

PlayStation®2 IOP Library Reference

Release 2.4

Inet Libraries

© 2001 Sony Computer Entertainment Inc.

Publication date: October 2001

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

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

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


The *PlayStation®2 IOP Library Reference - Inet 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 - Inet 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 - Inet 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	v
Changes Since Last Release	v
Related Documentation	v
Typographic Conventions	v
Developer Support	vi
Chapter 1: Network (INET) Configuration Library	1-1
Structures	1-3
Control Functions	1-4
Chapter 2: Network Library	2-1
Structures	2-3
Functions	2-10
Control Codes for Connection Control	2-33
INET Layer Control Codes	2-36
Chapter 3: Modem Development Library	3-1
Modem Information Structure	3-3
Modem Control Functions	3-5
Modem Driver Layer Functions	3-7
Modem Driver Layer Common Control Codes	3-13
Modem Driver Layer Common Control Codes (for serial devices)	3-21
Chapter 4: Network Device Library	4-1
Structures	4-4
Library Functions	4-11
Functions Implemented in the NETDEV Layer	4-21
NETDEV Layer Common Control Codes	4-25
PPP Layer Control Codes	4-36

About This Manual

This is the Runtime Library Release 2.4 version of the *PlayStation®2 IOP Library Reference - Inet Libraries* manual.

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

Changes Since Last Release

Chapter 2: Network Library

- In the "Description" section of `sceInetGetLog()`, the description of timeout has been changed.
- A description of `sceInetAbortLog()` has been added.

Chapter 3: Modem Development Library

- A description of the modem information structure `sceModemOps` has been added.
- Descriptions of the following functions have been added:
 - `sceModemRegisterDevice()`
 - `sceModemUnregisterDevice()`

Chapter 4: Network Device Library

- Description of the PPP layer control codes have 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
<code>courier</code>	Indicates literal program code.
<i>italic</i>	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
<i>In North America:</i>	<i>In North America:</i>
Attn: Developer Tools Coordinator	E-mail: PS2_Support@playstation.sony.com
Sony Computer Entertainment America	Web: http://www.devnet.scea.com/
919 East Hillsdale Blvd.	Developer Support Hotline: (650) 655-5566
Foster City, CA 94404, U.S.A.	(Call Monday through Friday,
Tel: (650) 655-8000	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
<i>In Europe:</i>	<i>In Europe:</i>
Attn: Production Coordinator	E-mail: ps2_support@scee.net
Sony Computer Entertainment Europe	Web: https://www.ps2-pro.com/
30 Golden Square	Developer Support Hotline:
London W1F 9LD, U.K.	+44 (0) 20 7859-5777
Tel: +44 (0) 20 7859-5000	(Call Monday through Friday,
	9 a.m. to 6 p.m., GMT)

Chapter 1: Network (INET) Configuration Library

Table of Contents

Structures	1-3
scelnetCtlEventHandlers	1-3
Control Functions	1-4
scelnetCtlDownInterface	1-4
scelnetCtlRegisterEventHandler	1-5
scelnetCtlSetAutoMode	1-6
scelnetCtlSetConfiguration	1-7
scelnetCtlUnregisterEventHandler	1-8
scelnetCtlUpInterface	1-9

Structures

scelnetCtlEventHandlers

Event handler registration structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	March 26, 2001

Structure

```
typedef struct scelnetCtlEventHandlers {
struct scelnetCtlEventHandlers *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
} scelnetCtlEventHandlers_t;
```

Description

This is a calling data structure that becomes an argument when `scelnetCtlRegisterEventHandler()` 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 `scelnetCtlUnregisterEventHandler()`.

