PlayStation®2 IOP Library Reference
Release 2.4.2

Network Libraries

© 2001 Sony Computer Entertainment Inc.

Publication date: December 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.

The *PlayStation®2 IOP Library Reference - Network Libraries* manual is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® license agreements.

The *PlayStation®2 IOP Library Reference - Network Libraries* manual is intended for distribution to and use by only Sony Computer Entertainment licensed Developers and Publishers in accordance with the PlayStation® license agreements.

Unauthorized reproduction, distribution, lending, rental or disclosure to any third party, in whole or in part, of this book is expressly prohibited by law and by the terms of the Sony Computer Entertainment PlayStation® license agreements.

Ownership of the physical property of the book is retained by and reserved by Sony Computer Entertainment. Alteration to or deletion, in whole or in part, of the book, its presentation, or its contents is prohibited.

The information in the *PlayStation®2 IOP Library Reference - Network Libraries* manual is subject to change without notice. The content of this book is Confidential Information of Sony Computer Entertainment.

and PlayStation are registered trademarks of Sony Computer Entertainment Inc. All other trademarks are property of their respective owners and/or their licensors.

Summary Table of Contents

About This Manual

This is the Runtime Library Release 2.4.2 version of the *PlayStation®2 IOP Library Reference - Network Libraries* manual.

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

Changes Since Last Release

Chapter 1: Network (INET) Configuration Library

- A description of the sceInetCtlGetState() function has been added.

Chapter 4: Common Network Configuration Library

- A description of the sceNetCnfCheckCapacity() function has been added.

Related Documentation

Library specifications for the EE can be found in the *PlayStation®2 EE Library Reference* manuals and the *PlayStation®2 EE 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 Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: PS2_Support@playstation.sony.com Web: http://www.devnet.scea.com/ Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 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 Sony Computer Entertainment Europe 30 Golden Square London W1F 9LD, U.K. Tel: +44 (0) 20 7859-5000	E-mail: ps2_support@scee.net Web: https://www.ps2-pro.com/ Developer Support Hotline: +44 (0) 20 7859-5777 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT)

Chapter 1: Network (INET) Configuration Library
Table of Contents

Structures

sceInetCtlEventHandlers

Event handler registration structure

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	March 26, 2001

Structure

typedef struct sceInetCtlEventHandlers {

struct sceInetCtlEventHandlers **forw, *back;*	forw: Forward link used internally by inetctl.irx back: Backward link used internally by inetctl.irx
void (**func***)(int id, int type);**	Pointer to event handler function
int *gp;*	$gp value storage area when event handler is called

} sceInetCtlEventHandlers_t;

Description

This is a calling data structure that becomes an argument when sceInetCtlRegisterEventHandler() is used to register the event handler. Only the *func* member should be set by the caller.

This data area must be kept until registration is canceled with sceInetCtlUnregisterEventHandler().

See also

sceInetCtlRegisterEventHandler(), sceInetCtlUnregisterEventHandler()

Control Functions

sceInetCtlDownInterface

Take down interface

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	March 26, 2001

Syntax

```
#include <inet/inetctl.h>
int sceInetCtlDownInterface(
 int id);                          Interface ID
```

Calling conditions

Can be called from a thread

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

Description

This function takes down the interface specified by *id*. When 0 is specified for *id*, this function takes down all interfaces that can be taken down.

Return value

0

sceInetCtlGetState

Get transition state of interface

Library	*Introduced*	*Documentation last modified*
inetctl	2.4.2	December 3, 2001

Syntax

```
#include <inet/inetctl.h>
int sceInetCtlGetState(
 int id,                    Interface ID
 int *pstate);              Pointer to area for storing transition state code
```

Calling conditions

Can be called from a thread

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

Description

This function sets one of the following as the transition state code when processing completes normally.

sceINETCTL_S_DETACHED	Interface is not connected
sceINETCTL_S_STARTING	Up requested, waiting for Running state
sceINETCTL_S_RETRYING	Waiting for redial_interval to elapse
sceINETCTL_S_STARTED	Running state
sceINETCTL_S_STOPPING	Down requested, waiting for Stopped state
sceINETCTL_S_STOPPED	Interface is in Stopped state

When the interface is PPP, sceINETCTL_IEV_Error is reported even if redial processing has started. At this time, the sceInetCtlGetState() function can be called to determine whether redial processing is in progress. The following processing should be performed.

- When the transition state code is sceINETCTL_S_STOPPED:
  Display "Connection failed".
- Otherwise:
  Display "Redialing" and wait for the next event to occur.

Return value

0	Normal termination
sceNETCNF_NG	Specified ID does not exist

sceInetCtlRegisterEventHandler

Register event handler

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	March 26, 2001

Syntax

#include <inet/inetctl.h>

int sceInetCtlRegisterEventHandler(

sceInetCtlEventHandlers_t ****eh*);** Pointer to event handler registration data

Calling conditions

Can be called from a thread

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

Description

This function registers an event handler that is called when events such as Attach / Detach / Start / Stop / Error of an interface occur. In the *eh* argument, only the func member should be set by the caller.

The event handler prototype is as follows.

```
void interface_event_handler(
     int id,
     int type
);
```

Multiple event handlers can be registered to the extent permitted by the INETCTL memory area.

The interface ID of the interface at which the event occurred is passed in the id argument to the event handler and one of the following values, which indicates the type of event, is passed in the type argument.

Table 1

Constant	Meaning
sceINETCTL_IEV_Attach	Interface was attached
sceINETCTL_IEV_Detach	Interface was detached
sceINETCTL_IEV_Start	Interface was brought up and is available
sceINETCTL_IEV_Stop	Interface was brought down and is unavailable
sceINETCTL_IEV_Error	Interface reported an error (Error=1)
sceINETCTL_IEV_Conf	Configuration that matches device exists
sceINETCTL_IEV_NoConf	No configuration that matches device exists

Either an sceINETCTL_IEV_Conf or sceINETCTL_IEV_NoConf event is reported directly before sceINETCTL_IEV_Attach. When the event handler is called, the $gp value when the event handler was registered is set in the $gp register.

No wait processing can be performed within the event handler. If wait processing is performed for another resource, normal inetctl.irx control and the notification of the next event will not be able to be performed from the time the event handler is called until it returns control because there is only one thread within inetctl.irx.

Return value

0

sceInetCtlSetAutoMode

Set/cancel auto mode

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	July 2, 2001

Syntax

#include <inet/inetctl.h>

int sceInetCtlSetAutoMode(

int *f_auto*);

== 0 Non-auto mode
!= 0 Auto mode

Calling conditions

Can be called from a thread

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

Description

The auto mode setting, which is specified according to whether or not the -no_auto option was specified when inetctl.irx was started up, is changed according to the *f_auto* argument of this function.

In auto mode, inetctl.irx automatically reports the relevant configuration to inet.irx and brings up the interface whenever a device is attached or an INETCTL setting is overwritten.

In non auto mode, inetctl.irx does not report the configuration to inet.irx and does not bring up the interface unless sceInetCnfUpInterface() is called.

Return value

0

sceInetCtlSetConfiguration

Write configuration in INETCTL

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	March 26, 2001

Syntax

#include <netcnf.h>

#include <inet/inetctl.h>

int sceInetCtlSetConfiguration(

sceNetCnfEnv_t *e); Pointer to load environment

Calling conditions

Can be called from a thread

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

Description

This function replaces the configuration data that is referenced by inetctl.irx with the load environment that was specified by the e argument and restarts inetctl.irx. That is, it writes the configuration in INETCTL according to the *e* argument. For information about how the configuration is actually reported to INET, see sceInetCtlSetAutoMode().

The load environment is a data structure that corresponds to one entry in the configuration management file. For details, see the NETCNF document.

The main member that is referenced in the environment is e->root, and an interface definition that can be traced from e->root->pair_head must also be set.

Notes

The load environment that was passed to sceInetCtlSetConfiguration() must not be released or changed until "do not use" is explicitly set for it.

To set "do not use" for the load environment, specify NULL as the *e* argument and call sceInetCtlSetConfiguration(). From then on, devices will no longer be able to be attached according to the previous configuration data.

Return value

sceNETCNF_NG Configuration failed

sceInetCtlUnregisterEventHandler

Delete event handler registration

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	March 26, 2001

Syntax

#include <inet/inetctl.h>

int sceInetCtlUnregisterEventHandler(

sceInetCtlEventHandlers_t ****eh*);** Pointer to event handler registration data

Calling conditions

Can be called from a thread

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

Description

This function deletes an event handler that was registered with sceInetCtlRegisterEventHandler().

Return value

0

sceInetCtlUpInterface

Bring up interface

Library	*Introduced*	*Documentation last modified*
inetctl	2.2	March 26, 2001

Syntax

```
#include <inet/inetctl.h>
int sceInetCtlUpInterface(
 int id);                          Interface ID
```

Calling conditions

Can be called from a thread

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

Description

This function brings up the interface specified by the id argument, after the current configuration contents are reported to inet.irx. If 0 was specified for id, this function brings up all interfaces that can be brought up and for which configuration data exists.

Return value

0

Chapter 2: Network Library
Table of Contents

Structures

sceInetAddress_t

Internal-format IP address structure

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Structure

```
typedef struct sceInetAddress {
    int reserved;          Reserved area (always 0)
    char data[12];         IP address (4 bytes) + Reserved (8 bytes)
} sceInetAddress_t;
```

Description

This is a structure for maintaining the IP address within the library.

In IPv4, the IP address is maintained in the first four bytes of the data member. However, to prepare for future extensions, a user program must not directly access this structure. sceInetName2Address() and sceInetAddress2String() should be used to obtain the IP address.

sceInetInfo_t

Connection information delivery structure

Library	*Introduced*	*Documentation last modified*
inet	2.2	July 2, 2001

Structure

```
typedef struct sceInetInfo {
    int cid;                          Connection ID
    int proto;                        Protocol
    int recv_queue_length;            Number of data bytes in receive buffer
    int send_queue_length;            Number of data bytes in send buffer
    struct sceInetAddress local_adr;  Local address
    int local_port;                   Local port number
    struct sceInetAddress remote_adr; Remote address
    int remote_port;                  Remote port number
    int state;                        Connection state
    int reserved[4];                  Reserved area (0)
} sceInetInfo_t;
```

Description

This is a structure that is used when information related to a Connection is obtained by specifying sceINETC_CODE_GETINFO (=1) for the *code* argument in sceInetControl().

Any of the following constants may be stored in the *proto* member.

Table 2-1

Constant	Value	Meaning (Corresponding Connection Type)
sceINETI_PROTO_TCP	1	TCP (sceINETT_CONNECT or sceINETT_LISTEN)
sceINETI_PROTO_UDP	2	UDP (sceINETT_DGRAM)
sceINETI_PROTO_IP	3	Raw IP (sceINETT_RAW)

Any of the following constants may be stored in the *state* member.

Table 2-2

Constant	Value	Meaning (Corresponding Connection Type)
sceINETI_STATE_UNKNOWN	0	State unknown (TCP, UDP, Raw IP)
sceINETI_STATE_CLOSED	1	Closed (TCP, UDP, Raw IP)
sceINETI_STATE_CREATED	2	Created (UDP)
sceINETI_STATE_OPENED	3	Opened (UDP, Raw IP)
sceINETI_STATE_LISTEN	4	TCP internal state
sceINETI_STATE_SYN_SENT	5	TCP internal state
sceINETI_STATE_SYN_RECEIVED	6	TCP internal state
sceINETI_STATE_ESTABLISHED	7	TCP internal state
sceINETI_STATE_FIN_WAIT_1	8	TCP internal state

Constant	Value	Meaning (Corresponding Connection Type)
sceINETI_STATE_FIN_WAIT_2	9	TCP internal state
sceINETI_STATE_CLOSE_WAIT	10	TCP internal state
sceINETI_STATE_CLOSING	11	TCP internal state
sceINETI_STATE_LAST_ACK	12	TCP internal state
sceINETI_STATE_TIME_WAIT	13	TCP internal state

See also

sceINETC_CODE_GETINFO

sceInetParam_t

Connection creation parameter structure

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Structure

```
typedef struct sceInetParam {
    int type;                       Connection type (TCP, UDP, etc.)
    int local_port;                 Local port number
    struct sceInetAddress remote_addr;  Destination IP address
    int remote_port;                Destination port number
    int reserved[9];                Reserved area (always 0)
} sceInetParam_t;
```

Description

This structure contains the parameters to be passed when a Connection is created with sceInetCreate().

The Connection type is specified in the *type* member with the following constants.

Table 2-3

Constant	Value	Meaning
sceINETT_DGRAM	0x0	UDP
sceINETT_CONNECT	0x1	TCP Connect operation (Active-Open)
sceINETT_LISTEN	0x2	TCP Listen operation (Passive-Open)
sceINETT_RAW	0x3	Raw IP (direct handling of IP packets)

For the port number specifications in the *local_port* and *remote_port* members, use either a numeric value representing the port number or one of the following constants.

Table 2-4

Constant	Value	Meaning
sceINETP_AUTO	-1	Automatic assignment
sceINETP_ANY	0	Any port

If sceINETP_AUTO is specified for local port when the Connection type is sceINETT_CONNECT or for remote_port when the Connection type is sceINETT_LISTEN, a free port among the user ports (5000 to 65535) is automatically assigned when sceInetCreate() is called.

If sceINETP_ANY is specified for remote_port when the Connection type is sceINETT_LISTEN, and if the connection is a TCP connection, the port number is determined by the first receive, and the port number of that transmission source is used for subsequent communications. However, if the connection is a UDP connection, the port number is not determined in this way. Arbitrary port numbers are accepted for subsequent receives.

An internal-format IP address of the connection destination is specified for remote_addr. When the internal format (see sceInetName2Address) matches any address, if the connection is a TCP connection, the IP address is determined by the first receive. If the connection is a UDP connection or Raw IP connection, the IP address is not determined in this way. Receive data from any IP address will be accepted.

See also

sceInetCreate()

sceInetRoutingEntry_t

Routing control table entry

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Structure

```
typedef struct sceInetRoutingEntry {
    struct sceInetAddress dstaddr;      Destination address
    struct sceInetAddress gateway;      Next POP router address
    struct sceInetAddress genmask;      Subnet mask
    int flags;                          Flags indicating the state
    int mss;                            Maximum segment size
    int window;                         TCP window size
    char interface[8 + 1];              Network interface name
} sceInetRoutingEntry_t;
```

Description

This is a structure for storing routing control table entry information.

The value obtained by taking the logical OR of the following bit flags is entered in the *flags* member.

Table 2-5

Constant	Value	Meaning
sceInetRoutingF_Up	0x01	Route is valid
sceInetRoutingF_Host	0x02	Direct delivery (not via a router)
sceInetRoutingF_Gateway	0x04	Indirect delivery (via a router)
sceInetRoutingF_Dynamic	0x08	Dynamically set
sceInetRoutingF_Modified	0x10	Same entry with modification

Although the window size (windows) can be referenced, that value currently cannot be used within INET.

See also

sceInetAddress_t, sceInetGetRoutingTable()

Error Code

INET function return value

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Table 2-6

Constant	Value	Meaning
sceINETE_OK	0	Normal termination
sceINETE_TIMEOUT	-500	Timeout specified by argument in function occurred or TCP resend timeout occurred
sceINETE_ABORT	-501	Interruption due to sceInetAbort() call
sceINETE_BUSY	-502	INET module initialization is not completed
sceINETE_LINK_DOWN	-503	Device initialization or connection processing is not completed
sceINETE_INSUFFICIENT_ RESOURCES	-504	Insufficient memory area
sceINETE_LOCAL_SOCKET_ UNSPECIFIED	-505	Invalid value was specified as local_port
sceINETE_FOREIGN_ SOCKET_ UNSPECIFIED	-506	Invalid value was specified as remote_addr or remote_port
sceINETE_CONNECTION_ ALREADY_EXISTS	-507	An attempt was made to open a Connection that was already established
sceINETE_CONNECTION_ DOES_NOT_EXIST	-508	No Connection was established
sceINETE_CONNECTION_ CLOSING	-509	Connection status is Closing (TCP only)
sceINETE_CONNECTION_ RESET	-510	Connection was Reset (TCP only)
sceINETE_CONNECTION_ REFUSED	-511	Connection was Reset when status is Syn-Received (TCP only)
sceINETE_INVALID_ ARGUMENT	-512	An argument is invalid
sceINETE_INVALID_CALL	-513	Invalid function call
sceINETE_NO_ROUTE	-514	No routing to destination exists

An IOP Kernel API return value (KE_xxx) may be returned as a negative value representing an error besides those listed above.

Functions

sceInetAbort

Abort processing

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

#include <inet.h>

int sceInetAbort(

int *cid,*	Connection ID
int *flags***);**	Reserved (0 must be specified)

Calling conditions

Can be called from a thread

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

Description

This function cancels the wait states of all functions (sceInetOpen, sceInetClose, sceInetRecv, sceInetRecvFrom, sceInetSend, or sceInetSendTo) that are in wait states related to the Connection specified by *cid*. An individual function cannot be specified.

All aborted functions return with an sceINETE_ABORT error.

The *flags* argument is reserved for future use. Zero must always be specified for this argument.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetAbortLog

Abort acquisition of log

Library	*Introduced*	*Documentation last modified*
inet	2.4	October 11, 2001

Syntax

#include <inet.h>

int sceInetAbortLog(void);

Calling conditions

Can be called from a thread

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

Description

This function aborts the wait state by sceInetGetLog() using sceINETE_ABORT.

