                                  Descriptor of socket to be used for connection
 const sceInsockSockaddr_t *addr,        Pointer to local address structure
 sceInsockSocklen_t addrlen);            Local address structure size (always 16 bytes)
#define connect sceInsockConnect
#define socklen_t sceInsockSocklen_t
```

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function uses socket s to connect to the address indicated by (addr, addrlen). For TCP, the connection is established. For UDP, the socket behaves as if the connection were established.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

0 Normal termination

-1 Error

### See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t, sceInsockErrno

## sceInsockErrno

Get socket function error value

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockErrno;**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function returns the error code of socket functions (sceInsockAccept(), sceInsockBind(), sceInsockConnect(), sceInsockListen(), sceInsockRecv(), sceInsockRecvfrom(), sceInsockSend(), sceInsockSendto(), sceInsockShutdown(), sceInsockSocket()).

### Return value

Error code

### See also

sceInsockAccept(), sceInsockBind(), sceInsockConnect(), sceInsockListen(), sceInsockRecv(), sceInsockRecvfrom(), sceInsockSend(), sceInsockSendto(), sceInsockShutdown(), sceInsockSocket()

## sceInsockGethostbyaddr

Get host structure from 32-bit IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**sceInsockHostent_t *sceInsockGethostbyaddr(**

| | |
|---|---|
| **const char** **addr,* | Pointer to 32-bit IPv4 address value |
| **int** *len,* | Address structure size (4 bytes) |
| **int** *type***);** | Address family (AF_INET only) |

**#define gethostbyaddr sceInsockGethostbyaddr**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function gets the Internet host structure corresponding to the 32-bit IPv4 address that was specified by the argument and returns a pointer to it. len is always 4 bytes, and type is always AF_INET.

If an error occurs, details of the error can be found with sceInsockHErrno.

### Return value

| | |
|---|---|
| Pointer to Internet host structure | Normal termination |
| 0 | Error |

### See also

sceInsockHErrno

## sceInsockGethostbyname

Get host structure from hostname

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**sceInsockHostent_t *sceInsockGethostbyname(**

**const char** **name*); Internet host name

**#define gethostbyname sceInsockGethostbyname**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function gets the Internet host structure corresponding to the hostname specified in the name argument and returns a pointer to it.

If an error occurs, details of the error can be found with sceInsockHErrno.

### Return value

| | |
|---|---|
| Pointer to Internet host structure | Normal termination |
| 0 | Error |

### See also

sceInsockHErrno

## sceInsockGetpeername

Get socket connection destination information

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**typedef u_int sceInsockSocklen_t;**

**int sceInsockGetpeername(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket for which information is to be obtained |
| **sceInsockSockaddr_t** **addr,* | Pointer to area for storing address structure of connection destination host |
| **sceInsockSocklen_t** **paddrlen***);** | Pointer to area for storing size of addr (size is always 16 bytes) |

**#define getpeername sceInsockGetpeername**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function stores the address structure of the connection destination host of socket *s* in the area specified by (*addr, paddrlen*).

### Return value

0 Normal termination

-1 Error

### See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t

## sceInsockGetSockName

Get local information of socket

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockGetsockname(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket for which information is to be obtained |
| **sceInsockSockaddr_t** **addr,* | Pointer to area for storing local address structure of socket |
| **sceInsockSocklen_t** **paddrlen***);** | Pointer to area for storing size of local address structure of socket (size is always 16 bytes) |

**#define getsockname sceInsockGetsockname**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function stores the local address structure of socket *s* in the area specified by (*addr, paddrlen*).

### Return value

0 Normal termination

-1 Error

### See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t

## sceInsockGetsockopt

Get socket option

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**typedef u_int sceInsockSocklen_t;**

**int sceInsockGetsockopt(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket for which socket option is to be obtained |
| **int** *level,* | Socket option level |
| **int** *optname,* | Socket option name |
| **void** **optval,* | Pointer to area for storing socket option value |
| **sceInsockSocklen_t *** *optlen***);** | Pointer to area for storing size of socket option value |

**#define getsockopt sceInsockGetsockopt**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function stores the socket option (level: level, option name: optname) of socket *s* in the area specified by (*optval, optlen*). Currently the supported socket options are as follows.

**Table 2-1**

| Socket Option Level | Meaning |
|---|---|
| IPPROTO_TCP | TCP related |

**Table 2-2**

| Socket Option Name | Meaning |
|---|---|
| TCP_NODELAY | Sets Nagle algorithm ON or OFF (1 means OFF and 0 means ON) |

### Return value

0 Normal termination

-1 Error

### See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t

## sceInsockHErrno

Get error value of host structure function

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockHErrno;**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function returns the error code of a host structure function (sceInsockGethostbyaddr() or sceInsockGethostbyname()).

### Return value

**Table 2-3**

| Error Code | Value | Meaning |
|---|---|---|
| NETDB_SUCCESS | 0 | Normal termination |
| NETDB_INTERNAL | -1 | Internal error |
| HOST_NOT_FOUND | 1 | Target host not found |
| TRY_AGAIN | 2 | Temporary error |
| NO_RECOVERY | 3 | Error due to illegal reply from server |
| NO_DATA NO_ADDRESS | 4 | Reply is valid, but IP address is not registered |

### See also

sceInsockGethostbyaddr(), sceInsockGethostbyname()

## sceInsockInetAddr

Get 32-bit address from dot-format IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**u_int sceInsockInetAddr(**

**const char** **cp*); Pointer to dot-decimal IPv4 address string

**#define inet_addr sceInsockInetAddr**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function takes the dot-decimal notation IPv4 address string in the argument and returns the value obtained by converting it to a 32-bit IPv4 address (network byte order).