See also

`scelnetCtlRegisterEventHandler()`, `scelnetCtlUnregisterEventHandler()`

Control Functions

scelnetCtlDownInterface

Take down interface

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	March 26, 2001

Syntax

```
#include <inet/inetctl.h>
int scelnetCtlDownInterface(
    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

scelNetCtlRegisterEventHandler

Register event handler

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	March 26, 2001

Syntax

```
#include <inet/inetctl.h>
```

```
int scelNetCtlRegisterEventHandler(
    scelNetCtlEventHandlers_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
scelINETCTL_IEV_Attach	Interface was attached
scelINETCTL_IEV_Detach	Interface was detached
scelINETCTL_IEV_Start	Interface was brought up and is available
scelINETCTL_IEV_Stop	Interface was brought down and is unavailable
scelINETCTL_IEV_Error	Interface reported an error (Error=1)
scelINETCTL_IEV_Conf	Configuration that matches device exists
scelINETCTL_IEV_NoConf	No configuration that matches device exists

Either an scelINETCTL_IEV_Conf or scelINETCTL_IEV_NoConf event is reported directly before scelINETCTL_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

scelnetCtlSetAutoMode

Set/cancel auto mode

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	July 2, 2001

Syntax**#include** <inet/inetctl.h>**int** scelnetCtlSetAutoMode(

int <i>f_auto</i>);	== 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 scelnetCnfUpInterface() is called.

Return value

0

scelnetCtlSetConfiguration

Write configuration in INETCTL

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	March 26, 2001

Syntax

```
#include <netcnf.h>
#include <inet/inetctl.h>
int scelnetCtlSetConfiguration(
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 scelnetCtlSetAutoMode().

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 scelnetCtlSetConfiguration() 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 scelnetCtlSetConfiguration(). From then on, devices will no longer be able to be attached according to the previous configuration data.

Return value

sceNETCNF_NG Configuration failed

scelnetCtlUnregisterEventHandler

Delete event handler registration

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	March 26, 2001

Syntax

```
#include <inet/inetctl.h>
```

```
int scelnetCtlUnregisterEventHandler(
```

```
    scelnetCtlEventHandlers_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 scelnetCtlRegisterEventHandler().

Return value

0

scelnetCtlUpInterface

Bring up interface

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inetctl	2.2	March 26, 2001

Syntax

```
#include <inet/inetctl.h>
```

```
int scelnetCtlUpInterface(  
    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	2-3
scelnetAddress_t	2-3
scelnetInfo_t	2-4
scelnetParam_t	2-6
scelnetRoutingEntry_t	2-8
Error Code	2-9
Functions	2-10
scelnetAbort	2-10
scelnetAbortLog	2-11
scelnetAddress2Name	2-12
scelnetAddress2String	2-13
scelnetChangeThreadPriority	2-14
scelnetClose	2-15
scelnetControl	2-16
scelnetCreate	2-17
scelnetGetInterfaceList	2-18
scelnetGetLog	2-19
scelnetGetNameServers	2-20
scelnetGetRoutingTable	2-21
scelnetInterfaceControl	2-22
scelnetName2Address	2-24
scelnetOpen	2-26
scelnetRecv	2-27
scelnetRecvFrom	2-29
scelnetSend	2-30
scelnetSendTo	2-32
Control Codes for Connection Control	2-33
scelnetC_CODE_GET_FLAGS	2-33
scelnetC_CODE_GET_INFO	2-34
scelnetC_CODE_SET_FLAGS	2-35
INET Layer Control Codes	2-36
scelnetCC_GetAddress	2-36
scelnetCC_GetBroadcast	2-37
scelnetCC_GetBusLoc	2-38
scelnetCC_GetBusType	2-39
scelnetCC_GetDeviceName	2-40
scelnetCC_GetDHCPFlags	2-41
scelnetCC_GetDHCPHostName	2-42
scelnetCC_GetFlags	2-43
scelnetCC_GetHWaddr	2-44
scelnetCC_GetImpl	2-45
scelnetCC_GetInterfaceName	2-46
scelnetCC_GetModuleName	2-47
scelnetCC_GetMTU	2-48
scelnetCC_GetNetmask	2-49
scelnetCC_GetProt	2-50

scelnetCC_GetVendorName

2-51

Structures

scelnetAddress_t

Internal-format IP address structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Structure

```
typedef struct scelnetAddress {  
    int reserved;           Reserved area (always 0)  
    char data[12];         IP address (4 bytes) + Reserved (8 bytes)  
} scelnetAddress_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. scelnetName2Address() and scelnetAddress2String() should be used to obtain the IP address.