Return value

0 Normal termination

>0 Number of times errors occurred for SetEventFlag()

sceInetAddress2Name

Convert dot-format IP address to host name (reverse lookup)

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetAddress2Name(
 int flags,                    Conversion flags (always 0)
 char *buf,                    Buffer address for storing conversion results
 int len,                      Buffer length (bytes)
 sceInetAddress_t *paddr       Internal-format IP address
 int ms,                       Timeout interval (ms)
 int nretry);                  Retry count
```

Calling conditions

Can be called from a thread

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

Description

This function sends a reverse lookup inquiry to DNS based on an internal-format IP address and stores the domain name that is obtained in the area specified by *buf* and *len*.

Zero must always be specified for the conversion flags (*flags* argument).

The timeout interval (*ms*) specifies the timeout interval for an inquiry to one DNS. When 0 or a negative value is specified, it is treated as the default value of 6 seconds.

If multiple DNSs have been set and a timeout occurs for an inquiry to a given single DNS, an inquiry to another DNS is attempted. If timeouts occur for the specified number of retries (*nretry*) during inquiries to all DNSs, this function returns with an error (sceINETE_TIMEOUT).

When 0 or a negative value is specified for the retry count (*nretry*), it is treated as the default value of 4 retries.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetAddress2String

Conversion from internal-format IP address to dot format

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetAddress2String(
 char *buf,                    Buffer address for storing conversion results
 int len,                      Buffer length (bytes)
 sceInetAddress_t *paddr);     Internal-format IP address
```

Calling conditions

Can be called from a thread

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

Description

This function converts an internal-format IP address to a dot-format character string.

It is a display and debugging function.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetChangeThreadPriority

Change thread priority

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

#include <inetctl.h>

int sceInetChangeThreadPriority(

int *prio***);** Priority (1-63)

Calling conditions

Can be called from a thread

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

Description

This function changes the priority of a thread that was created by the INET layer to *prio*. The priorities of subsequently created threads will also be set to *prio*.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetClose

Close Connection

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetClose(
 int cid,                    Connection ID
 int ms);                    Timeout interval (ms)
```

Calling conditions

Can be called from a thread

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

Description

This function closes a Connection.

If unsent data (a confirmation response after transmission has not been confirmed) remains in the buffer, this function waits until the transmission is completed and then sends FIN and waits for a Time-Wait or Close state to occur.

The timeout interval (*ms*) is the upper limit for the total of these two waiting intervals. The Connection ID that was specified by *cid* becomes invalid when sceInetClose() is called. No subsequent processing can be performed by using that Connection ID, including sceInetAbort().

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetControl

Control Connection

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetControl(
 int cid,                    Connection ID to be controlled
 int code,                   Control code representing operation contents
 void *ptr,                  Starting address of data area
 int len)                    Size of data area (bytes)
```

Calling conditions

Can be called from a thread

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

Description

This function performs various control operations for the Connection specified by *cid*. The operation is specified by the *code* argument. The currently defined control codes are as follows.

Table 2-7

Control code	Value	Function
sceInetC_CODE_GET_INFO	1	Get Connection information
sceInetC_CODE_GET_FLAGS	2	Get control flags
sceInetC_CODE_SET_FLAGS	3	Set control flags

For details, see the "Control Codes" section.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCreate

Create Connection

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetCreate(
 sceInetParam_t *param);          Connection parameter
```

Calling conditions

Can be called from a thread

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

Description

This function creates a Connection and returns the Connection ID (cid).

After this Connection is created, no packets are transmitted externally and all received packets are discarded until sceInetOpen() is called.

Return value

Connection ID

sceInetGetInterfaceList

Get list of network interfaces

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetGetInterfaceList(
 int *interface_id,        Buffer address for storing interface ID list
 int n);                   Maximum number of interface IDs to be obtained
```

Calling conditions

Can be called from a thread

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

Description

This function gets a list of IDs of connected network interfaces. The buffer size must be 4*n* bytes.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetGetLog

Get log

Library	*Introduced*	*Documentation last modified*
inet	2.2	October 11, 2001

Syntax

```
#include <inetctl.h>
int sceInetGetLog(
 char *buf,          Pointer to buffer for storing log
 int len,            Size of buffer for storing log (bytes)
 int ms);            Interval until timeout (ms)
```

Calling conditions

Can be called from a thread

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

Description

This function gets the log that is being saved internal to INET and stores it in the area indicated by *buf* and *len*.

This function blocks and will not return until the log is obtained, a timeout occurs, or it is aborted by sceInetAbortLog().

If the third argument is negative (-1), no timeout will be set.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetGetNameServers

Get name server information

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetGetNameServers(
 sceInetAddress_t *paddr,        Pointer to area for getting name server information
 int n);                         Maximum number of entries to be obtained
```

Calling conditions

Can be called from a thread

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

Description

This function gets the current name server address.

The size of the area that is pointed to by *paddr* must be sizeof(sceInetAddress_t) * *n* bytes.

Return value

When processing terminates normally, the number of entries (greater than or equal to 0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetGetRoutingTable

Get routing control table

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetGetRoutingTable(
 sceInetRoutingEntry_t *p,        Pointer to area for getting routing control tables
 int n);                          Maximum number of entries to be obtained
```

Calling conditions

Can be called from a thread

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

Description

This function gets up to n of the routing control tables maintained by INET and stores them at the address pointed to by *p*. The size of the required area is sizeof(sceInetRoutingEntry_t) * *n* bytes.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetInterfaceControl

Reference or set parameters of network interface

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,        ID of network interface to be controlled
 int code,                Control code for representing operation contents
 void *ptr,               Starting address of data area
 int len);                Size of data area (bytes)
```

Calling conditions

Can be called from a thread

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

Description

This function references parameters of the network interface (it can also set some of the parameters).

The *code* argument, which is a code that represents the type of control, is classified into the following five types.

Table 2-8

Range	Type
0x00000000 - 0x7fffffff	INET layer control code
0x80000000 - 0x8fffffff	NETDEV layer common control code (*)
0x90000000 - 0xbfffffff	NETDEV module-dependent control code (*)
0xc0000000 - 0xcfffffff	MODEM layer common control code (*)
0xd0000000 - 0xffffffff	MODEM module-dependent control code (*)

When bit 31 of the code argument is 1 (indicated by (*) above), the *code*, *ptr*, and *len* arguments are relayed directly to the relevant NETDEV module without any special processing being performed by the INET layer.

The *ptr* and *len* arguments indicate the starting address of the data area accompanying a data transfer and the size in bytes of that area.

Depending on the control code, data may also be returned as the function's return value.

The control codes that are defined for use by the INET layer are shown below. Since each control code corresponds to a member of the sceInetDevOps_t network interface structure, refer to the relevant sections of the network device interface specifications for details. For information about the control codes for other layers, refer to the corresponding documents.

Table 2-9

Control Code	Value	Function
sceInetCC_GetInterfaceName	0x00000000	Get interface name (string)
sceInetCC_GetModuleName	0x00000001	Get module name (string)
sceInetCC_ GetVendorName	0x00000002	Get vendor name (string)
sceInetCC_ GetDeviceName	0x00000003	Get device name (string)
sceInetCC_ GetBusType	0x00000004	Get bus type
sceInetCC_GetBusLoc	0x00000005	Get device location information
sceInetCC_GetProt	0x00000006	Get NETDEV layer protocol version
sceInetCC_GetImpl	0x00000007	Get NETDEV layer implementation version
sceInetCC_GetFlags	0x00000008	Get flags (flags) in sceInetDevOps structure that NETDEV layer has
sceInetCC_GetAddress	0x00000009	Get IP address of interface
sceInetCC_GetNetmask	0x0000000a	Get subnet mask of interface
sceInetCC_GetBroadcast	0x0000000b	Get broadcast address of interface
sceInetCC_GetMTU	0x0000000c	Get MTU value of interface
sceInetCC_GetHWaddr	0x0000000d	Get hardware address of interface
sceInetCC_ GetDHCPHostName	0x0000000e	Get DHCP host name
sceInetCC_ GetDHCPFlags	0x0001000f	Get DHCP control flags

Notes

A setting operation for a network interface is not performed by directly calling this function from a title application. Rather, it should be performed by using the NET configuration file and inetctl.irx. (As an exception, this function is used to make a priority setting or a direct modem layer setting from an application.)

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetName2Address

Convert from host name or dot format to internal-format IP address

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetName2Address(
```

int *flags,*	Conversion flag
sceInetAddress_t ***paddr,**	Address of structure variable for receiving internal-format IP address
char ***name,**	Dot-format IP address or host name
int *ms,*	Timeout interval (ms)
int *nretry*);	Retry count

Calling conditions

Can be called from a thread

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

Description

This function converts an IP address that is expressed in dot format to an internal-format IP address.

An IP address that is obtained from DNS (name server) by assigning a host name can also be converted to internal format.

Although 0 is normally specified for the conversion flag (flags argument), the logical OR of the following bit values can be specified as necessary.

Table 2-10

Constant	Value	Meaning
sceInetN2AF_NoDot	0x01	Dot-format conversion is prohibited
sceInetN2AF_NoHosts	0x02	(Reserved)
sceInetN2AF_NoDNS	0x04	Inquiry to DNS is prohibited

Conversion processing is performed according to the following sequence. If all steps fail, an error will occur.

1. When name is NULL, it is converted to an internal format that matches an arbitrary address.
2. If sceInetN2AF_NoDot has not been specified by the conversion flag, the name argument is treated as a dot-format IP address and conversion is attempted.
3. If sceInetN2AF_NoDNS has not been specified by the conversion flag, the name argument is treated as a host name and an inquiry to DNS is attempted.

The timeout interval (*ms*) specifies the timeout interval for an inquiry to one DNS. When 0 or a negative value is specified, it is treated as the default value of 6 seconds.

If multiple DNSs have been set and a timeout occurs for an inquiry to a given single DNS, an inquiry to another DNS is attempted. If timeouts occur for the specified number of retries (*nretry*) during inquiries to all DNSs, this function returns with an error (sceINETE_TIMEOUT).

When 0 or a negative value is specified for the retry count (*nretry*), it is treated as the default value of 4 retries.

Dot Format

Dot-format IP address means any of the following formats.

- *num8.num8.num8.num8* (Class C)
- *num8.num8.num16* (Class B)
- *num8.num24* (Class A)
- *num32* (direct specification)

num8	Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 8 bits
num16	Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 16 bits
num24	Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 24 bits
num32	Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 32 bits

The octal, decimal, or hexadecimal notation rules are the same as those used for the C language.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetOpen

Establish a Connection

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetOpen(
 int cid,                    Connection ID
 int ms);                    Timeout interval (ms)
```

Calling conditions

Can be called from a thread

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

Description

This function opens the Connection specified by *cid*.

For TCP, a wait state will occur until any of the following conditions is satisfied.

a. A Connection is established (=Established).
b. An error occurred, excluding a timeout.
c. The specified timeout interval (*ms*) elapsed (sceINETE_TIMEOUT).

For case c, this function returns sceINETE_TIMEOUT. However, if sceInetRecv() or sceInetSend() is called at this time, processing will wait for a Connection to be established in those functions.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetRecv

Receive data

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetRecv(
 int cid,                    Connection ID
 void *ptr,                  Buffer address for storing receive data
 int count,                  Data size to be received (bytes)
 int *pflags,                Address of variable for storing status flags
 int ms);                    Timeout interval (ms)
```

Calling conditions

Can be called from a thread

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

Description

This function receives data from the Connection specified by cid and stores that data in the buffer specified by *ptr* and *count*.

The logical OR of the constants shown below is specified for **pflags* as the receive termination conditions, and the logical OR of the constants shown below is also returned as the termination status.

Table 2-11

Constant	Value	Meaning
sceINETF_URG	0x02	A segment with an Urgent flag was received
sceINETF_FIN	0x04	A segment with a Fin flag was received
sceINETF_TIM	0x08	A timeout occurred

The upper limit of the interval for waiting for a data receive is specified for the *ms* argument. Specifying 0 indicates that the function is to return with no wait interval. Specifying a negative value indicates that the function is to wait indefinitely.

For a TCP connection, the function returns according to the following conditions. A value enclosed in square brackets indicates the flag that is returned in **pflags* as status.

1. An error occurred (such as the Connection was reset).
2. The number of data specified by *count* was received.
3. At least one byte of data was received.
4. FIN was received and all data up to that point was received. [sceINETF_FIN]
5. A segment with an Urgent flag was received. [sceINETF_URG]
6. The timeout interval specified by *ms* elapsed. [sceINETF_TIM]

Although condition (3) did not cause the function to return in the previous version unless a special flag was specified, this has currently been changed so that it causes the function to return by default. When Urgent data is received, only the Urgent data is transferred to the buffer indicated by *ptr*. Normal data and Urgent data cannot coexist.

For a UDP or Raw IP connection, this function should be called with zero specified for **pflags.* In this case, when one packet is received, the function returns regardless of the *count* specification. When more packets than count are received, only the specified size data is transferred to the buffer, and the remaining packets are discarded. The only value that is set in **pflags* as the status is sceINETF_TIM.

Even if a timeout occurs, the number of bytes that were received up to that point will be the return value of the function. Whether or not a timeout occurred can be determined by the sceINETF_TIM bit of **pflags*.

If the connection is closed from the connection destination (if FIN is received) when a negative value has been specified for the timeout interval (*ms*) (i.e. an indefinite wait has been specified), the return value of the function will be zero. However, since zero is also returned when a timeout occurs if zero or a positive value has been specified for the timeout interval, the sceINETF_FIN bit of **pflags* should be referenced to determine whether the connection was closed from the connection destination.

When zero is specified for the timeout interval (*ms*), the function will return with no wait interval. However, since other threads will no longer operate, processing for repeatedly reading with no wait interval, as shown below, must not be performed. A receive thread should be created for each Connection, and an unlimited timeout interval or suitable timeout interval should be specified.

```
while(1){
     flags = 0;
     if(0 > (r = sceInetRecv(cid, buf, sizeof(buf), &flags, 0)))
         Error handling;
     else
         Receive processing;
}
```

Notes

The size of the TCP receive buffer is fixed at 31.3K bytes, the size of the UDP receive buffer is unlimited, and the size of the Raw IP receive buffer is fixed at 32K bytes. Since TCP is a reliable protocol, no data will be lost even if the receive buffer is full. We have plans to allow buffer size to be specified in the future.

Return value

The number of bytes that were received is returned.

sceInetRecvFrom

Receive data (also get IP address and port number of sender)

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetRecvFrom(
```

int *cid,*	Connection ID
void * *ptr,*	Buffer address for storing reception data
int *count,*	Data size to be received (bytes)
int * *pflags,*	Address of variable for storing status flags
sceInetAddress_t * *iadr,*	Address of variable for storing IP address of sender
int * *port,*	Address of variable for storing port
int *ms*);	Timeout interval (ms)

Calling conditions

Can be called from a thread

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

Description

This function receives data from the Connection specified by *cid* and stores that data in the buffer specified by *ptr* and *count*, in a similar manner as sceInetRecv(), but it also stores the IP address of the sender in **iadr* and the port of the sender in *po*rt*. This function can only be used for a UDP connection. If it is used for a TCP connection, an error (sceINETE_INVALID_CALL) will occur.

Since UDP, unlike TCP, is stateless, data can be received from multiple destinations by using a single Connection, if remote_addr is set to an arbitrary value in sceInetParam or remote_port is set to sceINETP_ANY when the Connection is created. sceInetRecvFrom() is used in this case to determine the destination from which the data was received.

Return value

The number of bytes that were received is returned.

sceInetSend

Send data

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
int sceInetSend(
```

int *cid,*	Connection ID
void * *ptr,*	Buffer address where send data was stored
int *count,*	Size of send data (bytes)
int * *pflags,*	Address of variable for storing status flags
int *ms*);	Timeout interval (ms)

Calling conditions

Can be called from a thread

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

Description

This function sends the data that is in the buffer specified by *ptr* and *count* to the Connection specified by *cid* and returns the number of bytes that were sent if no error occurs.

For a TCP connection, the sceINETF_URG bit can be specified in **pflags*. When the sceINETF_URG bit is specified, no wait state occurs because the data is directly sent as Urgent data. However, for normal data, a wait state occurs if there is no free space in the send buffer. The timeout interval (*ms*) specifies an upper limit on waiting for free space in this send buffer.

With the previous version, by specifying a special flag, the segment that contained the last data of the specified data was sent with an appended PUSH flag. However, with the current version, the PUSH flag is automatically added inside the stack, and the application need not be aware of this flag.

TCP performs resend processing if no confirmation response is returned from the send destination. At first the interval for waiting for the confirmation response is 1 second. If no confirmation response arrives, this interval is sequentially doubled until it reaches 24 seconds and the data is resent a maximum of 12 times. Therefore, even if a sufficiently large value is specified for the timeout interval (*ms*), if there is no confirmation response, a timeout will occur after 447 seconds.

Even if a timeout occurs, sceInetSend() returns the number of transmitted bytes. Whether or not a timeout occurred, can be determined by the sceINETF_TIM bit in the value returned in **pflags*. This is the only bit that is set by INET.

When zero is specified for the timeout interval (*ms*), the function will return with no wait interval. However, since other threads will no longer operate, processing for repeatedly sending data with no wait interval, as shown below, must not be performed. A thread should be created for each Connection, and an unlimited timeout interval or suitable timeout interval should be specified.

```
while(1){
    flags = 0;
    if(0 > (r = sceInetSend(cid, buf, sizeof(buf), &flags, 0)))
        Error handling;
    else
        Send processing;
}
```

For UDP and Raw IP, since send data is not sent via a buffer, no wait state will occur.

Notes

The size of the TCP send buffer is fixed at 31.3K bytes. Although we have plans to allow the buffer size to be specified in the future, the time frame for doing this has not been decided.