### Return value

| | |
|---|---|
| 32-bit IPv4 address value (network byte order) | Normal termination |
| INADDR_NONE (0xffffffff) | String is illegal |

## sceInsockInetAton

Get 32-bit address from dot-format IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockInetAton(**

| | |
|---|---|
| **const char** **cp,* | Pointer to dot-decimal IPv4 address string |
| **sceInsockInAddr_t** **addr*); | Pointer to area for storing converted 32-bit IPv4 address value |

**#define inet_aton sceInsockInetAton**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function takes the dot-decimal notation IPv4 address string in the argument and returns the value obtained by converting it to a 32-bit IPv4 address (network byte order). The converted value is stored in the area indicated by *addr*.

### Return value

1 Normal termination

0 String is illegal

### See also

sceInsockInAddr_t

## sceInsockInetLnaof

Get local network address from IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**u_int sceInsockInetLnaof(**

**sceInsockInAddr_t** *in***);** 32-bit IPv4 address value

**#define inet_lnaof sceInsockInetLnaof**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function takes the 32-bit IPv4 address value in the argument and returns only the local network address part.

### Return value

Local network address value

### See also

sceInsockInAddr_t

## sceInsockInetMakeaddr

Construct IPv4 address from network address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**sceInsockInAddr_t sceInsockInetMakeaddr(**

| | |
|---|---|
| **u_int** *net,* | Network address part |
| **u_int** *host***);** | Local network address part |

**#define inet_makeaddr sceInsockInetMakeaddr**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function combines the network address and local network address that were indicated by the arguments to construct one IPv4 address and returns that IPv4 address.

### Return value

Combined IPv4 address value

### See also

sceInsockInAddr_t

## sceInsockInetNetof

Get network address from IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**u_int sceInsockInetNetof(**

**sceInsockInAddr_t** *in***);** 32-bit IPv4 address value

**#define inet_netof sceInsockInetNetof**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function takes the 32-bit IPv4 address value in the argument and returns only the network address part.

### Return value

Network address value

### See also

sceInsockInAddr_t

## sceInsockInetNetwork

Get 32-bit address from dot-format IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**u_int sceInsockInetNetwork(**

**const char ***_cp_**);** Pointer to dot-decimal IPv4 address string

**#define inet_network sceInsockInetNetwork**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function takes the dot-decimal notation IPv4 address string in the argument and returns the value obtained by converting it to a 32-bit IPv4 address (network byte order).

### Return value

| | |
|---|---|
| 32-bit IPv4 address value (network byte order) | Normal termination |
| INADDR_NONE (0xffffffff) | String is illegal |

## sceInsockInetNtoa

Get dot-format address from 32-bit IPv4 address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

```
#include < libinsck.h >
char *sceInsockInetNtoa(
 sceInsockInAddr_t in);                    32-bit IPv4 address value
#define inet_ntoa sceInsockInetNtoa
```

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function takes the 32-bit IPv4 address (network byte order) in the argument, converts it to a dot-decimal notation IPv4 address string, and returns a pointer to that string.

### Return value

Pointer to dot-decimal IPv4 address string

### See also

sceInsockInAddr_t

## sceInsockListen

Accept TCP connection

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockListen(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket for which the TCP connection wait will be performed |
| **int** *backlog***);** | Size of queue for accepting connections (number of pending connections) |

**#define listen sceInsockListen**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function is used to declare that socket *s* is to wait for a TCP connection (i.e. behave as a server).

*backlog* indicates the maximum size of the queue for accepting connections.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

0 Normal termination

-1 Error

### See also

sceInsockErrno

## sceInsockRecv

Receive data

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**size_t sceInsockRecv(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket that is to receive data |
| **void** **buf,* | Pointer to area for storing receive data |
| **size_t** *len,* | Data size to be received (in bytes) |
| **int** *flags***);** | Not supported (must be set to 0) |

**#define recv sceInsockRecv**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function receives *len* bytes of data from socket *s*. The receive data is stored in the area specified by *buf*.

Since the *flags* argument is not supported, it must always be set to 0.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

| | |
|---|---|
| Positive number | Size of received data (in bytes) |
| -1 | Error |

### See also

sceInsockErrno

## sceInsockRecvfrom

Receive data (also get address structure of sending host)

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**typedef u_int sceInsockSocklen_t;**

**size_t sceInsockRecvfrom(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket that is to receive data |
| **void** **buf,* | Pointer to area for storing receive data |
| **size_t** *len,* | Data size to be received (in bytes) |
| **int** *flags,* | Not supported (must be set to 0) |
| **sceInsockSockaddr_t** **addr,* | Pointer to area for storing address structure of sending host |
| **sceInsockSocklen_t** **paddrlen***);** | Pointer to area for storing size of address structure of sending host (size is always 16 bytes) |

**#define recvfrom sceInsockRecvfrom**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function receives *len* bytes of data from socket *s*. The receive data is stored in the area specified by *buf*.

Since the *flags* argument is not supported, it must be set to 0. The area for storing the address structure is specified by (*addr, paddrlen*), and the address structure of the sending host is stored in that area when data is received.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

| | |
|---|---|
| Positive number | Size of received data (in bytes) |
| -1 | Error |

### See also

sceInsockSockaddr_t, sceInsockErrno

## sceInsockSend

Send data

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | November 5, 2001 |

### Syntax

```
#include < libinsck.h >
size_t sceInsockSend(
 int s,                          Descriptor of socket that is to send data
 const void *buf,                Pointer to send data
 size_t len,                     Size of data to be sent (in bytes)
 int flags);                     Not supported (must be set to 0)
#define send sceInsockSend
```

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function sends *len* bytes of data from socket *s*. The data to send is specified by *buf*.