scelnetInfo_t

Connection information delivery structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	July 2, 2001

Structure**typedef struct scelnetInfo {**

int <i>cid</i> ;	Connection ID
int <i>proto</i> ;	Protocol
int <i>recv_queue_length</i> ;	Number of data bytes in receive buffer
int <i>send_queue_length</i> ;	Number of data bytes in send buffer
struct scelnetAddress <i>local_adr</i> ;	Local address
int <i>local_port</i> ;	Local port number
struct scelnetAddress <i>remote_adr</i> ;	Remote address
int <i>remote_port</i> ;	Remote port number
int <i>state</i> ;	Connection state
int <i>reserved</i> [4];	Reserved area (0)

} scelnetInfo_t;**Description**

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

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

Table 2-1

Constant	Value	Meaning (Corresponding Connection Type)
<code>scelNETI_PROTO_TCP</code>	1	TCP (<code>scelNETT_CONNECT</code> or <code>scelNETT_LISTEN</code>)
<code>scelNETI_PROTO_UDP</code>	2	UDP (<code>scelNETT_DGRAM</code>)
<code>scelNETI_PROTO_IP</code>	3	Raw IP (<code>scelNETT_RAW</code>)

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

Table 2-2

Constant	Value	Meaning (Corresponding Connection Type)
<code>scelNETI_STATE_UNKNOWN</code>	0	State unknown (TCP, UDP, Raw IP)
<code>scelNETI_STATE_CLOSED</code>	1	Closed (TCP, UDP, Raw IP)
<code>scelNETI_STATE_CREATED</code>	2	Created (UDP)
<code>scelNETI_STATE_OPENED</code>	3	Opened (UDP, Raw IP)
<code>scelNETI_STATE_LISTEN</code>	4	TCP internal state
<code>scelNETI_STATE_SYN_SENT</code>	5	TCP internal state
<code>scelNETI_STATE_SYN_RECEIVED</code>	6	TCP internal state
<code>scelNETI_STATE_ESTABLISHED</code>	7	TCP internal state
<code>scelNETI_STATE_FIN_WAIT_1</code>	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

scelnetParam_t

Connection creation parameter structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Structure

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

Description

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

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

Table 2-3

Constant	Value	Meaning
scelNETT_DGRAM	0x0	UDP
scelNETT_CONNECT	0x1	TCP Connect operation (Active-Open)
scelNETT_LISTEN	0x2	TCP Listen operation (Passive-Open)
scelNETT_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
scelNETP_AUTO	-1	Automatic assignment
scelNETP_ANY	0	Any port

If `scelNETP_AUTO` is specified for local port when the Connection type is `scelNETT_CONNECT` or for `remote_port` when the Connection type is `scelNETT_LISTEN`, a free port among the user ports (5000 to 65535) is automatically assigned when `scelnetCreate()` is called.

If `scelNETP_ANY` is specified for `remote_port` when the Connection type is `scelNETT_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 `scelnetName2Address`) 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

scelnetCreate()

scelnetRoutingEntry_t

Routing control table entry

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Structure

```
typedef struct scelnetRoutingEntry {
    struct scelnetAddress dstaddr;      Destination address
    struct scelnetAddress gateway;      Next POP router address
    struct scelnetAddress 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
} scelnetRoutingEntry_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
scelnetRoutingF_Up	0x01	Route is valid
scelnetRoutingF_Host	0x02	Direct delivery (not via a router)
scelnetRoutingF_Gateway	0x04	Indirect delivery (via a router)
scelnetRoutingF_Dynamic	0x08	Dynamically set
scelnetRoutingF_Modified	0x10	Same entry with modification

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

See also

scelnetAddress_t, scelnetGetRoutingTable()

Error Code

INET function return value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

scelnetAbort

Abort processing

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetAbort(
```

<code>int cid,</code>	Connection ID
<code>int flags);</code>	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 (scelnetOpen, scelnetClose, scelnetRecv, scelnetRecvFrom, scelnetSend, or scelnetSendTo) 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.

scelnetAbortLog

Abort acquisition of log

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.4	October 11, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetAbortLog(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 scelnetGetLog() using scelNETE_ABORT.