The timing of send processing can be adjusted with the setting of the sceINETCC_CODE_SET_FLAGS control code. For details, see sceInetControl().

Return value

The number of bytes that were sent is returned.

sceInetSendTo

Send data to specified IP address and port number

Library	*Introduced*	*Documentation last modified*
inet	2.2	December 3, 2001

Syntax

#include <inet.h>

int sceInetSendTo(

int *cid,*	Connection ID
**void ***ptr,*	Buffer address where send data was stored
int *count,*	Size of send data (bytes)
int *pflags,*	Address of variable for storing status flags
**sceInetAddress_t ***iadr,*	Address of variable for storing IP address of send destination
int *port,*	Port
int *ms***);**	Timeout interval (ms)

Calling conditions

Can be called from a thread

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

Description

This function specifies the IP address of the send destination (**iadr*) and the port (*port*) to send the data that is in the buffer specified by *ptr* and *count* for the Connection specified by *cid*, in a similar manner as sceInetSend(), and returns the number of bytes that were sent if no error occurs. This function can only be called for a UDP connection. If it is called for a TCP connection, an error (sceINETE_INVALID_CALL) will occur.

Since UDP, unlike TCP, is stateless, data can be sent to multiple destinations by using a single Connection ID, if remote_addr is set to an arbitrary value or remote_port is set to sceINETP_ANY in sceInetParam when sceInetCreate() is called. sceInetSendTo() is used in this case to specify the IP address of the destination and the port number.

If a transmission is attempted by specifying a different value for a Connection for which remote_addr is not an arbitrary value and remote_port is not sceINETP_ANY, an error (sceINETE_INVALID_ARGUMENT) will occur.

Currently, no value is returned to **pflags*.

Return value

The number of bytes that were sent is returned.

Control Codes for Connection Control

sceInetC_CODE_GET_FLAGS

Get control flags

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

int sceInetControl(

int *cid,*	Connection ID for which operation is to be performed
sceINETC_CODE_GET_FLAGS,	
**void ****ptr,*	Starting address of data area
int *len***)**	Size of data area (=sizeof(int))

Description

This function gets the control flags related to the Connection specified by *cid* and stores them in the area specified by *ptr* and *len*. For *len*, sizeof(int) should be specified. For the meaning of each control flag, see the description of sceInetC_CODE_SET_FLAGS.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetC_CODE_GET_INFO

Get Connection information

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

int sceInetControl(

int *cid,*	Connection ID for which operation is to be performed
sceINETC_CODE_GET_INFO,	
**void ****ptr,*	Starting address of data area (sceInetInfo_t)
int *len***)**	Size of data area (bytes)

Description

This function stores information (sceInetInfo_t structure) related to the Connection specified by *cid* in the area specified by *ptr* and *len*. Normally, sizeof(sceInetInfo_t) is specified for *len*.

If zero is specified for *cid*, information related to all TCP, UDP and Raw IP Connections will be stored in the area specified by *ptr* and *len* as an array of sceInetInfo_t structures. In this case, be sure to prepare a sufficiently large area. A return value that is greater than or equal to zero indicates the number of sceInetInfo_t structures (that is, the number of Connections) that are stored in the specified area.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetC_CODE_SET_FLAGS

Set control flags

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

int sceInetControl(

int *cid,*	Connection ID for which operation is to be performed
sceINETC_CODE_SET_FLAGS,	
void **ptr,*	Starting address of data area (control flags)
int *len***)**	Size of data area (=sizeof(int))

Description

This function sets the value that is stored in the area specified by *ptr* and *len* as control flags related to the Connection specified by *cid*. sizeof(int) should be specified for *len*.

The following control flags are currently defined.

Table 2-12

Control Flag	Value (Bit Position)	Meaning
sceINETC_FLAGS_NODELAY	0x01 (bit0)	TCP no delay 0 Nagle algorithm enabled 1 Nagle algorithm disabled

To increase transmission efficiency, processing (Nagle algorithm) for delaying transmission until the segment is of a certain size is applied by default with TCP. If the sceINETC_FLAGS_NODELAY control flag is set to 1, this delay processing is not performed.

This is effective for an application that is sensitive to delay such as an application that frequently exchanges small data.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

INET Layer Control Codes

sceInetCC_GetAddress

Get IP address

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                    Network interface ID for which operation is to be
                                      performed
 sceInetCC_GetAddress,   // 0x00000009
 void *ptr,                           Starting address of data area
 int len)                             Size of data area (sizeof(sceInetAddress_t))
```

Description

This function gets the IP address of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not sizeof(sceInetAddress_t), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetBroadcast

Get broadcast address

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                    Network interface ID for which operation is to be performed
 sceInetCC_GetBroadcast,// 0x0000000b
 void *ptr,                           Starting address of data area
 int len)                             Size of data area (sizeof(sceInetAddress_t))
```

Description

This function gets the broadcast address of the specified network interface and stores it in the area specified by (ptr, len).

If len is not sizeof(sceInetAddress_t), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetBusLoc

Get device location information

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                  Network interface ID for which operation is to be
                                    performed
 sceInetCC_GetBusLoc,  // 0x00000005
 void *ptr,                         Starting address of data area
 int len)                           Size of data area (31 bytes)
```

Description

This function gets location information on the bus for the device corresponding to the specified network interface and stores it in the area specified by (*ptr, len*).

Currently, this function is valid only when the bus type is sceInetBus_USB. The position information is in the format returned by UsbdGetDeviceLocation.

The size is set to 31 bytes to provide room for future use.

If *len* is not 31, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetBusType

Get bus type

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                    Network interface ID for which operation is to be performed
 sceInetCC_GetBusType,   // 0x00000004
 void *ptr,                           Starting address of data area
 int len)                             Size of data area (sizeof(u_char))
```

Description

This function gets the bus type (currently only sceInetBus_USB) of the specified network interface and stores it in the area specified by (*ptr, len*).

The following bus types are currently defined.

Table 2-13

Bus type	Value	Meaning
sceInetBus_USB	1	USB ethernet
sceInetBus_NIC	5	HDD ethernet

If *len* is not sizeof(u_char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetDeviceName

Get device name

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                    Network interface ID for which operation is to be
                                      performed
 sceInetCC_GetDeviceName, // 0x00000003
 void *ptr,                           Starting address of data area
 int len)                             Size of data area (bytes)
```

Description

This function gets the device name of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not large enough to store the device name, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetDHCPFlags

Get DHCP control flags

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
int interface_id,                    Network interface ID for which operation is to be performed
sceInetCC_GetDHCPFlags, // 0x0000000f
void *ptr,                           Starting address of data area
int len)                             Size of data area (sizeof(int))
```

Description

This function gets the DHCP control flags of the specified network interface and stores them in the area specified by (ptr, len).

The following DHCP control flags are currently defined.

Table 2-14

Control Flag	Value (Bit Position)	Meaning
sceInetDevDHCP_RelOnStop	0x00000001	DHCPRELEASE is issued when the interface goes down

If *len* is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetDHCPHostName

Get DHCP host name

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
int interface_id,                    Network interface ID for which operation is to
                                     be performed
sceInetCC_GetDHCPHostName, // 0x0000000e
void *ptr,                           Starting address of data area
int len)                             Size of data area (bytes)
```

Description

This function gets the DHCP host name of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not large enough to store the DHCP host name, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetFlags

Get flags

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
int interface_id,                    Network interface ID for which operation is to be performed
sceInetCC_GetFlags,    // 0x00000008
void *ptr,                           Starting address of data area
int len)                             Size of data area (sizeof(int))
```

Description

This function gets the flags (flags) in the sceInetDevOps structure kept by the NETDEV module that corresponds to the specified network interface and stores them in the area specified by (*ptr, len*). For details about each flag, refer to the network device interface specifications.

If *len* is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetHWaddr

Get hardware address

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                  Network interface ID for which operation is to be
                                    performed
 sceInetCC_GetHWaddr, //0x0000000d
 void *ptr,                         Starting address of data area
 int len)                           Size of data area (16 bytes)
```

Description

This function gets the hardware address (MAC address) of the specified network interface and stores it in the first 6 bytes of the area specified by (*ptr, len*). This function is valid only when that network interface is an Ethernet interface.

Although the value that is returned as the hardware address is 6 bytes, a 16-byte area should be prepared as the data delivery buffer. If *len* is not 16, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetImpl

Get implementation version

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                  Network interface ID for which operation is to be
                                    performed
 sceInetCC_GetImpl,    // 0x00000007
 void *ptr,                         Starting address of data area
 int len)                           Size of data area (sizeof(u_short))
```

Description

This function gets the implementation version of the NETDEV module corresponding to the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not sizeof(u_short) (=2), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetInterfaceName

Get interface name

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
int interface_id,                    Network interface ID for which operation is to be performed
sceInetCC_GetInterfaceName, // 0x00000000
void *ptr,                           Starting address of data area
int len)                             Size of data area (8 + 1 bytes)
```

Description

This function gets the interface name (such as eth0 or ppp0) of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not 8 + 1, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetModuleName

Get module name

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                    Network interface ID for which operation is to be performed
 sceInetCC_GetModuleName,   // 0x00000001
 void *ptr,                           Starting address of data area
 int len)                             Size of data area (bytes)
```

Description

This function gets the module name of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not large enough to store the module name, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetMTU

Get MTU value

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                  Network interface ID for which operation is to be
                                    performed
 sceInetCC_GetMTU,    // 0x0000000c
 void *ptr,                         Starting address of data area
 int len)                           Size of data area (sizeof(int))
```

Description

This function gets the MTU value of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetNetmask

Get subnet mask

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
int interface_id,                    Network interface ID for which operation is to be performed
 sceInetCC_GetNetmask,  // 0x0000000a
 void *ptr,                          Starting address of data area
 int len)                            Size of data area (sizeof(sceInetAddress_t))
```

Description

This function gets the subnet mask of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not sizeof(sceInetAddress_t), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetProt

Get protocol version

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                  Network interface ID for which operation is to be
                                    performed
 sceInetCC_GetProt,    // 0x00000006
 void *ptr,                         Starting address of data area
 int len)                           Size of data area (sizeof(u_short))
```

Description

This function gets the protocol version of the NETDEV module corresponding to the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not sizeof(u_short) (=2), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetCC_GetVendorName

Get vendor name

Library	*Introduced*	*Documentation last modified*
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int sceInetInterfaceControl(
 int interface_id,                    Network interface ID for which operation is to be
                                      performed
 sceInetCC_GetVendorName,   // 0x00000002
 void *ptr,                           Starting address of data area
 int len)                             Size of data area (bytes)
```

Description

This function gets the vendor name of the specified network interface and stores it in the area specified by (*ptr, len*).

If *len* is not large enough to store the vendor name, an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

Chapter 3: Modem Development Library
Table of Contents

Modem Information Structure

sceModemOps

Modem information structure

Library	*Introduced*	*Documentation last modified*
modem	2.4	October 11, 2001

Syntax

```
typedef struct sceModemOps {
    struct sceModemOps *forw, *back;                          Link to higher layer
    char *module_name                                         Module name
    char *vendor_name;                                        (Device's) vendor name
    char *device_name;                                        (Device's) device name
    u_char bus_type;                                          Bus type
    u_char bus_loc[31];                                       Location of bus information
    u_short prot_ver;                                         Protocol version
    u_short impl_ver;                                         Implementation version
    void *priv;                                               Modem layer pointer
    int evfid;                                                Event flag ID
    int rcv_len;                                              Receivable byte count
    int snd_len;                                              Sendable byte count
    int (*start)(void *priv, int flags);                      Start usage function
    int (*stop)(void *priv, int flags);                       Stop usage function
    int (*recv)(void *priv, void *ptr, int len);              Receive function
    int (*send)(void *priv, void *ptr, int len);              Send function
    int (*control)(void *priv, int code, void *ptr, int len); Control function
    void *reserved[4];                                        Reserved area
} sceModemOps_t;
```

Description

The higher layer links (*forw, back*) are fields used by the PPP layer. Set them both to NULL during registration.

Set the module name (*module_name*) to the module name of the modem driver layer. Use the filename of the executable, without the ".irx" suffix.

The vendor name (*vendor_name*) and device name (*device_name*) are the vendor name and device name of the device to be handled by that modem driver. A string should be specified even when a virtual device is used.

The *module_name, vendor_name*, and *device_name* must satisfy the following conditions.

- They must not be NULL.
- They must be terminated by NUL ('\0').
- They must not contain ',' or '='.
- They must not be empty ("").

There is no specific limit on string length.

bus type (bus_type) can be set to any of the following values during registration.

Table 3-1

Constant	Value	Meaning
sceModemBus_Unknown	0	Unknown. (This setting is not recommended)
sceModemBus_USB	1	USB device
sceModemBus_1394	2	(Reserved)
sceModemBus_PCMCIA	3	(Reserved)
sceModemBus_PSEUDO	4	Pseudo-device

The bus location information (*bus_loc*) is defined only for USB devices in the current specification.

During registration, the 7 bytes obtained with the sceUsbdGetDeviceLocation() function should be stored starting at the beginning of bus_loc.

The PPP layer manages configuration information for each device based on a combination of *module_name*, *vendor_name*, *device_name*, *bus_type*, and *bus_loc*.

bus_type and *bus_loc* are used for identification purposes when more than one device having exactly the same vendor_name and device_name are connected at the same time. Set these values even if the modem driver layer does not support multiple devices.

The protocol version (*prot_ver*) is for use when the modem interface specification is extended in the future. It should be set with the version of the modem interface specification assumed by the modem driver layer. 0 should be used for the current specification.

The implementation version (*impl_ver*) should contain the modem layer implementation version for each protocol version. Setting a sequence number starting with 0 that is incremented each time the modem layer implementation changes is recommended.

The modem layer pointer (*priv*) is a pointer to a data structure that the modem driver layer will use for each modem device. The value of this field is passed in the *priv* argument for each of the start, stop, recv, and send functions. The PPP layer doesn't use this value, so the modem layer is free to use it as desired.

The event flag ID (*evfid*), is set by the PPP layer and stores the ID of the event flag that was generated by the PPP layer. The modem driver layer uses this field to report state changes to the PPP layer.

The receivable byte count (*rcv_len*) and sendable byte count (*snd_len*) should both be initialized to 0 in advance.

The reserved area (*reserved*) is an area used by the PPP layer. This area need not be initialized, and its contents must not be referenced or changed.

Modem Control Functions

sceModemRegisterDevice

Registration function

Library	*Introduced*	*Documentation last modified*
modem	2.4	October 11, 2001

Syntax

#include < inet/modem.h>

int sceModemRegisterDevice(

sceModemOps_t ***ops*****);** Pointer to modem information structure

Description

When the modem driver layer has confirmed a connection with a new modem, it registers modem information and processing functions for the PPP layer. Also, when the connection with the modem is broken, the registration is deleted.

After the modem driver layer has confirmed that the modem has connected, it allocates memory for the modem information structure and sets the required fields. The memory area for storing this structure must be maintained until the registration is deleted.

This function is implemented in the PPP layer.

Return value

0 Normal termination

<0 Error (the value may vary with future implementations)

sceModemUnregisterDevice

Deletion function

Library	*Introduced*	*Documentation last modified*
modem	2.4	October 11, 2001

Syntax

#include <inet/modem.h>

int sceModemUnregisterDevice(

sceModemOps_t ****ops*);** Pointer to modem information structure

Description

When the modem connection is broken, the modem driver layer deletes the registration by calling this function, which is implemented in the PPP layer:

```
int sceModemUnregisterDevice(struct sceModemOps *ops);
```

A pointer to the same modem information structure that was specified during registration should be specified for the argument.

The registration can be deleted only when the modem is in not-in-use state. If the modem connection were broken when the modem was in in-use state, an event flag must be used to report the state change to the PPP layer, and the registration must be deleted by the stop usage function that gets called as a result.

If the modem connection were broken when the modem was registered and its state was not-in-use, the PPP layer does not need to be notified, and the modem registration can be deleted immediately by calling sceModemUnregisterDevice().

Return value

0 Normal termination

<0 Error (the value may vary with future implementations)

Modem Driver Layer Functions

control

Control function

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

int control(

void **priv,*	Pointer to private data (priv of modem information structure)
int *code,*	Control code
void **ptr,*	Starting address of data area
int *len***);**	Size of data area

Calling conditions

Can be called from a thread.

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

Description

This function is a control function that performs controls related to the operation of the modem driver layer, according to the specified control code (*code*).

Control codes are divided as follows into common control codes that do not depend on the modem driver layer implementation and control codes that are determined by the individual modem driver layer. Other values must not be passed to the modem driver layer.

Table 3-2

Value	Meaning
0xc0000000 - 0xcfffffff	Modem driver layer common control codes
0xd0000000 - 0xffffffff	Control codes determined for each modem driver layer implementation

The data area (*ptr, len*) is used when the control is accompanied by a data transfer. When there is no accompanying data transfer, the control function should be called with *ptr*=NULL and *len*=0. However, the modem driver layer is responsible for determining whether or not this check is performed.

If the size of the data to be transferred is zero or a positive value not exceeding sizeof(int), it may be returned as the return value of the control function without using the data area. The method that is used is determined by the control code.

Control codes that depend on the modem driver layer implementation may be freely defined by the person who implements the modem driver layer. Note that a control due to an implementation-dependent control code is performed only by a program that completely understands the operating specifications of that modem layer such as a modem driver installation program, for example.

Unlike other functions of the modem driver layer, the value returned when an error occurs in the control function is limited to the range from -600 to -649. As long as it is in this range, the value may be freely defined. (Zero and positive numbers can be returned if no error occurs.)

Notes

When each function is called, no special processing is performed on the PPP side in relation to the $gp register. If the $gp register is to be used, make sure that holding, setting, and return processing are performed by the called function.