Since the *flags* argument is not supported, if must be set to 0.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

| | |
|---|---|
| Positive number | Size of transmitted data (in bytes) |
| -1 | Error |

### See also

sceInsockErrno

## sceInsockSendto

Send data (specify address structure of receiving host)

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | November 5, 2001 |

### Syntax

**#include < libinsck.h >**

**typedef u_int sceInsockSocklen_t;**

**size_t sceInsockSendto(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket that is to send data |
| **const void** **buf,* | Pointer to send data |
| **size_t** *len,* | Size of data to be sent (in bytes) |
| **int** *flags,* | Not supported (must be set to 0) |
| **const sceInsockSockaddr_t** **addr,* | Pointer to address structure of receiving host |
| **sceInsockSocklen_t** *addrlen***);** | Size of address structure of receiving host (always 16 bytes) |

**#define sendto sceInsockSendto**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function sends *len* bytes of data from socket *s*. The data to send is specified by *buf,* and the address structure of the receiving host is specified by (*addr, addrlen*). Since the *flags* argument is not supported, it must be set to 0.

If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

| | |
|---|---|
| Positive number | Size of transmitted data (in bytes) |
| -1 | Error |

### See also

sceInsockSockaddr_t, sceInsockErrno

## sceInsockSetsockopt

Set socket option

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | November 5, 2001 |

### Syntax

**#include < libinsck.h >**

**typedef u_int sceInsockSocklen_t;**

**int sceInsockSetsockopt(**

| | |
|---|---|
| **int** *s,* | Descriptor of socket for which socket option is to be obtained |
| **int** *level,* | Socket option level |
| **int** *optname,* | Socket option name |
| **const void** **optval,* | Pointer to area for storing socket option value |
| **sceInsocksocklen_t** *optlen***);** | Size of socket option value |

**#define setsockopt sceInsockSetsockopt**

**#define socklen_t sceInsockSocklen_t**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function sets the socket option (level: level, option name: optname) of socket *s* for the value specified by (optval, optlen). Currently the supported socket options are as follows.

**Table 2-4**

| Socket Option Level | Meaning |
|---|---|
| IPPROTO_TCP | TCP related |

**Table 2-5**

| Socket Option Name | Meaning |
|---|---|
| TCP_NODELAY | Sets Nagle algorithm ON or OFF (1 means OFF and 0 means ON) |

### Return value

0 Normal termination

-1 Error

## sceInsockShutdown

Close socket

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

```
#include < libinsck.h >
int sceInsockShutdown(
 int s,                          Descriptor of socket to be closed
 int how);                       Shutdown method (not supported)
#define shutdown sceInsockShutdown
```

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function closes socket *s*. Since specifying a shutdown method with the argument *how* is not supported (i.e. half close cannot be performed), the argument *how* must be set to 0. If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

0 Normal termination

-1 Error

### See also

sceInsockErrno

## sceInsockSocket

Create socket

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.3 | July 2, 2001 |

### Syntax

**#include < libinsck.h >**

**size_t sceInsockSocket(**

| | | | |
|---|---|---|---|
| **int** *family,* | Address family of socket to be created (AF_INET only) | | |
| **int** *type,* | Socket type (any of the following) | | |
| | SOCK_STREAM | 1 | TCP socket |
| | SOCK_DGRAM | 2 | UDP socket |
| | SOCK_RAW | 3 | raw socket |
| **int** *protocol***);** | Protocol (not supported, must be set to 0) | | |

**#define socket sceInsockSocket**

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function creates a socket having the address family indicated by the family argument (always AF_INET) and the socket type indicated by the *type* argument. It returns the descriptor for that socket. If an error occurs, details of the error can be found with sceInsockErrno.

### Return value

| | |
|---|---|
| Positive value | Descriptor of generated socket |
| -1 | Error |

### See also

sceInsockErrno

# Other Functions

## sceInsockAbort

Abort processing

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.4.1 | November 5, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockAbort(**

| | |
|---|---|
| **int** *s*, | Socket descriptor |
| **int** *flags*); | Flags |

### Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

### Description

This function calls sceInetAbort() for the specified socket (s). The flags argument is provided for future expansion. Zero should always be specified for this argument.

### Return value

0 Normal termination

-1 Error

## sceInsockSetRecvTimeout

Set receive timeout

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.4.1 | November 5, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockSetRecvTimeout(**

| | |
|---|---|
| **int** *s*, | Socket descriptor |
| **int** *ms*); | Timeout interval |

### Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

### Description

This function sets the timeout interval for sceInsockRecv() and sceInsockRecvFrom(). The timeout interval is specified in milliseconds (ms).

If this function in not called, the default value for the timeout interval is -1 (unlimited).

### Return value

0 Normal termination

-1 Error

## sceInsockSetSendTimeout

Set send timeout

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.4.1 | November 5, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockSetSendTimeout(**

| | |
|---|---|
| **int** *s***,** | Socket descriptor |
| **int** *ms***);** | Timeout interval |

### Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

### Description

This function sets the timeout interval for sceInsockSend() and sceInsockSendTo(). The timeout interval is specified in milliseconds (ms). If this function in not called, the default value for the timeout interval is -1 (unlimited).

### Return value

0 Normal termination

-1 Error

## sceInsockSetSifMBindRpcValue

Set buffer size, stack size and priority

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.4 | October 11, 2001 |

### Syntax

**#include < libinsck.h>**

**int sceInsockSetSifMBindRpcValue(**

| | |
|---|---|
| **u_int** *buffersize***,** | Size of the receive buffer for capturing send data from SceSifMCallRpc().<br>The buffersize is normally 2048 bytes. |
| **u_int** *stacksize***,** | Stack size for IOP threads that perform SceSifMCallRpc() requests. The minimum size is 512 bytes.<br>The stacksize is normally 8192 bytes. |
| **int** *priority***)** | Priority for IOP threads that perform SceSifMCallRpc() requests. Since the system uses values of 10 or less, a greater value should be specified.<br>The priority is normally 32. |

### Calling conditions

Can be called from a thread

Multithread safe (must be called in interrupt-enabled state)

### Description

This function sets the buffer size, stack size and priority to be used when the libnet_init() function of libnet is called from libinsck. If this function is not called, a buffer size of 2048, stack size of 8192, and priority of 32 are assumed to have been specified.