Return value

When processing terminates normally, zero is returned. When an error occurs, an error code (-600 to -649) is returned.

recv

Receive function

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

#include <modem.h>

int recv(

void **priv,*	Pointer to private data (priv of modem information structure)
void **ptr,*	Starting address of receive buffer
int *len***);**	Size of receive buffer (bytes)

Calling conditions

Can be called from a thread.

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

Description

When sceModemEFP_Recv is reported from the modem driver layer, the PPP layer calls the receive function.

The receive function transfers data that was received from the modem device to the specified area (*ptr, len*). The amount of data to be transferred will be the smaller of the specified byte count (*len*) and the receivable byte count (*rcv_len* member of the modem information structure), and the number of transferred bytes will be returned as the function value.

Notes

When each function is called, no special processing is performed on the PPP side in relation to the $gp register. If the $gp register is to be used, make sure that holding, setting, and return processing are performed by the called function.

Return value

The number of transferred bytes is returned.

send

Send function

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

#include <modem.h>

int send(

void **priv,*	Pointer to private data (priv of modem information structure)
void **ptr,*	Starting address of send data
int *len***);**	Size of send data (bytes)

Calling conditions

Can be called from a thread.

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

Description

When sceModemEFP_Send is reported from the modem driver layer, the PPP layer references the snd_len member of the modem information structure. If the PPP layer considers that value to be a sufficiently large amount of data for starting transmission, it calls the send function. (If the PPP layer considers this value to be insufficient, it will wait for another sceModemEFP_Send to be reported.)

This send function transfers the send data from the specified area (*ptr, len*) to a buffer within the modem driver layer. The amount of data to be transferred will be the smaller of the specified byte count (*len*) and the sendable byte count (snd_len member of the modem information structure). Then, after setting snd_len to the size of free space in the buffer after the transfer, the number of transferred bytes will be returned as the function value.

Notes

When each function is called, no special processing is performed on the PPP side in relation to the $gp register. If the $gp register is to be used, make sure that holding, setting, and return processing are performed by the called function.

Return value

The number of transferred bytes is returned.

start

Start use function

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
#include <modem.h>
int start(
 void *priv,
 int flags);
```

priv — Pointer to private data (priv of modem information structure)

flags — Extension flag (currently, always 0)

Calling conditions

Can be called from a thread.

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

Description

The start use function performs processing such as initializing the modem device. The PPP layer calls the start use function when a start request is delivered to that modem from a higher layer. If the start use function returns zero, the PPP layer considers that the modem state is in-use.

As long as initialization processing has started, control may return before the start use function is completed (before there is an initialization completion response from the modem device). A notification should be reported to the PPP layer with the sceModemEFP_StartDone event flag when start use processing is completed.

flags is an argument that is reserved for future extensions. Currently, zero is passed in this argument.

Notes

When each function is called, no special processing is performed on the PPP side in relation to the $gp register. If the $gp register is to be used, make sure that holding, setting, and return processing are performed by the called function.

Return value

If processing terminates normally, zero is returned. If an error occurs, an error code is returned.

stop

Stop use function

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
#include <modem.h>
int stop(
 void *priv,          Pointer to private data (priv of modem information structure)
 int flags);          Extension flag (currently, always 0)
```

Calling conditions

Can be called from a thread.

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

Description

The PPP layer calls the stop use function for the modem in the following situations. This will cause the modem state to become not-in-use.

- When a line disconnection is explicitly directed from a higher layer than the PPP layer.
- When an automatic disconnection of the line is generated due to an internal timer of the PPP layer.
- When sceModemEFP_PlugOut is reported from the modem driver layer as a notification that data can no longer be exchanged with the modem.
- When sceModemEFP_Disconnect is reported from the modem driver layer as a notification of a line disconnection.

The stop use function performs processing such as disconnecting the line, switching from data mode to command mode, and reconfiguring the send/receive buffer. Wait processing such as waiting for a response from the modem is permitted only for this stop use function. If the connection with the modem had been broken, sceModemUnregisterDevice() is called to delete the modem registration.

flags is an argument that is reserved for future extensions. Currently, zero is passed in this argument.

Notes

When each function is called, no special processing is performed on the PPP side in relation to the $gp register. If the $gp register is to be used, make sure that holding, setting, and return processing are performed by the called function.

Return value

If processing terminates normally, zero is returned. If an error occurs, an error code is returned.

Modem Driver Layer Common Control Codes

sceModemCC_FLUSH_RXBUF

Clear recieve buffer

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                    Pointer to private data (priv of modem information structure)
 sceModemCC_FLUSH_RXBUF,
 // 0xc0000110
 void *ptr,                     Starting address of data area (unused)
 int len);                      Size of data area (unused)
```

Description

This control code clears the receive buffer within the modem driver layer. Whether or not it clears a buffer within an interface chip depends on the implementation.

Return value

Any of the following values may be returned.

Table 3-3

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_FLUSH_TXBUF

Clear transmit buffer

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information structure)
 sceModemCC_FLUSH_TXBUF,
 // 0xc0000111
 void *ptr,                           Starting address of data area (Unused)
 int len);                            Size of data area (Unused)
```

Description

This control code clears the transmit buffer within the modem driver layer. Whether or not it clears a buffer within an interface chip depends on the implementation.

Return value

Any of the following values may be returned.

Table 3-4

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_DIALCONF

Get dialing definition file path name

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

int control(

void **priv,*	Pointer to private data (priv of modem information structure)
sceModemCC_GET_DIALCONF, // 0xc0000200	
void **ptr,*	Starting address of data area
int *len***);**	Size of data area (bytes)

Description

This control code transfers the path name of the modem's dialing definition file to the area specified by *ptr* and *len*.

If *len* is not large enough to store the path name, an error occurs and sceINETE_INVALID_ARGUMENT is returned.

Return value

Any of the following values may be returned.

Table 3-5

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_IF_TYPE

Get device type

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information structure)
 sceModemCC_GET_IF_TYPE,
 // 0xc0000100
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code transfers the interface type that is handled by the modem driver layer to the u_int area indicated by *ptr* and *len* (=4). The interface type is represented by the following values.

Table 3-6

Constant	Value	Interface Type
sceModemIFT_GENERIC	0x00000000	Other than sceModemIFT_SERIAL
sceModemIFT_SERIAL	0x00000001	Modem connection via RS-232C

sceModemIFT_SERIAL corresponds to cases when the hardware is some kind of device that is handled from the modem driver layer as an RS-232C device for which the RS-232C level speed and other properties can be set, and signal lines can be controlled, with a "serial device control code" which is shown separately. sceModemIFT_GENERIC corresponds to all other cases.

If *len* is not sizeof(u_int) (=4), an error occurs.

Return value

Any of the following values may be returned.

Table 3-7

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_RX_COUNT

Get receive data size

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information structure)
 sceModemCC_GET_RX_COUNT,
 // 0xc0010000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code transfers the total number of bytes of data that the modem driver layer received from the device to the u_int area indicated by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Return value

Any of the following values may be returned.

Table 3-8

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_THPRI

Get thread priority

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information
                                      structure)
 sceModemCC_GET_THPRI,  // 0xc0000000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code transfers the priority of the thread that is to be created or was created by the modem driver layer to the u_int area indicated by *ptr* and *len* (=4).

When the modem driver layer creates multiple threads and different priorities must be assigned to them, the priority that is to be the base for the different priorities is handled with sceModemCC_GET_THPRI. The method by which the other priorities are determined from the base priority depends on the implementation.

If *len* is not sizeof(u_int) (=4), an error occurs.

Return value

Any of the following values may be returned.

Table 3-9

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_TX_COUNT

Get transmit data size

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                                  Pointer to private data (priv of modem
                                              information structure)
 sceModemCC_GET_TX_COUNT,  // 0xc0010001
 void *ptr,                                   Starting address of data area
 int len);                                    Size of data area (sizeof(u_int))
```

Description

This control code transfers the number of bytes of data that the modem driver layer sent to the device to the u_int area indicated by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Return value

Any of the following values may be returned.

Table 3-10

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_SET_THPRI

Set thread priority

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem
                                      information structure)
 sceModemCC_SET_THPRI,  // 0xc1000000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code considers the int value that was set in the area indicated by *ptr* and *len* (=4) as the priority of modem driver layer threads and changes the priority of threads to be created by the modem driver layer to that value.

For a thread that has already been activated, ChangeThreadPriority() is used to immediately change its priority. For a thread that has been created but has not yet been activated (that is, a DORMANT state thread), the priority is changed after the thread is activated.

When no thread priority has been set with sceModemCC_SET_THPRI, the default value is implementation-dependent.

If *len* is not sizeof(u_int) (=4), an error occurs.

Return value

Any of the following values may be returned.

Table 3-11

Value	Meaning
0 or positive number	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

Modem Driver Layer Common Control Codes (for serial devices)

sceModemCC_GET_BO_COUNT

Get buffer overflow count

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

int control(

void **priv,*	Pointer to private data (priv of modem information structure)
sceInetNDCC_GET_THPRI, // 0x80000000	
void **ptr,*	Starting address of data area
int *len***);**	Size of data area (sizeof(u_int))

Description

This control code stores the total number of bytes of receive data for which a buffer overflow occurred in the receive buffer that is in the modem layer (and is not on the serial interface chip) as a u_int value in the area specified by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Notes

The counter value is initialized to 0 only when the modem device is registered.

When the counter value causes a u_int overflow, whether it stays at 0xffffffff and is no longer incremented or is returned to 0 and continues to be incremented depends on the modem layer implementation.

Return value

Any of the following values may be returned.

Table 3-12

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_FE_COUNT

Get framing error count

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem
                                      information structure)
 sceModemCC_GET_FE_COUNT,  // 0xc0010004
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code stores the total number of times that framing errors were detected during receive as a u_int value in the area specified by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Notes

The counter value is initialized to 0 only when the modem device is registered.

When the counter value causes a u_int overflow, whether it stays at 0xffffffff and is no longer incremented or is returned to 0 and continues to be incremented depends on the modem layer implementation.

Return value

Any of the following values may be returned.

Table 3-13

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_LINE

Get state of each signal line

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information
                                      structure)
 sceModemCC_GET_LINE,  // 0xc0030000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code gets the state of each signal line and stores it as a u_int value in the area indicated by *ptr* and *len* (=4), after taking its logical OR with the following bit flags.

Table 3-14

Constant	Meaning
sceModemLINE_CTS	CTS (in)
sceModemLINE_DSR	DSR (in)
sceModemLINE_RI	RI (in)
sceModemLINE_DCD	DCD (in)
sceModemLINE_DTR	DTR (out)
sceModemLINE_RTS	RTS (out)

If *len* is not sizeof(u_int) (=4), an error occurs.

Notes

If the output signal cannot be read out by an exchange with the serial interface chip, the modem layer must always obtain the state of each signal line in advance.

Return value

Any of the following values may be returned.

Table 3-15

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_OE_COUNT

Get overrun error count

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem
                                      information structure)
 sceModemCC_GET_OE_COUNT,  // 0xc0010002
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code stores the total number of times that overrun errors were detected in receive registers (including FIFOs) on the serial interface chip as a u_int value in the area specified by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Notes

The counter value is initialized to 0 only when the modem device is registered.

When the counter value causes a u_int overflow, whether it stays at 0xffffffff and is no longer incremented or is returned to 0 and continues to be incremented depends on the modem layer implementation.

Return value

Any of the following values may be returned.

Table 3-16

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_PARAM

Get serial communication parameter

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information
                                      structure)
 sceModemCC_GET_PARAM,  // 0xc0020000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code gets a serial communication parameter such as the speed or parity mode and stores it as a u_int value in the area indicated by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Serial communication parameters are represented by the following bit fields.

Table 3-17

Constant	Value	Meaning
sceModemPARAM_SPEED	0x003fffff	Communication speed (bps) mask
sceModemPARAM_RESERVED	0x00400000	(Reserved)
sceModemPARAM_XON	0x00800000	XON/XOFF flow control enabled (reception)
sceModemPARAM_XOFF	0x01000000	XON/XOFF flow control enabled (transmission)
sceModemPARAM_RTSCTS	0x02000000	RTS/CTS flow control enabled
sceModemPARAM_STOPS	0x0c000000	Stop-bit bit mask
sceModemPARAM_STOP0	0x00000000	Stop 0 bits
sceModemPARAM_STOP1	0x04000000	Stop 1 bits
sceModemPARAM_STOP1H	0x08000000	Stop 1.5 bits
sceModemPARAM_STOP2	0x0c000000	Stop 2 bits
sceModemPARAM_CSIZE	0x30000000	Data-length bit mask
sceModemPARAM_CS5	0x00000000	5 bit/char
sceModemPARAM_CS6	0x10000000	6 bit/char
sceModemPARAM_CS7	0x20000000	7 bit/char
sceModemPARAM_CS8	0x30000000	8 bit/char
sceModemPARAM_PARODD	0x40000000	Odd parity
sceModemPARAM_PARENB	0x80000000	Parity enabled

The communication speed represents the communication speed between the modem layer and device, not the speed on the line. Values are in bps.

When sceModemPARAM_XON is 1, it indicates that a function is enabled so that when an XON/XOFF code is received, it will be handled as a control character representing a transmission stop/restart request, not as normal receive data.

When sceModemPARAM_XOFF is 1, it indicates that a function is enabled for requesting that transmission be stopped/restarted by sending XON/XOFF codes according to the amount of free space in the receive buffer. The timing for sending the XON/XOFF codes depends on the modem layer implementation.

When sceModemPARAM_RTSCTS is 1, it indicates that an RTS/CTS flow control function is enabled, which controls RTS as a function of the receive buffer state and stops send operations as a function of CTS state.

sceModemPARAM_STOPS represents stop bits. The following constants are defined for the stop bits.

Table 3-18

Constant	Meaning
sceModemPARAM_STOP0	0 stop bits
sceModemPARAM_STOP1	1 stop bit
sceModemPARAM_STOP1H	1.5 stop bits
sceModemPARAM_STOP2	2 stop bits

sceModemPARAM_CSIZE represents the character size. The following constants are defined for the character size.

Table 3-19

Constant	Meaning
sceModemPARAM_CS5	5 bits/char
sceModemPARAM_CS6	6 bits/char
sceModemPARAM_CS7	7 bits/char
sceModemPARAM_CS8	8 bits/char

The combination of sceModemPARAM_PARODD and sceModemPARAM_PARENB represents the parity mode. Only the following fixed parities are supported.

Table 3-20

Parity Mode	sceModemPARAM_PARODD	sceModemPARAM_PARENB
No parity	0	0
Even parity	0	1
Odd parity	1	1

Notes

Although the ability to obtain the serial communication parameter regardless of the start/stop state is recommended, this ability depends on the modem layer implementation. If it is not supported, sceINETE_INVALID_CALL is returned.

Return value

Any of the following values may be returned.

Table 3-21

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_GET_PE_COUNT

Get parity error count

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem
                                      information structure)
 sceModemCC_GET_PE_COUNT,  // 0xc0010003
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code stores the total number of times that parity errors were detected during receive as a u_int value in the area specified by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

Notes

The counter value is initialized to 0 only when the modem device is registered.

When the counter value causes a u_int overflow, whether it stays at 0xffffffff and is no longer incremented or is returned to 0 and continues to be incremented depends on the modem layer implementation.

Return value

Any of the following values may be returned.

Table 3-22

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_SET_BREAK

Send break

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information
                                      structure)
 sceModemCC_SET_BREAK,  // 0xc1040000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code treats the u_int value in the area indicated by *ptr* and *len* (=4) as a time interval in ms and generates a break condition for that time interval.

If *len* is not sizeof(u_int) (=4), an error occurs.

Return value

Any of the following values may be returned.

Table 3-23

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4, ms is greater than or equal to 0xffff)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_SET_LINE

Control each signal line

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information
                                      structure)
 sceModemCC_SET_LINE,  // 0xc1030000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code sets the output value of each signal line according to the u_int value in the area indicated by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

For the correspondence between the argument bit flags and the signal lines, see the description of sceModemCC_GET_LINE.

Notes

A request to set an input signal is simply ignored.

If RTS/CTS flow control had been enabled, the processing that is performed when there is a request to change the output value of the RTS signal depends on the modem layer. (Simply ignoring the request is recommended.)

Return value

Any of the following values may be returned.

Table 3-24

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

sceModemCC_SET_PARAM

Set serial communication parameter

Library	*Introduced*	*Documentation last modified*
modem	2.2	March 26, 2000

Syntax

```
int control(
 void *priv,                          Pointer to private data (priv of modem information
                                      structure)
 sceModemCC_SET_PARAM,  // 0xc1020000
 void *ptr,                           Starting address of data area
 int len);                            Size of data area (sizeof(u_int))
```

Description

This control code sets a serial communication parameter such as the speed according to the u_int argument that is in the area specified by *ptr* and *len* (=4).

If *len* is not sizeof(u_int) (=4), an error occurs.

For information about serial communication parameter specifications, see the description of sceModemCC_GET_PARAM.

Notes

Although the ability to set the serial communication parameter regardless of the start/stop state is recommended, this ability depends on the modem layer implementation. If it is not supported, sceINETE_INVALID_CALL is returned.

The default settings of the serial communication parameters depend on the modem layer implementation. It is recommended that default values be set that are as appropriate as possible with the intent that send/receive should be performed without setting parameters.

Return value

Any of the following values may be returned.

Table 3-25

Value	Meaning
0	Normal termination
sceINETE_INVALID_ARGUMENT	Invalid argument (*len* is not 4, a parameter that cannot be set was assigned)
sceINETE_INVALID_CALL	Invalid control code
-600 to –649	Modem layer-dependent error code

Chapter 4: Common Network Configuration Library
Table of Contents

Configuration File Structures

sceNetCnfAddress

Internal format IP address

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