The settings performed by this function are recorded for each thread and do not affect other threads. If this function is called more than once from the same thread, only the last setting will be valid.

### Return value

0 Normal termination

-1 Error

## sceInsockTerminate

Free memory area

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| libinsck | 2.4.1 | November 5, 2001 |

### Syntax

**#include < libinsck.h >**

**int sceInsockTerminate(**

**int** *thread_id***);** Thread ID

### Calling conditions

Can be called from a thread

Multithread safe (must be called in an interrupt-enabled state)

### Description

This function frees the memory area of each thread that was automatically allocated by libinsck.

thread_id specifies the thread ID for which the memory area is to be freed. A thread_id of 0 means the calling thread.

When the socket() function is called, memory is allocated as necessary. That memory is not automatically freed by the shutdown() function. This function should be explicitly called to free that memory.

### Return value

0 Normal termination

-1 Error

## Chapter 3: Network Configuration GUI Library
## Table of Contents

# Structures

## sceNetGuiCnf_Arg

Argument data for sceNetGuiCnf_Do()

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

**Structure**

**typedef struct sceNetGuiCnf_Arg {**

| | |
|---|---|
| **int** *flag*; | Startup options |
| **int** *_sema_vsync*; | Semaphore waiting for start of v-blank |
| **sceNetGuiCnfEnvData_t** *****_default_env_data_; | Pointer to default data to be used when adding |
| **sceNetGuiCnfEnvData_t** *****_result_env_data_; | Pointer to buffer for returning selection result |
| **sceNetGuiCnfCallback_Malloc** *cb_malloc*; | Pointer to malloc function |
| **sceNetGuiCnfCallback_Memalign** *cb_memalign*; | Pointer to memalign function |
| **sceNetGuiCnfCallback_Realloc** *cb_realloc*; | Pointer to realloc function |
| **sceNetGuiCnfCallback_Free** *cb_free*; | Pointer to free function |
| **sceNetGuiCnfCallback_SKBInit** *cb_skb_init*; | Pointer to software keyboard initialization function |
| **sceNetGuiCnfCallback_SKBDestroy** *cb_skb_destroy*; | Pointer to software keyboard termination processing function |
| **sceNetGuiCnfCallback_SKBGetVif1PktTopAddr** *cb_skb_getvif1pkttopaddr*; | Pointer to function for getting drawing packet address of software keyboard |
| **sceNetGuiCnfCallback_SKBGetStatus** *cb_skb_getstatus*; | Pointer to function for getting size of software keyboard |
| **sceNetGuiCnfCallback_SKBSendMouseMessage** *cb_skb_sendmousemessage*; | Pointer to function for sending mouse pointer message to software keyboard |
| **sceNetGuiCnfCallback_SKBEnableKey** *cb_skb_enablekey*; | Pointer to function for setting key state of software keyboard |
| **sceNetGuiCnfCallback_SKBEveryFrame** *cb_skb_everyframe*; | Pointer to function for processing software keyboard every frame |
| **sceNetGuiCnfCallback_SJIStoUTF8** *cb_sjis_to_utf8*; | Pointer to function for converting character code from SJIS to UTF8 |
| **sceNetGuiCnfCallback_UTF8toSJIS** *cb_utf8_to_sjis*; | Pointer to function for converting character code from UTF8 to SJIS |
| **sceNetGuiCnfCallback_UsbMouseRead** *cb_mouse_read*; | Pointer to function for receiving USB mouse input |
| **sceNetGuiCnfCallback_PadRead** *cb_pad_read*; | Pointer to function for receiving button state |

| | |
|---|---|
| **sceNetGuiCnfCallback_UsbKbRead** *cb_kb_read*; | Pointer to function for receiving USB keyboard input |
| **char** *****_str_path_bg_; | Pointer to string indicating background file path |
| **sceNetGuiCnf_Color4_t** *color_titlebar*; | Color of title bar that is always displayed at top of screen |
| **sceNetGuiCnf_Color4_t** *color_window*; | Background color of window that is always displayed in center of screen |
| **sceNetGuiCnf_Color4_t** *color_pagebutton*; | Color of Quit, Back, and Next buttons that are always displayed at bottom of screen |
| **sceNetGuiCnf_Color4_t** *color_msgbox_ok*; | Color of title bar of one-choice message box (*In the current version, this is the same as the color of the title bar of an error message box) |
| **sceNetGuiCnf_Color4_t** *color_msgbox_yesno*; | Color of title bar of two-choice message box |
| **sceNetGuiCnf_Color4_t** *color_msgbox_warning*; | Color of title bar of error message box (*Not used in the current version) |
| **sceNetGuiCnf_Color4_t** *color_msgbox_wait*; | Color of title bar of non-selectable message box |

**} sceNetGuiCnf_Arg_t;**

## Description

This structure is used to set argument data for the sceNetGuiCnf_Do function. Appropriate values and function pointers (non-NULL) must be set for all members when sceNetGuiCnf_Do() is used.

The flag value is the logical OR of the following bits.