```
typedef struct sceNetCnfAddress {
    int reserved;          Reserved area (always 0)
    char data[16];         IP address
} sceNetCnfAddress_t;
```

Description

This is a structure for maintaining an IP address within the library.

The current implementation only supports IPv4. To prepare for future extensions, a user program must not directly access the internal structure. sceNetCnfName2Address() and sceNetCnfAddress2String() should be used instead.

sceNetCnfCommand

Routing configuration information

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

```
typedef struct sceNetCnfCommand {
    struct sceNetCnfCommand *forw;          Forward link
    struct sceNetCnfCommand *back;          Backward link
    int code;                               Command code
    // sceNetCnfAddress_t address;          /* {ADD,DEL}_NAMESERVER */
    // sceNetCnfRoutingEntry_t routing;     /* {ADD,DEL}_ROUTING */
} sceNetCnfCommand_t;
```

Description

This is a data structure that corresponds to the nameserver and route keywords of an ATTACH_CNF file. netcnf.irx reads and interprets the configuration file then maintains the data in memory as this structure.

The command code (code) can be any of the following.

Table 4-1

Command Code	Keyword
sceNetCnf_CMD_ADD_NAMESERVER	nameserver add
sceNetCnf_CMD_DEL_NAMESERVER	nameserver del
sceNetCnf_CMD_ADD_ROUTING	route add
sceNetCnf_CMD_DEL_ROUTING	route del

The nameserver address (sceNetCnfAddress_t type) or routing information (sceNetCnfRoutingEntry_t type) that is to be added or deleted is placed immediately after the sceNetCnfCommand_t object.

See also

sceNetCnfInterface

sceNetCnfCtl

Configuration control information

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

typedef struct sceNetCnfCtl {

struct sceNetCnfDial **dial*	Pointer to dialing definition data
struct sceNetCnfInterface **ifc;*	Pointer to interface definition data
int *id;*	Interface ID
int *phone_index;*	Phone number index currently being referenced
int *redial_index;*	Current redial count
char *interface***[8 + 1];**	Interface name

} sceNetCnfCtl_t;

Description

This is a data structure for configuration processing that is used internally by netcnf.irx.

sceNetCnfDial

Dialing definition information

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

typedef struct sceNetCnfDial {

u_char **tone_dial;*	dialing_type_string for tone line
u_char **pulse_dial;*	dialing_type_string for pulse line
u_char **any_dial;*	dialing_type_string for other line
u_char **chat_init;*	chat_init script string
u_char **chat_dial;*	chat_dial script string
u_char **chat_answer;*	chat_answer script string
u_char **redial_string;*	redial_string result string
struct sceNetCnfUnknownList *unknown_list;*	List of data structures for storing undefined keywords and arguments

} sceNetCnfDial_t;

Description

This is a data structure that corresponds to one DIAL_CNF file. netcnf.irx reads and interprets a DIAL_CNF file, then maintains it in memory as this data structure.

sceNetCnfEnv

Load/save environment

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

typedef struct sceNetCnfEnv {

char **dir_name;*	Pathname on which relative path processing is based
char **arg_fname;*	Filename to be manipulated
void **mem_base;*	Starting address of memory area
void **mem_ptr;*	Address to be used next in memory area
void **mem_last;*	Memory area last byte + 1
int *req;*	Request code
struct sceNetCnfRoot **root;*	Pointer to data structure corresponding to NET_CNF file
struct sceNetCnfInterface **ifc;*	Pointer to data structure corresponding to ATTACH_CNF file
int *f_no_check_magic;*	Whether or not to check magic line
int *f_no_decode;*	Whether or not to encode/decode ATTACH_CNF
int *f_verbose;*	Whether or not to display verbose messages
int *file_err;*	Number of times errors occurred when opening, reading, or writing file
int *alloc_err;*	Number of times memory allocation failed
int *syntax_err;*	Number of times syntax errors were detected
char **fname;*	(Internal processing work area)
int *lno;*	(Internal processing work area)
u_char *lbuf***[1024];**	(Internal processing work area)
u_char *dbuf***[1024];**	(Internal processing work area)
int *ac;*	(Internal processing work area)
u_char **av***[10 + 1];**	(Internal processing work area)

} sceNetCnfEnv_t;

Description

This is a data structure that is used as a data passing area or work area when a configuration file is read by sceNetCnfLoadEntry() or written by saveNetCnfAddEntry().

Since the structure includes settable work areas, calls can be safely made even from a multithreaded program.

See also

sceNetCnfLoadEntry(), sceNetCnfAddEntry()

sceNetCnfInterface

Configuration information for each interface

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Structure

```
typedef struct sceNetCnfInterface {
    int type;
    u_char *vendor;
    u_char *product;
    u_char *location;
    u_char dhcp;
    u_char *dhcp_host_name;
    u_char dhcp_host_name_null_terminated;
    u_char dhcp_release_on_stop;
    u_char *address;
    u_char *netmask;
    u_char *chat_additional;
    int redial_count;
    int redial_interval;
    u_char *outside_number;
    u_char *outside_delay;
    u_char *phone_numbers
    [sceNetCnf_MAX_PHONE_NUMBERS];
    u_char answer_mode;
    int answer_timeout;
    int dialing_type;
    u_char *chat_login;
    u_char *auth_name;
    u_char *auth_key;
    u_char *peer_name;
    u_char *peer_key;
    int lcp_timeout;
    int ipcp_timeout;
    int idle_timeout;
    int connect_timeout;
    struct {
            u_char mru_nego;
            u_char accm_nego;
            u_char magic_nego;
            u_char prc_nego;
            u_char acc_nego;
            u_char address_nego;
```

```
            u_char vjcomp_nego;
            u_char dns1_nego;
            u_char dns2_nego;
            u_char reserved_nego[7];
            u_short mru;
            u_long accm;
            u_char auth;
            u_char f_mru;
            u_char f_accm;
            u_char f_auth;
            u_char *ip_mask;
            u_char *dns1;
            u_char *dns2;
            u_long reserved_value[8];
        } want, allow;
        int log_flags;
        u_char force_chap_type;
        u_char omit_empty_frame;
        u_char pppoe;
        u_char pppoe_host_uniq_auto;
        u_char pppoe_reserved[2];
        u_char *pppoe_service_name;
        u_char *pppoe_ac_name;
        u_int mtu;
        u_long reserved[3];
        int phy_config;
        struct sceNetCnfCommand *cmd_head;
        struct sceNetCnfCommand *cmd_tail;
        struct sceNetCnfUnknownList unknown_list;
} sceNetCnfInterface_t;
```

Description

This is a structure for maintaining configuration information related to one interface. netcnf.irx reads and interprets an ATTACH_CNF file, then maintains it in memory as this data structure.

The following table shows the correspondence between various members of this structure and keywords within ATTACH_CNF.

Table 4-2

Member Name	Corresponding Keyword in ATTACH_CNF	Data Type
type	type	number4
vendor	vendor	string
product	product	string
location	location	string
dhcp	dhcp	bool
dhcp_host_name	dhcp_host_name	string
dhcp_host_name_ null_terminated	dhcp_host_name_null_ter minated	bool
dhcp_release_on_stop	dhcp_release_on_stop	bool
address	address	string
netmask	netmask	string
chat_additional	chat_additional	string
redial_count	redial_count	number4
redial_interval	redial_interval	number4
outside_number	outside_number	string
outside_delay	outside_delay	string
phone_numbers	phone_number[0..9]	string
answer_mode	answer_mode	bool
answer_timeout	answer_timeout	number4
dialing_type	dailing_type	number4
chat_login	chat_login	string
auth_name	auth_name	string
auth_key	auth_key	string
peer_name	peer_name	string
peer_key	peer_key	string
lcp_timeout	lcp_timeout	number4
ipcp_timeout	ipcp_timeout	number4
idle_timeout	idle_timeout	number4
connect_timeout	connect_timeout	number4
want.mru_nego	want.mru_nego	bool
want.accm_nego	want.accm_nego	bool
want.magic_nego	want.magic_nego	bool
want.prc_nego	want.prc_nego	bool
want.acc_nego	want.acc_nego	bool
want.address_nego	want.address_nego	bool
want.vjcomp_nego	want.vjcomp_nego	bool
want.dns1_nego	want.dns1_nego	bool
want.dns2_nego	want.dns2_nego	bool
want.reserved_nego	(for future expansion)	
want.mru	want.mru	number2
want.accm	want.accm	number4

Member Name	Corresponding Keyword in ATTACH_CNF	Data Type
want.auth	want.auth	number1
want.f_mru	(1 if there is a want.mru setting, 0 if there is no setting)	number1
want.f_accm	(1 if there is a want.accm setting, 0 if there is no setting)	number1
want.f_auth	(1 if there is a want.auth setting, 0 if there is no setting)	number1
want.ip_address	want.ip_address	string
want.ip_mask	want.ip_mask	string
want.dns1	want.dns1	string
want.dns2	want.dns2	string
want.reserved_value	(for future expansion)	
allow.mru_nego	allow.mru_nego	bool
allow.accm_nego	allow.accm_nego	bool
allow.magic_nego	allow.magic_nego	bool
allow.prc_nego	allow.prc_nego	bool
allow.acc_nego	allow.acc_nego	bool
allow.address_nego	allow.address_nego	bool
allow.vjcomp_nego	allow.vjcomp_nego	bool
allow.dns1_nego	allow_dns1_nego	bool
allow.dns2_nego	allow.dns2_nego	bool
allow.reserved_nego	(for future expansion)	number2
allow.mru	allow.mru	number4
allow.accm	allow.accm	number1
allow.auth	allow.auth	number 1
allow.f_mru	(1 if there is an allow.mru setting, 0 if there is no setting)	number1
allow.f_accm	(1 if there is an allow.accm setting, 0 if there is no setting)	number1
allow.f_auth	(1 if there is an allow.auth setting, 0 if there is no setting)	number1
allow.ip_address	allow.ip_address	string
allow.ip_mask	allow.ip_mask	string
allow.dns1	allow.dns1	string
allow.dns2	allow.dns2	string
allow.reserved_value	(for future expansion)	
log_flags	log_flags	number4

Member Name	Corresponding Keyword in ATTACH_CNF	Data Type
force_chap_type	force_chap_type (sceNetCnf_BOOL_DEFAULT=0xff when there is no setting)	number1
omit_empty_frame	omit_empty_frame	bool
pppoe	pppoe	bool
pppoe_host_uniq_auto	pppoe_host_uniq_auto	bool
pppoe_reserved	(for future expansion)	
pppoe_service_name	pppoe_service_name	string
pppoe_ac_name	pppoe_ac_name	string
mtu	mtu	number4
reserved	(for future expansion)	
phy_config	phy_config	number4
cmd_head	nameserver / route (Pointer to beginning of bidirectional queue)	
cmd_tail	nameserver / route (Pointer to end of bidirectional queue)	
unknown_list	List of undefined keywords and arguments)	

The following represent the entries in the Data Type column.

Table 4-3

Data Type	Contents
string	String. NULL when there is no setting
bool	Boolean value. 0xff (sceNetCnf_BOOL_DEFAULT) when there is not setting
number1	1-byte numeric value
number2	2-byte numeric value
number4	4-byte numeric value. -1 when there is no setting

The correspondence between the numeric values of each member and the keyword arguments is shown below. For some members, the strings corresponding to the numeric values are defined in netcnf.h.

type can have any of the following values.

Table 4-4

String	Value	Argument	Meaning
sceNetCnf_IFC_TYPE_ANY	0		Type of lower layer unspecified [default]
sceNetCnf_IFC_TYPE_ETH	1	eth	Supports USB-Ethernet
sceNetCnf_IFC_TYPE_PPP	2	ppp	Supports PPP connection
sceNetCnf_IFC_TYPE_NIC	3	nic	Supports Ethernet (Network Adaptor)

dialing_type can have any of the following values.

Table 4-5

String	Value	Argument	Meaning
sceNetCnf_DIALING_TYPE_DEFAULT	-1		Not specified
sceNetCnf_DIALING_TYPE_TONE	0	tone	Tone line (analog) [default]
sceNetCnf_DIALING_TYPE_PULSE	1	pulse	Pulse line (analog)
sceNetCnf_DIALING_TYPE_ANY	2	any	Other line (such as digital)

phy_config can have any of the following values.

Table 4-6

String	Value	Argument	Meaning
	0		Physical layer chip configuration method not specified
sceNetCnf_PHYCONFIG_AUTO	1	phy_config auto	Auto Negotiation Mode
sceNetCnf_PHYCONFIG_10	2	phy_config 10	10BaseT, Half-Duplex
sceNetCnf_PHYCONFIG_10_FD	3	phy_config 10_fd	10BaseT, Full-Duplex, No-Flow-Control
sceNetCnf_PHYCONFIG_10_FD_PAUSE	4	phy_config 10_fd_pause	10BaseT, Full-Duplex, Flow-Control
sceNetCnf_PHYCONFIG_TX	5	phy_config tx	100BaseTX, Half-Duplex
sceNetCnf_PHYCONFIG_TX_FD	6	phy_config tx_fd	100BaseTX, Full-Duplex, No-Flow-Control
sceNetCnf_PHYCONFIG_TX_FD_PAUSE	7	phy_config tx_fd_pause	100BaseTX, Full-Duplex, Flow-Control

want.auth and *allow.auth* can have any of the following values.

Table 4-7

Value	Argument	Meaning
0	any	Do not request PAP or CHAP authentication [default]
1	pap	Request only PAP authentication
2	chap	Request only CHAP authentication
3	pap/chap	First, request PAP authentication, and if the resulting connection is denied, request CHAP authentication
4	chap/pap	First, request CHAP authentication, and if the resulting connection is denied, request PAP authentication

force_chap_type can have any of the following values.

Table 4-8

Value	Argument	Meaning
-1		Do not limit the authentication algorithm [default] (force_chap_type keyword is not written to ATTACH_CNF)
0	no	Do not limit the authentication algorithm [default] (force_chap_type keyword and argument are written to ATTACH_CNF)
5	md5	Limited to MD5 only
0x80	ms	Limited to MS (Version 1) only
0x80	ms-v1	Limited to MS (Version 1) only (same as ms)
0x81	ms-v2	Limited to MS (Version 2) only

See also

sceNetCnfCtl, sceNetCnfPair, scenetCnfEnv, sceNetCnfInitIFC()

sceNetCnfList

Configuration management file data

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

typedef struct sceNetCnfList {

int *type;*	File type 0: Environment configuration file 1: Connection destination configuration file 2: Modem configuration file
int *stat;*	File status 0: Deleted (invalid file) 1: Valid file
char *sys_name***[256];**	Configuration filename assigned by system
char *usr_name***[256];**	Configuration filename assigned by user

} sceNetCnfList_t;

Description

This is a structure that corresponds to the various entries in the configuration management file. netcnf.irx reads and interprets the configuration file, then maintains the data in memory as this structure.

See also

sceNetCnfGetList()

sceNetCnfPair

interface keyword information

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

typedef struct sceNetCnfPair {

struct sceNetCnfPair **forw;*	Forward link
struct sceNetCnfPair **back;*	Backward link
u_char **display_name;*	Display name
u_char **attach_ifc;*	ifc filename
u_char **attach_dev;*	dev filename
struct sceNetCnfInterface **ifc;*	Pointer to interface definition data
struct sceNetCnfInterface **dev;*	Pointer to device definition data
struct sceNetCnfUnknownList *unknown_list;*	List of data undefined keywords and arguments
struct sceNetCnfCtl **ctl;*	Pointer to configuration control information

} sceNetCnfPair_t;

Description

This is a data structure that corresponds to a single, specific interface keyword that is in the NET_CNF file.

netcnf.irx reads and interprets the configuration file, then maintains the data in memory as this structure.

sceNetCnfRoot

NET_CNF information

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

```
typedef struct sceNetCnfRoot {
    struct sceNetCnfPair *pair_head;          Beginning of interface keyword data structure list
    struct sceNetCnfPair *pair_tail;          End of interface keyword data structure list
    int version;                              Data structure version
    u_char *chat_additional;                  chat_additional script string
    int redial_count;                         redial_count data
    int redial_interval;                      redial_interval data
    u_char *outside_number;                   outside_number data
    u_char *outside_delay;                    outside_delay data
    int dialing_type;                         dialing_type data
    struct sceNetCnfUnknownList unknown_list; List of undefined keywords and arguments
} sceNetCnfRoot_t;
```

Description

This is a data structure that corresponds to a single NET_CNF file. netcnf.irx reads and interprets the configuration file, then maintains the data in memory as this structure.

sceNetCnfRoutingEntry

Routing control table entry

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

```
typedef struct sceNetCnfRoutingEntry {
    struct sceNetCnfAddress dstaddr;      Destination address
    struct sceNetCnfAddress gateway;      Next POP router address
    struct sceNetCnfAddress genmask;      Subnet mask
    int flags;                            Flags indicating the state
    int mss;                              Maximum segment size
    int window;                           TCP window size
    char interface[8 + 1];                Network interface name
} sceNetCnfRoutingEntry_t;
```

Description

This is a structure for storing routing control table entry information.

The *flags* member contains the value obtained from the logical OR of the following bit flags.

Table 4-9

Constant	Value	Meaning
sceInetRoutingF_Up	0x01	Route is valid
sceInetRoutingF_Host	0x02	Direct delivery (not via a router)
sceInetRoutingF_Gateway	0x04	Indirect delivery (via a router)
sceInetRoutingF_Dynamic	0x08	Dynamically set
sceInetRoutingF_Modified	0x10	Same entry with modification

Although the maximum segment size (mss) and window size (window) can be set and referenced, those values currently are not used in NETCNF.

See also

sceNetCnfAddress

sceNetCnfUnknown

Undefined keyword and argument data

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

```
typedef struct sceNetCnfUnknown {
    struct sceNetCnfUnknown *forw;          Forward link
    struct sceNetCnfUnknown *back;          Backward link
    // u_char unknown_keyword_and_arguments[0];
} sceNetCnfUnknown_t;
```

Description

This is a structure for storing (currently) undefined keywords and arguments that will be added when the specifications are extended in the future. netcnf.irx reads and interprets a configuration file, then maintains the data in memory as this structure.

See also

sceNetCnfUnknownList

sceNetCnfUnknownList

Undefined keyword and argument list

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Structure

```
typedef struct sceNetCnfUnknownList {
    struct sceNetCnfUnknown *head;      Pointer to beginning of list
    struct sceNetCnfUnknown *tail;      Pointer to end of list
} sceNetCnfUnknownList_t;
```

Description

This is a data structure that indicates the beginning and end of a bidirectional queue for storing (currently) undefined keywords and arguments that will be added when the specifications are extended in the future. netcnf.irx reads and interprets a configuration file, then maintains the data in memory as this structure.

See also

sceNetCnfInterface

Configuration File Functions

sceNetCnfAddEntry

Add entry to configuration management file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfAddEntry(
 char *fname,                Pathname of configuration management file
 int type,                   File type
                               0: Connection environment configuration file
                               1: Connection configuration file
                               2: Modem configuration file
 char *usr_name,             Configuration name
sceNetCnfEnv_t *e);          Save environment
```

Calling conditions

Can be called from a thread.