**Table 3-1**

| Constant | Bit | Meaning |
|---|---|---|
| SCE_NETGUICNF_FLAG_USE_HDD | 0 | Use hard disk drive |
| SCE_NETGUICNF_FLAG_USE_USB_MOUSE | 1 | Use USB mouse |
| SCE_NETGUICNF_FLAG_USE_USB_KB | 2 | Use USB keyboard |
| SCE_NETGUICNF_FLAG_USE_SELECT_OPTION | 3 | Enable bit 3 startup option |
| SCE_NETGUICNF_FLAG_SELECT_ONLY | 4 | 0: Skip configuration selection<br>1: Only select configuration |
| SCE_NETGUICNF_FLAG_MC_SLOT1_ONLY | 5 | Use memory card slot 1 only |

The value of *_sema_vsync* will be the return value from the EE kernel's CreateSema function. If no default data is set during an add, the value of *default_env_data* will be NULL. When the SCE_NETGUICNF_FLAG_USE_USB_MOUSE bit is set to 0, *cb_mouse_read* will be ignored even if it has been set with a function pointer. In this case, *cb_mouse_read* can also be set to NULL. Similarly, when the SCE_NETGUICNF_FLAG_USE_USB_KB bit is set to 0, *cb_kb_read* will be ignored even if has been set with a function pointer. In this case, *cb_kb_read* can also be set to NULL.

## See also

sceNetGuiCnfEnvData, sceNetGuiCnf_Color4, sceNetGuiCnf_Do

## sceNetGuiCnf_Color

Color data for one vertex of sceNetGuiCnf_Color4 structure

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Structure

```
typedef struct sceNetGuiCnf_Color {
    unsigned char r;        Red component (0 to 255)
    unsigned char g;        Green component (0 to 255)
    unsigned char b;        Blue component (0 to 255)
    unsigned char a;        Alpha value (128 is primary color)
} sceNetGuiCnf_Color_t;
```

### Description

This structure represents color data for one vertex in the sceNetGuiCnf_Color4 structure.

### See also

sceNetGuiCnf_Color4

## sceNetGuiCnf_Color4

Color specification structure

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Structure

**typedef struct sceNetGuiCnf_Color4 {**

**sceNetGuiCnf_Color_t** *aColor***[4];**

| | |
|---|---|
| aColor[0] | Upper left vertex color data |
| aColor[1] | Upper right vertex color data |
| aColor[2] | Lower left vertex color data |
| aColor[3] | Lower right vertex color data |

**} sceNetGuiCnf_Color_t;**

### Description

This structure allows colors to be specified for UI elements by setting the following members in the sceNetGuiCnf_Arg structure.

**Table 3-2**

| Member | Description |
|---|---|
| *color_titlebar* | Color of title bar that is always displayed at top of screen |
| *color_window* | Background color of window that is always displayed in center of screen |
| *color_pagebutton* | Color of Quit, Back, and Next buttons that are always displayed at bottom of screen |
| *color_msgbox_ok* | Color of title bar of one-choice message box<br>(*In the current version, this is the same as the color of the title bar of an error message box) |
| *color_msgbox_yesno* | Color of title bar of two-choice message box |
| *color_msgbox_warning* | Color of title bar of error message box<br>(*Not used in the current version) |
| *color_msgbox_wait* | Color of title bar of non-selectable message box |

### See also

sceNetGuiCnf_Arg

## sceNetGuiCnfEnvData

Network configuration data

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

**Structure**

**typedef struct sceNetGuiCnfEnvData {**

| | |
|---|---|
| **char** *attach_ifc***[256];** | Network service provider setting filename that is registered in a combination (used only by sceNetGuiCnf_Do()) |
| **char** *attach_dev***[256];** | Hardware setting filename that is registered in a combination (used only by sceNetGuiCnf_Do()) |
| **char** *address***[256];** | IP address |
| **char** *netmask***[256];** | Netmask |
| **char** *gateway***[256];** | Default router |
| **char** *dns1_address***[256];** | Primary DNS |
| **char** *dns2_address***[256];** | Secondary DNS |
| **char** *phone_numbers1***[256];** | Tel. Number1 |
| **char** *phone_numbers2***[256];** | Tel. Number2 |
| **char** *phone_numbers3***[256];** | Tel. Number3 |
| **char** *auth_name***[256];** | User ID |
| **char** *auth_key***[256];** | Password |
| **char** *vendor***[256];** | Vendor name |
| **char** *product***[256];** | Product name |
| **char** *chat_additional***[256];** | Additional AT command |
| **char** *outside_number***[256];** | Outside number setting |
| **char** *outside_delay***[256];** | Keyword for specifying outside number origination delay string (character string following numeric string in outside number setting) |
| **char** *dhcp_host_name***[256];** | DHCP host name |
| **int** *dialing_type***;** | Dialing type |
| **int** *type***;** | Device layer type |
| **int** *phy_config***;** | Ethernet hardware operating mode |
| **int** *idle_timeout***;** | Line timeout (minutes) |
| **unsigned char** *dhcp***;** | DHCP used/unused setting |
| **unsigned char** *dns1_nego***;** | Sets negotiation related to primary DNS |
| **unsigned char** *dns2_nego***;** | Sets negotiation related to secondary DNS |
| **unsigned char** *f_auth***;** | Enables/disables setting of authorization method allowed on local side |
| **unsigned char** *auth***;** | Authorization method allowed on local side |
| **unsigned char** *pppoe***;** | PPPoE (PPP over Ethernet) used/unused setting |
| **unsigned char** *prc_nego***;** | PRC (Protocol-Field-Compression) negotiation setting |
| **unsigned char** *acc_nego***;** | ACC (Address-and-Control-Field-Compression) negotiation setting |

| | |
|---|---|
| **unsigned char** *accm_nego*; | ACCM (Async-Control-Character-Map) negotiation setting |
| **unsigned char** *p0*; | Reserved area 0 (always 0) |
| **unsigned char** *p1*; | Reserved area 1 (always 0) |
| **unsigned char** *p2*; | Reserved area 2 (always 0) |
| **int** *mtu*; | MTU value |

**} sceNetGuiCnfEnvData_t;**

## Description

This structure is used to send default data when doing an add in the library and for receiving the selected network configuration from the library. To set default values, all of the following members must be set. Members other than those listed below are ignored.

**Table 3-3**

| Member | Description |
|---|---|
| *address* | IP address |
| *netmask* | Netmask |
| *gateway* | Default router |
| *dns1_address* | Primary DNS |
| *dns2_address* | Secondary DNS |
| *phone_numbers1* | Tel. Number1 |
| *phone_numbers2* | Tel. Number2 |
| *phone_numbers3* | Tel. Number3 |
| *auth_name* | User ID |
| *auth_key* | Password |
| *chat_additional* | Additional AT command |
| *outside_number* | Outside number setting |
| *outside_delay* | Keyword for specifying outside number origination delay string (character string following numeric string in outside number setting) |
| *dhcp_host_name* | DHCP host name |
| *dialing_type* | Dialing type |
| *idle_timeout* | Line timeout (minutes) |
| *phy_config* | Ethernet hardware operating mode |
| *dhcp* | DHCP used/unused setting |
| *pppoe* | PPPoE (PPP over Ethernet) used/unused setting |

For details about the values that can be set for each member, refer to the “Guidelines for Creating a Network Configuration Application” document. To not configure a string-format member, set '\0' at the str[0] position.

*dialing_type* can be any of the following values.

**Table 3-4**

| Constant | Value | Meaning |
|---|---|---|
| | -1 | No setting |
| SCE_NETGUICNF_DIALINGTYPE_TONE | 0 | Tone |
| SCE_NETGUICNF_DIALINGTYPE_PULSE | 1 | Pulse |

*phy_config* can be any of the following values.

**Table 3-5**

| Constant | Value | Meaning |
|---|---|---|
| | -1 | No setting |
| SCE_NETGUICNF_PHYCONFIG_AUTO | 1 | Auto Negotiation Mode |
| SCE_NETGUICNF_PHYCONFIG_10 | 2 | 10BaseT, Half-Duplex |
| SCE_NETGUICNF_PHYCONFIG_10_FD | 3 | 10BaseT, Full-Duplex, No-Flow-Control |
| SCE_NETGUICNF_PHYCONFIG_TX | 5 | 100BaseTX, Half-Duplex |
| SCE_NETGUICNF_PHYCONFIG_TX_FD | 6 | 100BaseTX, Full-Duplex, No-Flow-Control |

*dhcp* can be either of the following values.

**Table 3-6**

| Constant | Value | Meaning |
|---|---|---|
| SCE_NETGUICNF_NOUSE_DHCP | 0 | DHCP is used |
| SCE_NETGUICNF_USE_DHCP | 1 | DHCP is not used |

*pppoe* can be any of the following values.

**Table 3-7**

| Constant | Value | Meaning |
|---|---|---|
| | -1 | No setting |
| SCE_NETGUICNF_NOUSE_PPPOE | 0 | PPPoE (PPP over Ethernet) is used |
| SCE_NETGUICNF_USE_PPPOE | 1 | PPPoE (PPP over Ethernet) is not used |

*type* can be any of the following values.

**Table 3-8**

| Constant | Value | Meaning |
|---|---|---|
| SCE_NETGUICNF_TYPE_ETH | 1 | USB Ethernet is supported |
| SCE_NETGUICNF_TYPE_PPP | 2 | PPP is supported |
| SCE_NETGUICNF_TYPE_NIC | 3 | Ethernet that uses a network adaptor is supported |

When the selected network configuration is received from the library and set in the common network configuration library, the corresponding member configuration is as follows.

```
{
sceNetCnfEnv_t *e;
sceNetCnfInterface *ifc = e->root->pair_head->ifc;
sceNetCnfInterface *dev = e->root->pair_head->dev;
}
```

The meanings of the various pointers are described above. For coding examples, see /usr/local/sce/iop/sample/inet/ntguicnf/setinit.

**Table 3-9**

| Member | Description |
|---|---|
| *attach_ifc* | Not used |
| *attach_dev* | Not used |
| *address* | ifc->address |
| *netmask* | ifc->netmask |
| *gateway* | struct sceNetCnfRoutingEntry routing placed after ifc->cmd_head |
| *dns1_address* | struct sceNetCnfAddress address placed after ifc->cmd_head |
| *dns2_address* | struct sceNetCnfAddress address placed after ifc->cmd_head |
| *phone_numbers1* | ifc->phone_numbers[0] |
| *phone_numbers2* | ifc->phone_numbers[1] |
| *phone_numbers3* | ifc->phone_numbers[2] |
| *auth_name* | ifc->auth_name |
| *auth_key* | ifc->auth_key |
| *vendor* | dev->vendor |
| *product* | dev->product |
| *chat_additional* | dev->chat_additional |
| *outside_number* | dev->outside_number |
| *outside_delay* | dev->outside_delay |
| *dhcp_host_name* | ifc->dhcp_host_name |
| *dialing_type* | dev->dialing_type |
| *type* | dev->type or ifc->type<br>The value that is always set for dev->type is returned here<br>For PPPoE, the user must intentionally set SCE_NETGUICNF_TYPE_PPP for ifc->type<br>The value of type is set as is for dev->type |
| *phy_config* | dev->phy_config |
| *idle_timeout* | For PPPoE, ifc->idle_timeout<br>Otherwise, dev->idle_timeout |
| *dhcp* | ifc->dhcp |
| *dns1_nego* | ifc->want.dns1_nego |
| *dns2_nego* | ifc->want.dns2_nego |
| *f_auth* | ifc->allow.f_auth |
| *auth* | ifc->allow.auth |
| *pppoe* | If pppoe is 1, ifc->pppoe is set directly with the value of pppoe<br>If pppoe is 0, ifc->pppoe is set to –1 |
| *prc_nego* | ifc->want.prc_nego |
| *acc_nego* | ifc->want.acc_nego |
| *accm_nego* | ifc->want.accm_nego |
| *mtu* | ifc->mtu |

**See also**

sceNetGuiCnf_Arg

# Function Types

## sceNetGuiCnfCallback_Free

free

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <ntguicnf.h>
 typedef void (*sceNetGuiCnfCallback_Free)(
 void * ptr);                                    Area to be freed
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This is a free function that is ANSI-compliant.