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

Description

This function adds the entry specified by *type* and *usr_name* to the configuration management file, *fname*, expands the configuration data that was indicated by the save environment, e, in a text image, and saves the text image to the file.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

If the directory where the file will be saved does not exist, it will be created automatically and an icon and icon.sys file will be added. The directory contents are checked during a call and unnecessary files are deleted. If the icon and icon.sys have incorrect names or sizes, they will be corrected as well. The setting name is unconditionally set as shown below when type == 0.

Combination"index"

The following restrictions are placed on each target device for "index". If an "index" other than those listed below is specified, sceNETCNF_INVALID_USR_NAME will be returned.

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

The members that must be set in the save environment are mem_base and mem_last, which represent the text image expansion area. dir_name, arg_fname, and req are automatically set by processing within sceNetCnfAddEntry().

To add changes to the load environment where the configuration data was read, then save it as the save environment, set the following immediately before performing the save:

```
e->mem_base = e->mem_ptr;
```

Return value

0 <=	Normal termination
sceNETCNF_INVALID_USR_NAME	*usr_name* is invalid or name is already being used
sceNETCNF_INVALID_FNAME	*fname* is invalid
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_WRITE_ERROR	Error occurred when writing file
sceNETCNF_TOO_MANY_ENTRIES	Upper limit for number of entries given below was exceeded • PS2 Memory card Combinations 6 Hardware 4 Network service providers 4 • Hard disk drive Combinations 10 Hardware 30 Network service providers 30 (Upper limit of other devices is 1000 for each type of file.)
sceNETCNF_INVALID_TYPE	*type* is invalid
sceNETCNF_NG	Write to configuration file failed
sceNETCNF_CAPACITY_ERROR	Amount remaining is less than 94 Kbytes
sceNETCNF_IO_ERROR	I/O error occured

sceNetCnfAddress2String

Conversion from internal-format IP address to dot format

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfAddress2String(
 char *buf,                          Address of buffer where the conversion result will be stored
 int len,                            Buffer length (bytes)
 sceNetCnfAddress_t *paddr);         Internal-format IP address
```

Calling conditions

Can be called from a thread.

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

Description

This function converts an internal-format IP address to a dot-format string.

This function is used for display and debugging.

Return value

The starting address of the conversion result (=*buf*) is returned.

sceNetCnfAllocMem

Allocate memory area

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	July 2, 2001

Syntax

```
#include <netcnf.h>
void *sceNetCnfAllocMem(
 sceNetCnfEnv_t *e,          Load/save environment
 int size,                   Number of bytes of memory to be allocated
 int align);                 Alignment of beginning of memory area to be allocated
                               0: Byte alignment
                               2: Word alignment
```

Calling conditions

Can be called from a thread.

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

Description

This function allocates a memory area using the *size* and *align* specifications from the memory pool in the load or save environment specified by e.

When the memory is allocated, *e*->mem_ptr is updated. If allocation fails, *e*->alloc_err will be incremented.

Return value

!= NULL Allocation was successful

== NULL Allocation failed

sceNetCnfCheckCapacity

Check remaining capacity

Library	*Introduced*	*Documentation last modified*
netcnf	2.4.2	December 3, 2001

Syntax

#include <netcnf.h>

int sceNetCnfCheckCapacity(

char ****fname*);** Path name of configuration management file

Calling conditions

Can be called from a thread

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

Description

This function checks the remaining capacity of the device on which the configuration management file resides.

Return value

0 <=	Greater than or equal to 94K bytes
SceNETCNF_CAPACITY_ERROR	Less than 94K bytes

sceNetCnfDeleteAll

Delete all common network settings

Library	*Introduced*	*Documentation last modified*
netcnf	2.3	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfDeleteAll(
 char *dev);                    Device name (only "mc?:" and "pfs?:" are supported)
```

Calling conditions

Can be called from a thread

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

Description

Deletes the common network configuration present in the specified device in each directory.

If no common network configuration directory is present, 0 will be returned.

Return value

0	Normal end
sceNETCNF_REMOVE_ERROR	Delete failed
sceNETCNF_UNKNOWN_DEVICE	Unknown device
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfDeleteEntry

Delete entry from configuration management file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include < netcnf.h>
 int sceNetCnfDeleteEntry(
char *fname,                    Pathname of configuration management file
int type,                       File type
                                  0: Connection environment configuration file
                                  1: Connection configuration file
                                  2: Modem configuration file
char *usr_name);                Current configuration name
```

Calling conditions

Can be called from a thread.

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

Description

This function deletes the entries specified by *type* and *usr_name* from the configuration management file, *fname*, deletes the configuration files indicated in those entries, and returns the number of deleted entries.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

The directory contents are checked during a call and unnecessary files are deleted. If the icon and icon.sys have incorrect names or sizes, they will be corrected as well. The setting name is unconditionally set as shown below when type == 0.

Combination"index"

The following restrictions are placed on each target device for "index". If an "index" other than those listed below is specified, sceNETCNF_INVALID_USR_NAME will be returned.

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

Return value

0 <	Deletion was successful
sceNETCNF_INVALID_USR_NAME	*usr_name* was invalid

sceNETCNF_INVALID_FNAME	*fname* was invalid
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_WRITE_ERROR	Error occurred when writing file
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfEditEntry

Edit entry in configuration management file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfEditEntry(
```

char **fname,*	Pathname of configuration management file
int *type,*	File type 0: Connection environment configuration file 1: Connection configuration file 2: Modem configuration file
char **usr_name,*	Current configuration name
char **new_usr_name,*	Modified configuration name (NULL if unmodified)
sceNetCnfEnv_t **e*);	Save environment

Calling conditions

Can be called from a thread.

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

Description

This function edits the entry specified by *usr_name* in the configuration management file, *fname*, and saves the configuration data indicated by the save environment, *e*, to the file.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

The directory contents are checked during a call and unnecessary files are deleted. If the icon and icon.sys have incorrect names or sizes, they will be corrected as well. The setting name is unconditionally set as shown below when type == 0.

Combination"index"

The following restrictions are placed on each target device for "index". If an "index" other than those listed below is specified, sceNETCNF_INVALID_USR_NAME will be returned.

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

“index” must not be between 1 and 10.

Other

“index” must not be between 1 and 1000.

The members that must be set in the save environment are mem_base and mem_last. This memory area is used for saving a text image of the configuration file that is to be stored. Since the dir_name, arg_fname, and req members are automatically set by the function, they need not be specified.

Notes

To share the load environment and save environment, set the following immediately before performing the save:

```
e->mem_base = e->mem_ptr;
```

Return value

If processing terminates normally, a positive value is returned. If an error occurs, any one of the following error codes may be returned.

Table 4-10

Constant	Meaning
sceNETCNF_INVALID_USR_NAME	*usr_name* is invalid or *new_user_name* is the same as a configuration name that is already being used)
sceNETCNF_INVALID_FNAME	*fname* is invalid
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_WRITE_ERROR	Error occurred when writing file
sceNETCNF_ENTRY_NOT_FOUND	No entry exists
sceNETCNF_CAPACITY_ERROR	Amount remaining is less than 94 Kbytes
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfGetCount

Get number of files

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfGetCount(
 char *fname,                Pathname of configuration management file
 int type);                  File type
                               0: Connection environment configuration file
                               1: Connection configuration file
                               2: Modem configuration file
```

Calling conditions

Can be called from a thread.

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

Description

This function gets the number of files of the type specified by type that appear in the configuration management file specified by *fname*.

If the configuration management file specified by *fname* does not exist, no error occurs and 0 is returned.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF
pfs?:/etc/network/net.db

Return value

0 <=	Number of valid files of specified type
sceNETCNF_INVALID_FNAME	*fname* is invalid
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfGetList

Get file list

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfGetList(
 char *fname,                    Pathname of configuration management file
 int type,                       File type
                                   0: Connection environment configuration file
                                   1: Connection configuration file
                                   2: Modem configuration file
 sceNetCnfList_t *p);            Pointer to beginning of file list
```

Calling conditions

Can be called from a thread.

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

Description

This function gets a list of configuration files of the type specified by *type* that appear in the configuration management file specified by *fname*. The area pointed to by *p* must be allocated in advance by first calling sceNetCnfGetCount() to obtain the number of configuration files, then calling AllocSysMemory() for the required size.

If the configuration management file specified by *fname* does not exist, no error occurs and 0 is returned.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF
pfs?:/etc/network/net.db

Return value

0 <=	Number of valid files of specified type
sceNETCNF_INVALID_FNAME	*fname* is invalid
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfInitIFC

Initialize configuration information for each interface

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	July 2, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfInitIFC(
 sceNetCnfInterface_t *ifc);
```

ifc — Pointer to configuration information for each interface to be initialized

Calling conditions

Can be called from a thread.

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

Description

This function initializes each member of the sceNetCnfInterface_t structure (configuration information for each interface) specified by *ifc* to an "unset" state.

Return value

Always 0.

sceNetCnfLoadConf

Load configuration file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfLoadConf(
 sceNetCnfEnv_t *e);          Load environment
```

Calling conditions

Can be called from a thread.

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

Description

This function loads the configuration file indicated by *e*->arg_fname and saves it in the load environment, *e*.

When *e*->req is sceNetCnf_REQ_NET, the configuration file is loaded as a NET_CNF file, and the data is stored in members below *e*->root. When *e*->req is sceNetCnf_REQ_ATTACH, the configuration file is loaded as an ATTACH_CNF file, and the data is stored in members below *e*->ifc.

Notes

This function is provided for use with a program that delivers the configuration to the network stack.

Return value

If processing terminates normally, zero is returned. If an error occurs, any of the following error codes may be returned.

Table 4-11

Constant	Meaning
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_SYNTAX_ERROR	Syntax error
sceNETCNF_MAGIC_ERROR	Magic missing or incorrect
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfLoadDial

Load dialing definition file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfLoadDial(
 sceNetCnfEnv_t *e,          Load environment
 sceNetCnfPair_t *pair);     interface keyword information
```

Calling conditions

Can be called from a thread.

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

Description

This function loads the dialing definition file that is indicated by *e*->arg_fname and stores it in *pair*->ctl->dial.

Notes

This function is provided for use with a program that delivers the configuration to the network stack.

Return value

If processing terminates normally, zero is returned. If an error occurs, any of the following error codes may be returned.

Table 4-12

Constant	Meaning
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_SYNTAX_ERROR	Syntax error
sceNETCNF_MAGIC_ERROR	Magic missing or incorrect
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfLoadEntry

Load configuration file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfLoadEntry(
 char *fname,                Pathname of configuration management file
 int type,                   File type
                               0: Connection environment configuration file
                               1: Connection configuration file
                               2: Modem configuration file
 char *usr_name,             Configuration name
 sceNetCnfEnv_t *e);         Load environment
```

Calling conditions

Can be called from a thread.

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

Description

This function reads the configuration data of the entry specified by usr_name of the configuration management file, fname, using the load environment, *e*.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

The setting name is unconditionally set as shown below when type == 0.

Combination"index"

The following restrictions are placed on each target device for "index". If an "index" other than those listed below is specified, sceNETCNF_INVALID_USR_NAME will be returned.

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

The following members of the load environment *e* need to be set when the function is called.

mem_ptr	The next address used within the memory area
mem_last	Last byte of the memory region + 1
f_no_check_magic	0 as long as there are no special circumstances during development

f_no_decode	Can be 1 for development, but usually 0 for titles
f_verbose	Can be 1 for development, but usually 0 for titles
file_err	Must be initialized to 0
alloc_err	Must be initialized to 0
syntax_err	Must be initialized to 0

dir_name, arg_fname and req are automatically set during sceNetCnfLoadEntry() processing.

When no add processing is performed for the same load environment, mem_ptr is always set to the starting address of the prepared memory area, and mem_last is always set to the address following the end of the prepared memory area.

When add processing is performed, mem_ptr and mem_last are set only when the configuration data is first read.

Return value

0 <=	Normal termination
sceNETCNF_INVALID_USR_NAME	*usr_name* is invalid
sceNETCNF_INVALID_FNAME	*fname* is invalid
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_ENTRY_NOT_FOUND	Entry specified by *usr_name* could not be found
sceNETCNF_NG	Error occurred during loading
sceNETCNF_SYNTAX_ERROR	Syntax error
sceNETCNF_MAGIC_ERROR	Magic missing or incorrect
sceNETCNF_IO_ERROR	I/O error occurred

sceNetCnfMergeConf

Merge configuration data

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfMergeConf(
 sceNetCnfEnv_t *e);          Load environment
```

Calling conditions

Can be called from a thread.

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

Description

This function merges the ifc and dev data within the lists from *e*->root and *e*->pair_head in priority order, and stores the result as the ctl member within each interface keyword information. It also allocates the dial member area within each interface keyword information.

Notes

This function is provided for use with a program that delivers the configuration to the network stack.

Return value

If processing terminates normally, zero is returned. If an error occurs, the following error code is returned.

Table 4-13

Constant	Meaning
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed

sceNetCnfName2Address

Convert internal-format IP address

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	March 26, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfName2Address(
 sceNetCnfAddress_t *paddr,        Address of structure variable for receiving internal-format IP address
 char *name,);                     Dot-format IP address
```

Calling conditions

Can be called from a thread.

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

Description

This function converts an IP address expressed in dot format to an internal-format IP address and saves it in the area pointed to by *paddr*.

Dot-format IP addresses include any of the following formats.

- num8.num8.num8.num8 (Class C)
- num8.num8.num16 (Class B)
- *num8.num24* (Class A)
- *num32* (direct specification)

num8 Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 8bit

num16 Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 16bit

num24 Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 24bit

num32 Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 32bit

The octal, decimal, or hexadecimal notation rules are the same as those used for the C language.

Return value

If processing terminates normally, 1 is returned. If conversion fails, 0 is returned.

sceNetCnfSetLatestEntry

Change list position in configuration management file

Library	*Introduced*	*Documentation last modified*
netcnf	2.2	October 11, 2001

Syntax

```
#include <netcnf.h>
int sceNetCnfSetLatestEntry(
 char *fname,                Pathname of configuration management file
 int type,                   File type
                               0: Connection environment configuration file
                               1: Connection configuration file
                               2: Modem configuration file
 char *usr_name);            Configuration name
```

Calling conditions

Can be called from a thread.

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

Description

This function moves the *usr_name* entry within the configuration management file specified by *fname* to the beginning of the file. By calling this function each time a device is connected, the entries in the configuration management file will be arranged in the order that the devices were connected.

A title application should perform processing that displays a list of configurations to the user so that the user can select the configuration for which the connection is to be made. At this time, the first entry of the list should be presented as the default.

The pathname of the configuration management file is unconditionally set as shown below when the device is "mc?:" or "pfs?:".

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

The setting name is unconditionally set as shown below when type == 0.

Combination"index"

The following restrictions are placed on each target device for "index". If an "index" other than those listed below is specified, sceNETCNF_INVALID_USR_NAME will be returned.

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

Return value

0 <	Processing was successful
sceNETCNF_INVALID_USR_NAME	*usr_name* is invalid
sceNETCNF_INVALID_FNAME	*fname* is invalid
sceNETCNF_OPEN_ERROR	File cannot be opened
sceNETCNF_SEEK_ERROR	Attempt to get file size failed
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed
sceNETCNF_READ_ERROR	Error occurred when reading file
sceNETCNF_WRITE_ERROR	Error occurred when writing file
sceNETCNF_IO_ERROR	I/O error occurred

Chapter 5: Network Device Library
Table of Contents

Structures

sceInetDevOps_t

Network interface structure

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Structure

```
struct sceInetDevOps {
    struct sceInetDevOps *forw, *back;  /* links for INET layer */
    char interface[8 + 1];  /* interface name */
    char *module_name;  /* module name */
    char *vendor_name;  /* vendor name of device */
    char *device_name;  /* device name of device */
    u_char bus_type;  /* bus type */
    u_char bus_loc[31];  /* bus location */
    u_short prot_ver;  /* protocol version */
    u_short impl_ver;  /* implement version */
    void *priv;  /* private for NETDEV layer */
    int flags;  /* various flags */
    int evfid;  /* event flag ID */
    struct sceInetPktQ rcvq;  /* receive queue */
    struct sceInetPktQ sndq;  /* send queue */
    int (*start)(void *priv, int flags);  /* start function */
    int (*stop)(void *priv, int flags);  /* stop function */
    int (*xmit)(void *priv, int flags);  /* transmit function */
    int (*control)(void *priv, int code, void *ptr, int len);  /* control function */
    u_long ip_addr;  /* IP address */
    u_long ip_mask;  /* IP subnet mask */
    u_long broad_addr;  /* IP broadcast address */
    u_long gw_addr;  /* gateway address */
    u_long ns_addr1;  /* primary DNS address */
    int mtu;  /* MTU */
    u_char hw_addr[16];  /* hardware address */
    u_char dhcp_hostname[256];  /* host name for DHCP */
    int dhcp_hostname_len;  /* length of hostname */
    int dhcp_flags;  /* flags for DHCP */
    void *reserved[4];  /* reserved */
    u_long ns_addr2;  /* secondary DNS address */
    void *pppoe_priv;  /* private for PPPoE */
} sceInetDevOps_t;
```

Description

The interface name (*interface*) is a name that is used for identifying each network interface.

It is set to "eth*X*" for an Ethernet interface or "ppp*X*" for PPP when the interface is registered by the INET layer. X is an identification number that is incremented beginning with 0. In the current implementation, the identification numbers are basically assigned in the order in which the interfaces are registered. The identification number cannot be predicted by the NETDEV layer.

For the module name (*module_name*), specify the NETDEV layer module name. Use the string obtained by removing ".irx" from the execution file name. If that NETDEV layer has an even lower layer, specify a string formed by separating the module name and that lower layer's module name by a comma (,).

The vendor name (*vendor_name*) and device name (*device_name*) are the vendor name and device name of the device when the interface handles physical devices. When the interface handles virtual devices, you should also set some string.

The *module_name*, *vendor_name*, and *device_name* must each satisfy the following conditions.

- They must not be NULL.
- They must be strings that end with NUL ('\0').
- They must not be strings that contain ',' or '='.
- They must not be empty strings ("").

The string lengths are not specifically limited.

For the bus type (*bus_type*), set any of the following values during registration.

Table 5-1

Constant	Value	Meaning
sceInetBus_Unknown	0	Unknown (This setting is not recommended)
sceInetBus_USB	1	USB device
sceInetBus_1394	2	(Reserved)
sceInetBus_PCMCIA	3	(Reserved)
sceInetBus_PSEUDO	4	Pseudo device
sceInetBus_NIC	5	(Reserved)

Information about the location on the bus (*bus_loc*) is defined in the current specifications only when the device is a USB device. During registration, store the 7 bytes that are obtained with the sceUsbdGetDeviceLocation() function starting at the beginning of *bus_loc*.

bus_type and *bus_loc* are used for identification when multiple devices having exactly the same *vendor_name* and *device_name* are connected at the same time. Set these values even when the NETDEV layer does not support multiple devices.

For the protocol version (*prot_ver*), which is a field that was prepared for a future extension of the NETDEV interface specifications, the version of these NETDEV interface specifications that is assumed by the NETDEV layer must be set. With the current specifications, 2 should be set.

For the implementation version (*impl_ver*), set the NETDEV layer implementation version for each protocol version. Setting a sequence number that starts with 0 and is incremented each time the NETDEV layer implementation changes is recommended.

The NETDEV layer pointer (*priv*) is a pointer to the data structure that the NETDEV layer is to use for each device. The value of this field is directly passed for the argument *priv* for each of the *start*, *stop*, and *xmit* functions. However, the INET layer has nothing to do with this value. The kind of value to set for *priv* is freely determined by the NETDEV layer side.

There are two types of flags (*flags*). One type is used by the NETDEV layer to report the attributes of the network interface to the INET layer. The other type is used internally within the INET layer or by the INET Control interface. The flag values are defined as follows.

Table 5-2

Constant	Value	Meaning
sceInetDevF_Up	0x0001	Interface is up
sceInetDevF_Running	0x0002	Interface is available
sceInetDevF_Broadcast	0x0004	Broadcast address has been explicitly set
sceInetDevF_ARP	0x0010	Module is an Ethernet module
sceInetDevF_DHCP	0x0020	DHCP is to be used
sceInetDevF_PPP	0x0040	Module is a PPP module
sceInetDevF_NIC	0x0080	(Reserved)
sceInetDevF_Error	0x0100	Error
sceInetDevF_PPPoE	0x0200	(Reserved)

The only bit among these that the INET layer must set is sceInetDevF_ARP for Ethernet or sceInetDevF_PPP for PPP. All other bits must be off when the interface is registered.

The event flag ID (*evfid*), which is a field set by the INET layer, stores the ID of the event flag that was generated by the INET layer. The NETDEV layer references this field and reports state changes to the INET layer. The following event flag patterns are used when state changes are reported.

Table 5-3

Constant	Value	Meaning
sceInetDevEFP_StartDone	0x00000001	Notification that start processing has completed for the start (start use) function. The INET layer waits for this notification before starting send/receive processing.
sceInetDevEFP_PlugOut	0x00000002	Notification that data cannot be exchanged between the NETDEV layer and the device because the device was unplugged from the connector.
sceInetDevEFP_Recv	0x00000004	Notification that the received packet was added to the receive packet queue.
sceInetDevEFP_Error	0x00000010	Notification that an error (excluding a timeout) occurred for which send/receive could not continue for a reason other than a PlugOut.
sceInetDevEFP_TimeOut	0x00000020	Notification that a timeout error occurred for which send/receive could not continue for a reason other than a PlugOut.
sceInetDevEFP_InetUse	0xffff0000	Bit used by the INET layer. The reference or setting of this bit by the NETDEV layer is prohibited.

The receive packet queue (rcvq) and send packet queue (sndq) are pointers that point to packet queues that the INET layer and NETDEV layer use for exchanging send/receive packets. For information about the packet queue structure, see the descriptions of sceInetPktQ and sceInetPkt.

start, stop, xmit, and control are pointers to the start use function, stop use function, start transmission function, and control function. For details, see "Functions in the NETDEV Layer."

The following fields are mainly used internally by the INET layer.

- IP address (*ip_addr*)
- Subnet mask (*ip_mask*)
- Broadcast address (*broad_addr*)
- Gateway address (*gw_addr*)
- Name server address (*ns_addr1* / *ns_addr2*)
- Maximum number of transfer bytes (*mtu*)
- Hardware address (*hw_addr*)
- DHCP host name and its length (*dhcp_hostname* / *dhcp_hostname_len*)
- DHCP flags (*dhcp_flags*)
- PPPoE area (*pppoe_priv*)

A NETDEV layer module that handles Ethernet should perform the following processing for these fields.

- During registration, set Ethernet Address for *hw_addr*, set 1500 for *mtu*, and set 0 for all other fields.
- After registration, the above fields need not be referenced. Their values must not be changed because they may be used by the INET layer.

A NETDEV layer module that handles PPP should perform the following processing for these fields.

- Set all fields to 0 during registration.
- Before the start completion notification is reported, set the remote side MRU (Maximum-Receive-Unit) value that was defined for PPP negotiation (or the default MRU value) for the *mtu* field and set the following values for the *ip_addr* and *ip_mask* fields according to the connection method.

Table 5-4

PPP Connection	ip_addr	ip_mask
Connected by using numbered	Local IP address	Local netmask
Connected by using un-numbered	0x00000000	0x00000000

There are no fields that must be referenced when the start function is called. All PPP parameters that are required for start processing are set by using the control function (*control*). No fields other than the above fields need be referenced or changed. Change is "prohibited" because they may be used by the INET layer.

The reserved area (*reserved*) is an area that is used by the INET layer. This area need not be initialized, and its contents must not be referenced or changed.

See also

sceInetRegisterNetDevice()

sceInetPkt_t

Packet structure

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

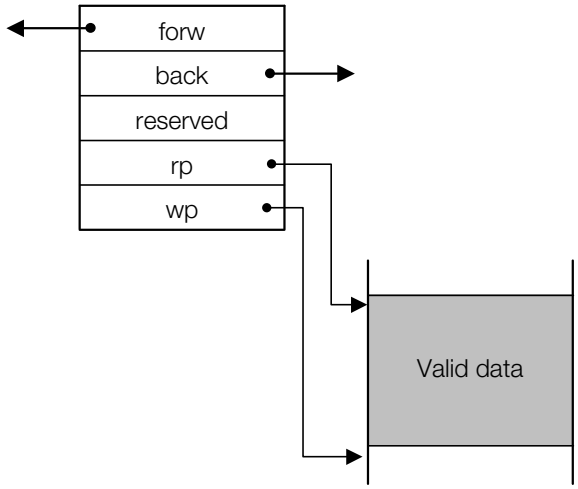
Structure

```
typedef struct sceInetPkt {
    struct sceInetPkt *forw, *back;     Forward and backward links
    int reserved[2];                    Reserved
    u_char *rp;                         Reference pointer (beginning of valid data)
    u_char *wp;                         Setting pointer (immediately after valid data)
} sceInetPkt_t;
```

Description

This is a data structure that corresponds to individual Ethernet packets or IP packets. The actual packets are placed in a separate area and pointed to by pointers.

Figure 5-1



The forward and backward links (*forw*, *back*) are fields that are used for connecting packets as a queue.

The reserved member (*reserved*) is a field that is used internally by each layer. Although this field can be freely used within the NETDEV layer until the packet is passed to the INET layer or modem layer, the value that was set cannot be referenced by another layer. Also, when the packet is moved between layers, the value cannot be expected to be maintained.

The reference pointer (*rp*) points to the beginning of the valid data area, and the setting pointer (*wp*) points to the byte that is immediately after the valid data area.

During transmit, the data from the byte indicated by *rp* to the byte immediately before the byte indicated by *wp* is transmitted. When that transmission data will not be subsequently accessed, the sceInetFreePkt() function should be used to free that packet structure and the related data area.

During receive, if the sceInetAllocPkt() function is used to allocate the packet area, both the reference pointer (*rp*) and setting pointer (*wp*) are initialized so that they point to the beginning of the data area. After the received data has been written starting from the setting pointer (*wp*), the setting pointer should be updated. The following example shows receive processing.

```
// Example in which packets are created from receive data having
// ptr as the starting address and len as the number of data bytes

struct sceInetPkt *p;

if(NULL == (p = sceInetAllocPkt(ops, len)))
            return(-1);          // Error due to insufficient memory
bcopy(ptr, p->wp, len);     // Copy data contents
p->wp += len;               // Update setting pointer
sceInetPktEnQ(&ops->rcvq, p);                 // Add to queue
SetEventFlag(ops->evfid, sceInetDevEFP_Recv); // Event notification
```

In this example, the allocated data size and valid data size are the same. However, since the INET layer determines the valid data area only according to *rp* and *wp*, the number of valid data bytes may also be smaller than the allocated area.

See also

sceInetAllocPkt(), sceInetFreePkt(), sceInetPktEnq(), sceInetPktDeQ()

sceInetPktQ_t

Packet queue structure

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Structure

```
typedef struct sceInetPktQ {
    struct sceInetPkt *head;        Pointer to starting packet
    struct sceInetPkt *tail;        Pointer to final packet
} sceInetPktQ_t;
```

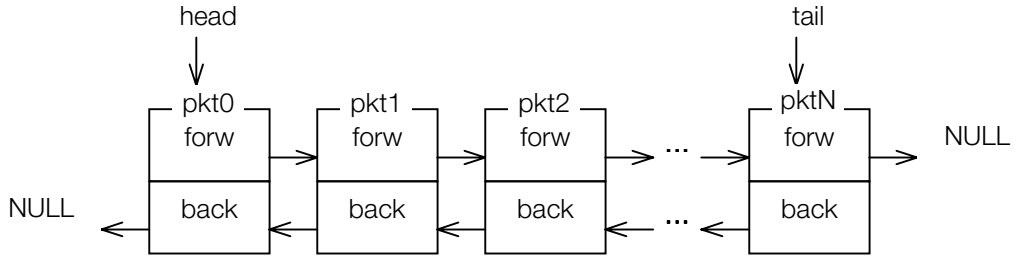
Description

Multiple packets are queued within the NETDEV layer with a packet queue having forward and backward links. sceInetPktQ is a structure that points to that packet queue.

If there are no packets in the packet queue, both *head* and *tail* are NULL. If there is only one packet in the packet queue, *head* and *tail* both point to that packet, and *forw* and *back* within that packet are both NULL.

If there are multiple packets in the packet queue, they are set or referenced as shown in the following figure.

Figure 5-2



Library Functions

sceInetAllocMem

Allocate memory area

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
void *sceInetAllocMem(
 struct sceInetDevOps *ops,          Pointer to network interface structure
 int siz);                           Memory size to be allocated (bytes)
```

Calling conditions

Can be called from a thread.

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

Description

This function allocates a memory area from the memory pool that is managed by the INET layer. The allocated area is to be used by the network interface for purposes other than packets. It differs from AllocSysMemory() in that 4-byte alignment is guaranteed for the starting address of the memory area that is allocated.

sceInetAllocPkt() and sceInetFreePkt() should be used to allocate and free packet areas.

Besides this sceInetAllocMem() function, the AllocSysMemory() function may be used to allocate a memory area that is to be used for purposes other than packets, However, when these areas are to be freed, the correct free functions that correspond to these allocate functions should be used to free these areas.

To allocate an area for a network interface structure, NULL should be specified for *ops*.

Return value

Starting address of allocated area

sceInetAllocPkt

Allocate packet memory area

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
struct sceInetPkt *sceInetAllocPkt(
 struct sceInetDevOps *ops,          Pointer to network interface structure
 int siz);                           Memory size to be allocated (bytes)
```

Calling conditions

Can be called from a thread.

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

Description

This function allocates a receive packet structure and data area and returns a pointer to the packet structure.

The number of bytes in the data area is specified with *siz*.

The starting address of the allocated data area is always 4-byte aligned, and the reference pointer (*rp*) and setting pointer (*wp*) are both initialized to the beginning of the allocated data area.

Return value

Pointer to allocated packet structure.

sceInetFreeMem

Free memory area

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
void sceInetFreeMem(
 struct sceInetDevOps *ops,          Pointer to network interface structure
 void *pkt);                         Starting address of memory area to be freed
```

Calling conditions

Can be called from a thread.

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

Description

This function frees a memory area that was allocated by sceInetAllocMem().

Return value

None.

sceInetFreePkt

Free packet memory area

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
void sceInetFreePkt(
 struct sceInetDevOps *ops,          Pointer to network interface structure
 struct sceInetPkt *pkt);            starting address of memory area to be freed
```

Calling conditions

Can be called from a thread.

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

Description

This function frees the specified packet structure and the related data area.

This function is only called by the NETDEV layer send complete process. It is not called from anywhere else. Send complete is used here to mean simply that "there will be no subsequent access to that packet structure and its related data area." It has nothing to do with whether or not that packet was sent to the network.

The memory area that is passed as the send packet must be freed with the sceInetFreePkt() function, not the FreeSysMemory() or sceInetAllocMem() function.

Return value

None.

sceInetPktDeQ

Extract packet from packet queue

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
struct sceInetPkt *sceInetPktDeQ(
 struct sceInetPktQ *que);
```

que Packet queue

Calling conditions

Can be called from a thread.

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

Description

This function extracts a packet from the beginning of the packet queue, *que*.

Notes

A packet can also be extracted by directly manipulating the data structure without using this function. However, in this case, exclusive control processing between threads must not be forgotten.

Return value

Pointer to extracted packet structure.

sceInetPktEnQ

Add packet to packet queue

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
void sceInetPktEnQ(
 struct sceInetPktQ *que,          Pointer to packet queue
 struct sceInetPkt *pkt);          Pointer to packet structure
```

Calling conditions

Can be called from a thread.

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

Description

This function adds the packet, *pkt*, to the end of the packet queue, *que*.

Notes

A packet can also be added to the queue by directly manipulating the data structure without using this function. However, in this case, exclusive control processing between threads must not be forgotten.

Return value

None.

sceInetPrintf

Record log

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
int sceInetPrintf(
 const char *fmt,...);                    Same specification as the format string of printf
```

Calling conditions

Can be called from a thread.

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

Description

This function writes arbitrary information in the form of a log, to the logging area that is managed by the INET layer. The format and output data specifications are the same as those for the printf() function of the standard C library.

The output results can be read with sceInetGetLog().

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetRand

Generate random number

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
u_int sceInetRand(
void);
```

Calling conditions

Can be called from a thread.

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

Description

This function returns a random number that is generated by the INET layer.

Values that are returned by this function are all in the u_int range from 0 to 0xffffffff.

Since this function can be called from multiple threads, repeatability is not guaranteed at all.

Return value

Random number generated by the INET layer.

sceInetRegisterNetDevice

Register network interface

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
int sceInetRegisterNetDevice(
 struct sceInetDevOps *ops);
```

ops — Pointer to network interface structure to be registered

Calling conditions

Can be called from a thread.

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

Description

This function registers a network interface.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetUnregisterNetDevice

Delete network interface registration

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
int sceInetUnregisterNetDevice(
 struct sceInetDevOps *ops);
```

ops — Pointer to network interface structure for which registration is to be deleted

Calling conditions

Can be called from a thread.

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

Description

This function deletes a network interface registration.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

Functions Implemented in the NETDEV Layer

control

Control function

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

int (*control)(

void **priv,*	Pointer to private data
int *code,*	Control code
void **ptr,*	Data buffer address
int *len***)**	Data buffer size (bytes)

Calling conditions

Can be called from a thread.