### Return value

None

## sceNetGuiCnfCallback_Malloc

malloc

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <ntguicnf.h>**

**typedef void * (* sceNetGuiCnfCallback_Malloc)(**

**size_t** *size***);** Size of area in bytes

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This is a malloc function that is ANSI-compliant.

### Return value

When allocation succeeds, a pointer to the allocated area is returned. When size is 0, NULL is returned. When the area cannot be allocated, NULL is returned.

## sceNetGuiCnfCallback_Memalign

memalign

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <ntguicnf.h>**

**typedef void * (*sceNetGuiCnfCallback_Memalign)(**

| | |
|---|---|
| **size_t** *size***,** | Size of area in bytes |
| **size_t** *align***);** | Alignment (must be a power of 2 and at least 4 bytes) |

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function allocates an area of storage that is a multiple of the specified alignment, exceeding the number of bytes specified by size and starting at an address that is a multiple of the specified alignment. Other allocation actions are the same as those of a malloc function that is ANSI-compliant.

### Return value

When allocation succeeds, a pointer to the allocated area is returned. When size is 0, NULL is returned. When the area cannot be allocated, NULL is returned.

## sceNetGuiCnfCallback_PadRead

Get controller's button information

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <ntguicnf.h>**

**typedef void * (*sceNetGuiCnfCallback_PadRead)(**

**unsigned int *** *paddata***);** State of controller's digital buttons

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function gets the state of the controller's digital buttons. The meaning of each bit is the same as the digital button state that is defined by the scePadRead() function.

### Return value

None

## sceNetGuiCnfCallback_Realloc

realloc

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <ntguicnf.h>
typedef void * (*sceNetGuiCnfCallback_Realloc)(
 void * old_ptr,                    Area to be reallocated
 size_t new_size);                  Size of area in bytes
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This is a realloc function that is ANSI-compliant.

### Return value

When allocation succeeds, a pointer to the allocated area is returned. When size is 0, NULL is returned. When the area cannot be allocated, NULL is returned.

## sceNetGuiCnfCallback_SJIStoUTF8

Convert string from Shift-JIS to UTF8

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <netguicnf.h>
typedef void (*sceNetGuiCnfCallback_SJIStoUTF8)(
 unsigned char * dst,                    Output buffer pointer
 size_t dst_size,                        Output buffer size
 unsigned char const * src);             Input string
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function converts a Shift-JIS string to a UTF8 string.

### Return value

None

## sceNetGuiCnfCallback_SKBDestroy

Software keyboard termination processing

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <ntguicnf.h>
typedef void (*sceNetGuiCnfCallback_SKBDestroy)(
 void);
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function performs software keyboard termination processing.

### Notes

This function is called only once by sceNetGuiCnf_Do().

### Return value

None

## sceNetGuiCnfCallback_SKBEnableKey

Configure software keyboard key states

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <ntguicnf.h>**

**typedef void**
**(*sceNetGuiCnfCallback_SKBEnableKey)(**

| | |
|---|---|
| **int** *type***,** | Configuration type |
| **unsigned char *** *keynames***[ ],** | Key identification character array |
| **int** *keynames_size***);** | Size of key identification character array |

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function enables/disables keys on the software keyboard.

*type* can have any of the following values.

**Table 3-10**

| Constant | Value | Meaning |
|---|---|---|
| SCE_NETGUICNF_ENABLE_KEY_TYPE_ENABLE_LISTED_AND_DISABLE_NOTLISTED | 0 | Enable listed keys and disable other keys |
| SCE_NETGUICNF_ENABLE_KEY_TYPE_ENABLE_ALL | 1 | Enable all keys |
| SCE_NETGUICNF_ENABLE_KEY_TYPE_DISABLE_LISTED | 2 | Disable listed keys (do nothing to other keys) |

### Notes

The following strings can be used for the key identification character array. (Other character keys and control keys cannot be used, even if they exist.)

- BS
- DEL
- LEFT
- RIGHT
- HOME
- END
- Other Shift-JIS characters that can be used are described in the “Guidelines for Creating a Network Configuration Application” document.

### Return value

None

## sceNetGuiCnfCallback_SKBEveryFrame

Software keyboard every frame processing

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <ntguicnf.h>
typedef void (*sceNetGuiCnfCallback_SKBEveryFrame)(
 void);
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function performs every frame processing for the software keyboard.

### Return value

None

## sceNetGuiCnfCallback_SKBGetStatus

Get software keyboard size

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <ntguicnf.h>**

**typedef void (*sceNetGuiCnfCallback_SKBGetStatus)(**

| | |
|---|---|
| **int *** *w***,** | Pointer to variable for returning width (pixels) |
| **int *** *h***);** | Pointer to variable for returning height (pixels) |

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function returns the size of the software keyboard.

### Return value

None

## sceNetGuiCnfCallback_SKBGetVif1PktTopAddr

Get software keyboard drawing packet address

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <ntguicnf.h>
typedef void * (*sceNetGuiCnfCallback_SKBGetVif1PktTopAddr)(
 void);
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function returns the starting address of the drawing packet for displaying the software keyboard.

### Notes

The drawing packet must satisfy the following specifications.

- It must be a drawing packet via PATH2.
- It must end with RET because it is called with a DMA CALL.
- There must be a double buffer.
- The position must be drawn starting at the upper left corner of the screen. (The display position, which is the GS offset, is changed within the sceNetGuiCnf_Do function.)
- The GS offset must not be changed.
- Context 2 must be used.
- It must be a packet that sends the texture every time. (A texture base point of 8960 or later can be used.)
- It must have a resolution of 640x448.

### Return value

Starting address of software keyboard drawing packet.

## sceNetGuiCnfCallback_SKBInit

Initialize software keyboard

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <ntguicnf.h>
typedef void (*sceNetGuiCnfCallback_SKBInit)(
 void);
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function initializes the software keyboard.

### Notes

This function is called only once by sceNetGuiCnf_Do().

### Return value

None

## sceNetGuiCnfCallback_SKBSendMouseMessage

Send mouse pointer message

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <ntguicnf.h>**

**typedef int**
**(*sceNetGuiCnfCallback_SKBSendMouseMessage)(**

| | |
|---|---|
| **int** *type***,** | Activation point of mouse |
| **int** *x***,** | Relative x coordinate with respect to software keyboard display position origin |
| **int** *y***);** | Relative y coordinate with respect to software keyboard display position origin |

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function sends a mouse pointer message to the software keyboard.

*type* can be any of the following values.

**Table 3-11**

| Constant | Value | Meaning |
|---|---|---|
| SCE_NETGUICNF_MOUSE_MESSAGE_TYPE_PRESS | 0 | Pressed |
| SCE_NETGUICNF_MOUSE_MESSAGE_TYPE_RELEASE | 1 | Released |
| SCE_NETGUICNF_MOUSE_MESSAGE_TYPE_MOVE | 2 | Moved |

### Return value

If the mouse cannot be clicked at the position with coordinates (x,y), 0 is returned. If the mouse can be clicked at that position, 1 is returned.

## sceNetGuiCnfCallback_UsbKbRead

Receive USB keyboard input

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <netguicnf.h>
typedef void (*sceNetGuiCnfCallback_UsbKbRead)(
 void);
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function reports USB keyboard input information internally to the network configuration GUI library using the sceNetGuiCnf_SendKBMessage() function.

### Return value

None

## sceNetGuiCnfCallback_UsbMouseRead

Receive USB mouse input

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

**#include <netguicnf.h>**

**typedef void**
**(*sceNetGuiCnfCallback_UsbMouseRead)(**

| | |
|---|---|
| **int *** *delta_x***,** | Amount of movement in x direction |
| **int *** *delta_y***,** | Amount of movement in y direction |
| **int *** *buttons***,** | Button state |
| **int *** *wheel***);** | Amount of wheel movement |

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function returns USB mouse input information to the pointers specified in the arguments. *delta_x* and *delta_y* return positive values for motion down and to the right, and negative values for motion up and to the left. *wheel* returns a negative value for upward rotation and a positive value for downward rotation.

The value of *buttons* will be the logical OR of the following bits.

**Table 3-12**

| Constant | Bit | Meaning |
|---|---|---|
| SCE_NETGUICNF_MOUSE_BUTTON_LEFT | 0 | Left button is pressed |
| SCE_NETGUICNF_MOUSE_BUTTON_RIGHT | 1 | Right button is pressed |
| SCE_NETGUICNF_MOUSE_BUTTON_MIDDLE | 2 | Middle button is pressed |

### Return value

None

## sceNetGuiCnfCallback_UTF8toSJIS

Convert string from UTF8 to Shift-JIS

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <netguicnf.h>
typedef void (*sceNetGuiCnfCallback_ UTF8toSJIS)(
 unsigned char * dst,                    Output buffer pointer
 size_t dst_size,                        Output buffer size
 unsigned char const * src);             Input string
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function converts a UTF8 string to a Shift-JIS string.

### Return value

None

# Functions

## sceNetGuiCnf_Do

Start network configuration application

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <netguicnf.h>
void sceNetGuiCnf_Do(
 sceNetGuiCnf_Arg_t * arg);                    Startup arguments
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function starts up the network configuration application. For the start-up arguments, see the sceNetGuiCnf_Arg structure. The following IOP modules must be loaded before this function is started.

**Required IOP modules**

- sio2man.irx
- padman.irx
- mcman.irx
- mcserv.irx
- netcnf.irx
- inet.irx
- inetctl.irx
- ppp.irx
- pppoe.irx
- usbd.irx
- ntguicnf.irx

IOP module required to autoload USB connection device driver

- usbmload.irx

IOP modules required to use the hard disk drive

- dev9.irx
- atad.irx
- hdd.irx
- pfs.irx
- smap.irx

- sceNetGuiCnf_Do() resets and reconfigures the drawing environment such as the GS. Consequently, after the function completes, the IOP and GS must be reconfigured as necessary. sceNetGuiCnf_Do() invokes the WaitSema function from the end of one frame of work until the start of v-blank. As a result, the SignalSema function must be invoked when v-blank begins. For more information, refer to the Network Configuration GUI Library Overview.

**Return value**

None

## sceNetGuiCnf_SendSoftKBMessage

Send key information to network configuration application

| *Library* | *Introduced* | *Documentation last modified* |
|---|---|---|
| ntguicnf | 2.4 | October 1, 2001 |

### Syntax

```
#include <netguicnf.h>
void sceNetGuiCnf_SendSoftKBMessage(
 int type,                                  Keyboard type
 unsigned char * keyname);                  Key identification characters
```

### Calling conditions

Can be called from a thread

Not multithread safe (must be called in interrupt-enabled state)

### Description

This function reports key information to the network configuration GUI library.

*type* can be any of the following values.

**Table 3-13**

| Constant | Value | Meaning |
|---|---|---|
| SCE_NETGUICNF_KBMSG_TYPE_SOFTKB | 0 | Input from software keyboard |
| SCE_NETGUICNF_KBMSG_TYPE_HARDKB | 1 | Input from USB keyboard |

### Remark

The following strings can be used for the key identification character array. (Other character keys and control keys cannot be used, even if they exist.)

- BS
- DEL
- LEFT
- RIGHT
- HOME
- END
- Other Shift-JIS characters that can be used are described in the “Guidelines for Creating a Network Configuration Application” document.

### Return value

None