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

Description

The control function (control) is called via the INET layer when an application calls the sceInetInterfaceControl() function and specifies a NETDEV layer or MODEM layer control code. Appropriate processing should be performed and an appropriate value should be returned according to the value of *code*.

Control codes for requesting Ethernet statistical information or for configuring Ethernet interface-dependent settings may be passed to a NETDEV layer module that handles the Ethernet interface.

Also, MODEM layer control codes may be passed to a NETDEV layer that handles a modem. Since a MODEM layer control code is determined by whether bit 30 is 1, *code*, *ptr*, and *len* should be relayed directly to that MODEM layer.

The NETDEV layer common control codes are each explained in detail later.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

start

Start use function

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

int (*start)(

void **priv,*	Pointer to private data
int *flags***)**	Flags (for extension; currently, always 0)

Calling conditions

Can be called from a thread.

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

Description

The start use function (start) is called from a higher layer when the network interface is brought up.

The start use function should perform processing such as initializing the device as necessary. However, initialization processing need not be completed when control returns from the start use function. The completion of processing is reported to the INET layer by setting sceInetDevEFP_StartDone in the evfid field of the sceInetDevOps structure. If the start use function returns 0, the INET layer considers that interface to be in use.

The *flags* argument is reserved for a future extension. In the current specifications, 0 is passed in this argument.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

stop

Stop use function

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

int (*stop)(

void **priv,*	Pointer to private data
int *flags***)**	Flags (for extension; currently, always 0)

Calling conditions

Can be called from a thread.

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

Description

The stop use function (stop) is called from a higher layer when the network interface is taken down or when sceInetDevEFP_PlugOut is reported from the NETDEV layer to the INET layer.

In the latter case, sceInetUnregisterNetDevice() should be used to delete the registration.

The *flags* argument is reserved for a future extension. In the current specifications, 0 is passed in this argument.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

xmit

Start transmission function

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

int (*xmit)(

void **priv,*	Pointer to private data
int *flags***)**	Flags (for extension; currently, always 0)

Calling conditions

Can be called from a thread.

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

Description

The start transmission function (xmit) is called when a packet is added to the transmission packet queue within the INET layer.

When the NETDEV layer status is sendable status, this transmission start function is used to perform transmission processing. Otherwise, transmission processing is performed when the NETDEV layer state becomes sendable.

The *flags* argument is reserved for a future extension. In the current specifications, 0 is passed in this argument.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

NETDEV Layer Common Control Codes

sceInetNDCC_GET_IF_TYPE

Get network interface type

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_IF_TYPE,  // 0x80000100
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code returns the type of the NETDEV layer module as a network interface.

sceInetNDIFT_ETHERNET indicates that the NETDEV layer module is an Ethernet interface and, therefore, implicitly has the capability of returning Ethernet interface-dependent statistical information.

sceInetNDIFT_PPP indicates that the NETDEV layer module is a PPP interface. In the current specifications, no implicitly assumed capabilities are defined.

sceInetNDIFT_GENERIC indicates that the NETDEV layer module is a network interface that has only general NETDEV layer functions other than those described above.

Return value

Any of the following values representing network interface types may be returned.

Table 5-5

Constant	Value	Meaning
sceInetNDIFT_GENERIC	0x00000000	General
sceInetNDIFT_ETHERNET	0x00000001	Ethernet interface
sceInetNDIFT_PPP	0x00000002	PPP

sceInetNDCC_GET_RX_BYTES

Get number of receive bytes

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_RX_BYTES,  // 0x80010002
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of bytes of data that the NETDEV layer received from a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_RX_DROPPED

Get number of dropped receive packets

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_RX_DROPPED,  // 0x80010006
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of packets that were lost due to insufficient buffer area when the NETDEV layer receives data from a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_RX_ERRORS

Get number of receive errors

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_RX_ERRORS,  // 0x80010004
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of errors that occurred when the NETDEV layer received data from a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_RX_PACKETS

Get number of receive packets

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_RX_PACKETS,  // 0x80010000
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of packets that the NETDEV layer received from a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_THPRI

Get thread priority

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_THPRI,  // 0x80000000
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code returns the priority of a thread that will be or was created by the NETDEV layer module.

If the NETDEV layer module creates multiple threads and different priorities must be assigned to them, sceInetNDCC_GET_THPRI is used for dealing with the priority on which those priorities are to be based. The method by which each priority is determined from the base depends on the NETDEV layer module.

Return value

NETDEV layer thread priority.

sceInetNDCC_GET_TX_BYTES

Get number of transmit bytes

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_TX_BYTES,  // 0x80010003
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of bytes of data that the NETDEV layer sent to a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_TX_DROPPED

Get number of dropped transmit packets

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_TX_DROPPED,  // 0x80010007
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of packets that were lost due to insufficient buffer area when the NETDEV layer sends data to a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_TX_ERRORS

Get number of transmit errors

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_TX_ERRORS,  // 0x80010005
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of errors that occurred when the NETDEV layer sent data to a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_GET_TX_PACKETS

Get number of transmit packets

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_GET_TX_PACKETS,  // 0x80010001
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code gets the total number of packets that the NETDEV layer sent to a lower level layer and stores this in the area specified by (*ptr*, *len*).

If *len* is not sizeof(int) (=4), an error occurs.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

sceInetNDCC_SET_THPRI

Set thread priority

Library	*Introduced*	*Documentation last modified*
netdev	2.2	March 26, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID to be manipulated
 sceInetNDCC_SET_THPRI,  // 0x81000000
 void *ptr,                               Starting address of data area
 int len);                                Size of data area (sizeof(int))
```

Description

This control code treats the value that is stored in the area indicated by (*ptr*, *len*=4) as an integer priority and changes the priority of a thread that will be or was created by the NETDEV layer module to that value. If *len* is not sizeof(int) (=4), an error occurs.

If the NETDEV layer module has not yet created a thread, the initial priority when the thread is created is changed to this value. If the thread was created but has not yet been activated (it is in DORMANT state), the priority is changed to this value after the thread is activated. If the thread has already been activated, ChangeThreadPriority() is used to immediately change the priority.

Notes

The initial value of the thread priority before the priority has been set according to sceInetNDCC_SET_THPRI depends on the NETDEV layer module.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

PPP Layer Control Codes

scePPPCC_GetAllowAccmNego

Get allow ACCM negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowAccmNego, // 0x90008018
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing ACCM negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowAccmValue

Get allow ACCM value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowAccmValue,    // 0x9000801f
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code gets the ACCM value that is used as the allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowAccNego

Get allow ACC negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowAccNego,     // 0x9000801b
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing ACC negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowAddressNego

Get allow IP address negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowAddressNego,   // 0x9000801c
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing IP address negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowAuth

Get allow authentication method

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetAllowAuth,     // 0x90008020
 void *ptr,                             Starting address of data area (not checked)
 int len);                              Size of data area (not checked)
```

Description

This control code gets the value that represents the authentication method that is used as an allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Notes

The correspondence between values representing authentication methods and keywords in the "NET Configuration File Specification" is as follows.

0	any	(no authentication)
1	pap	(PAP authentication only)
2	chap	(CHAP authentication only)
3	pap/chap	(PAP authentication followed by CHAP authentication)
4	chap/pap	(CHAP authentication followed by PAP authentication)

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowDNS1

Get allow first DNS address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                          Network interface ID of target
 scePPPCC_GetAllowDNS1,   // 0x90008029
 void *ptr,                       Starting address of data area (not checked)
 int len);                        Size of data area (not checked)
```

Description

This control code gets the first DNS address that is used as an allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowDNS1Nego

Get allow first DNS address negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowDNS1Nego,     // 0x90008027
 void *ptr,                               Starting address of data area (not checked)
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing first DNS address negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowDNS2

Get allow second DNS address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                          Network interface ID of target
 scePPPCC_GetAllowDNS2,   // 0x9000802a
 void *ptr,                       Starting address of data area (not checked)
 int len);                        Size of data area (not checked)
```

Description

This control code gets the second DNS address that is used as an allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowDNS2Nego

Get allow second DNS address negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetAllowDNS2Nego,     // 0x90008028
 void *ptr,                             Starting address of data area (not checked)
 int len);                              Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing second DNS address negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowIpAddress

Get allow IP address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowIpAddress,   // 0x90008021   Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the IP address that is used as an allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowIpMask

Get allow subnet mask

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetAllowIpMask,    // 0x90008022   Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code gets the subnet mask that is used as an allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowMagicNego

Get allow MAGIC negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowMagicNego,   // 0x90008019   Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing MAGIC negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowMruNego

Get allow MRU negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowMruNego,    // 0x90008017   Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing MRU negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowMruValue

Get allow MRU value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowMruValue,    // 0x9000801e  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the MRU value that is used as an allowance value and stores it in the area specified by ptr and len.

If len is not sizeof(short) (=2), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowPrcNego

Get allow PRC negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowPrcNego,    // 0x9000801a  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing PRC negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowVjcompNego

Get allow VJCOMP negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAllowVjcompNego,  // 0x9000801d  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for allowing VJCOMP negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAnyDial

Get dialing string for special line

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetAnyDial,    // 0x90008030  Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code transfers the string set for the script used when dialing with a special line, such as a digital line, to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAuthKey

Get auth_key string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetAuthKey,     // 0x90008005   Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code transfers the string used as the local key during authentication to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAuthName

Get auth_name string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetAuthName,    // 0x90008004 Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code transfers the string used as the local name during authentication to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetChapType

Get CHAP authentication algorithm number

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_ChapType,      // 0x90008037    Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the number representing the algorithm that is forcibly used during CHAP authentication and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Notes

The correspondence between algorithm number and keywords in the "NET Configuration File Specification" is as follows.

0x00	no	(no algorithm forcibly used)
0x05	md5	(MD5)
0x80	ms or ms-v1	(MS CHAP version 1)
0x81	ms-v2	(MS CHAP version 2)

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetChatDial

Get char_dial string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetChatDial,     // 0x90008001  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code transfers the string that was set for the script that is used for modem initialization to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Notes

The relationship between this string and keywords in the "NET Configuration File Specification" is as follows.

When answer mode is false:

> String formed by concatenating chat_init, chat_additional, and chat_dial

When answer_mode is true:

> String formed by concatenating chat_init, chat_additional, "TIMEOUT %d", and chat_answer ("TIMEOUT %d" is included only when answer_timeout is specified)

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetChatLogin

Get chat_login string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetChatLogin,    // 0x90008002 Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code transfers the string set for authenticating a PPP server that performs login-type authentication to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetConnectTimeout

Get connection timeout value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetConnectTimeout,  // 0x9000802b  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the timeout value (in seconds) used during line connection and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetCurrentStatus

Get status string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_CurrentStatus,    // 0x90008038  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code transfers the string indicating the current state kept by the PPP layer to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Strings set by PPP will not exceed 79 characters.

Notes

Since the string indicating the current state kept by the PPP layer is modem-dependent, this string should only be referenced, and should not be used to affect the behavior of the application.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetDialingType

Get dialing type

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetDialingType,    // 0x9000802c  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the value representing the dialing type and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Notes

The correspondence between the values of dialing type for the line and keywords in the "NET Configuration File Specification" is as follows.

0	tone	(Tone line)
1	pulse	(Pulse line)
2	any	(Other type)

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetHisAddress

Get remote IP address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetHisAddress,  // 0x90008034 Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code gets the remote IP address that was obtained as the result of negotiation with IPCP and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetHisMask

Get remote subnet mask

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetHisMask,    // 0x90008035    Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the remote subnet mask that was obtained as the result of negotiation with IPCP and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetIdleTimeout

Get Idle timeout value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

int sceInetInterfaceControl(

int *id***,**		Network interface ID of target
scePPPCC_GetIpcpTimeout,	**// 0x9000800a**	Starting address of data area (not checked)
void ****ptr*,**		
int *len***);**		Size of data area (not checked)

Description

This control code gets the timeout value (in seconds) for the PPP no-communication state and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetIpcpTimeout

Get IPCP timeout value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetIpcpTimeout,    // 0x90008009  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the timeout value (in seconds) for the IPCP phase of PPP and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetLcpEchoTimeout

Get LCP Keep-Alive timeout value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetLcpEchoTimeout,  // 0x90008039  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the timeout value (in seconds) for the Keep-Alive function in the PPP LCP and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetLcpTimeout

Get LCP timeout value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetLcpTimeout,    // 0x90008008  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the timeout value (in seconds) for the LCP phase of PPP and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetLogFlags

Get log display contents specification

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetLogFlags,     // 0x90008033  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the set of bits specifying the PPP log display contents and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Notes

The correspondence between bit definitions and keywords in the "NET Configuration File Specification" is as follows.

0x00000001	phase	Display PPP state
0x00000002	cp	Display LCP, IPCP state
0x00000004	auth	Display PAP, CHAP state
0x00000008	chat	Display chat processing and replies
0x00000010	private	Also display private information
0x00000020	dll	Display exchange of data in DLL layer
0x00000040	dump	Perform packet dump when DLL data is displayed
0x00010000	timer	Display timer-related information
0x00020000	event	Display events generated for PPP

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetOmitEmptyFrame

Get delete empty frame flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetOmitEmptyFrame,  // 0x90008036  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for determining whether or not to delete empty PPP frames and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetPeerKey

Get peer_key string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetPeerKey,    // 0x90008007  Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code transfers the string used as the remote key during authentication to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetPeerName

Get peer_name string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                Network interface ID of target
 scePPPCC_GetPeerName,    // 0x90008006  Starting address of data area (not checked)
 void *ptr,
 int len);                              Size of data area (not checked)
```

Description

This control code transfers the string used as the remote name during authentication to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetPhoneNumber

Get phone_number string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetPhoneNumber,    // 0x90008003  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code transfers the string representing the dialing number to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetPulseDial

Get pulse dialing string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetPulseDial,    // 0x9000802f  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code transfers the string set for the script used when dialing with a pulse line to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetToneDial

Get tone dialing string

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetToneDial,     // 0x9000802e  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code transfers the string set for the script used when dialing with a tone line to the specified data area.

An error will occur if the length of the string plus 1 is larger than the size of the data area specified by the arguments.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAccmNego

Get request ACCM negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                    Network interface ID of target
 scePPPCC_GetWantAccmNego,    // 0x9000800c  Starting address of data area (not checked)
 void *ptr,
 int len);                                  Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting ACCM value negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAccmValue

Get request ACCM value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                    Network interface ID of target
 scePPPCC_GetWantAccmValue,    // 0x90008013  Starting address of data area (not checked)
 void *ptr,
 int len);                                  Size of data area (not checked)
```

Description

This control code gets the ACCM value that is used as the request value and stores it in the area specified by ptr and len.

If len is not sizeof(int) (=4), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAccNego

Get request ACC negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantAccNego,    // 0x9000800f  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting ACC negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAddressNego

Get request IP address negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                          Network interface ID of target
 scePPPCC_GetWantAddressNego,    // 0x90008010    Starting address of data area (not checked)
 void *ptr,
 int len);                                        Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting IP address negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAuth

Get request authentication method

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantAuth,     // 0x90008014  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the value that represents the authentication method that is used as a request value and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Notes

The correspondence between authentication methods values and keywords in the "NET Configuration File Specification" is as follows.

0	any	(no authentication)
1	pap	(PAP authentication only)
2	chap	(CHAP authentication only)
3	pap/chap	(PAP authentication followed by CHAP authentication)
4	chap/pap	(CHAP authentication followed by PAP authentication)

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantDNS1

Get request first DNS address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantDNS1,     // 0x90008025  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the first DNS address that is used as a request value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantDNS1Nego

Get request first DNS address negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                    Network interface ID of target
 scePPPCC_GetWantDNS1Nego,   // 0x90008023  Starting address of data area (not checked)
 void *ptr,
 int len);                                  Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting first DNS address negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantDNS2

Get request second DNS address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantDNS2,     // 0x90008026  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the second DNS address that is used as a request value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantDNS2Nego

Get request second DNS address negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantDNS2Nego,    // 0x90008024  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting second DNS address negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantIpAddress

Get request IP address

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantIpAddress,    // 0x90008015  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the IP address that is used as a request value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantIpMask

Get request subnet mask

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantIpMask,    // 0x90008016  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the subnet mask that is used as a request value and stores it in the area specified by ptr and len.

If len is not sizeof(struct sceInetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantMagicNego

Get request MAGIC negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                    Network interface ID of target
 scePPPCC_GetWantMagicNego,   // 0x9000800d  Starting address of data area (not checked)
 void *ptr,
 int len);                                  Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting MAGIC negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantMruNego

Get request MRU negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantMruNego,    // 0x9000800b Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting MRU value negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantMruValue

Get request MRU value

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantMruValue,    // 0x90008012  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the MRU value that is used as a request value and stores it in the area specified by ptr and len.

If len is not sizeof(short) (=2), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantPrcNego

Get request PRC negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                  Network interface ID of target
 scePPPCC_GetWantPrcNego,    // 0x9000800e  Starting address of data area (not checked)
 void *ptr,
 int len);                                Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting PRC negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantVjcompNego

Get request VJCOMP negotiation flag

Library	*Introduced*	*Documentation last modified*
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
 int id,                                          Network interface ID of target
 scePPPCC_GetWantVjcompNego,  // 0x90008011       Starting address of data area (not checked)
 void *ptr,
 int len);                                        Size of data area (not checked)
```

Description

This control code gets the flag (Boolean value) for requesting VJCOMP negotiation and stores it in the area specified by ptr and len.

If len is not sizeof(char) (=1), an error will occur.

Return value

When processing terminates normally, sceINETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.