Return value

0 Normal termination

>0 Number of times errors occurred for SetEventFlag()

scelnetAddress2Name

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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetAddress2Name(
```

<code>int flags,</code>	Conversion flags (always 0)
<code>char *buf,</code>	Buffer address for storing conversion results
<code>int len,</code>	Buffer length (bytes)
<code>scelnetAddress_t *paddr</code>	Internal-format IP address
<code>int ms,</code>	Timeout interval (ms)
<code>int nretry);</code>	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 (scelNETE_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetAddress2String

Conversion from internal-format IP address to dot format

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetAddress2String(
```

```
    char *buf,                Buffer address for storing conversion results
```

```
    int len,                  Buffer length (bytes)
```

```
    scelnetAddress_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.

scelnetChangeThreadPriority

Change thread priority

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetChangeThreadPriority(
```

```
    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, `scelNETE_OK` (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetClose

Close Connection

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetClose(
```

<code>int cid,</code>	Connection ID
<code>int ms);</code>	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 `scelnetClose()` is called. No subsequent processing can be performed by using that Connection ID, including `scelnetAbort()`.

Return value

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

scelnetControl

Control Connection

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetControl(
```

<code>int cid,</code>	Connection ID to be controlled
<code>int code,</code>	Control code representing operation contents
<code>void *ptr,</code>	Starting address of data area
<code>int len)</code>	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
scelnetC_CODE_GET_INFO	1	Get Connection information
scelnetC_CODE_GET_FLAGS	2	Get control flags
scelnetC_CODE_SET_FLAGS	3	Set control flags

For details, see the "Control Codes" section.

Return value

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

scelnetCreate

Create Connection

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetCreate(  
    scelnetParam_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 scelnetOpen() is called.

Return value

Connection ID

scelnetGetInterfaceList

Get list of network interfaces

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetGetInterfaceList(
```

```
    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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetGetLog

Get log

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	October 11, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetGetLog(
```

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 `scelnetAbortLog()`.

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

Return value

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

scelnetGetNameServers

Get name server information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetGetNameServers(
```

```
    scelnetAddress_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(scelnetAddress_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.

scelnetGetRoutingTable

Get routing control table

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetGetRoutingTable(
    scelnetRoutingEntry_t *p,
    int n);
```

Pointer to area for getting routing control tables

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 $\text{sizeof}(\text{scelnetRoutingEntry_t}) * n$ bytes.

Return value

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

scelnetInterfaceControl

Reference or set parameters of network interface

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    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 scelnetDevOps_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
scelnetCC_GetInterfaceName	0x00000000	Get interface name (string)
scelnetCC_GetModuleName	0x00000001	Get module name (string)
scelnetCC_GetVendorName	0x00000002	Get vendor name (string)
scelnetCC_GetDeviceName	0x00000003	Get device name (string)
scelnetCC_GetBusType	0x00000004	Get bus type
scelnetCC_GetBusLoc	0x00000005	Get device location information
scelnetCC_GetProt	0x00000006	Get NETDEV layer protocol version
scelnetCC_GetImpl	0x00000007	Get NETDEV layer implementation version
scelnetCC_GetFlags	0x00000008	Get flags (flags) in scelnetDevOps structure that NETDEV layer has
scelnetCC_GetAddress	0x00000009	Get IP address of interface
scelnetCC_GetNetmask	0x0000000a	Get subnet mask of interface
scelnetCC_GetBroadcast	0x0000000b	Get broadcast address of interface
scelnetCC_GetMTU	0x0000000c	Get MTU value of interface
scelnetCC_GetHWaddr	0x0000000d	Get hardware address of interface
scelnetCC_GetDHCPHostName	0x0000000e	Get DHCP host name
scelnetCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetName2Address

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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetName2Address(
```

<code>int flags,</code>	Conversion flag
<code>scelnetAddress_t *paddr,</code>	Address of structure variable for receiving internal-format IP address
<code>char *name,</code>	Dot-format IP address or host name
<code>int ms,</code>	Timeout interval (ms)
<code>int nretry);</code>	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
scelnetN2AF_NoDot	0x01	Dot-format conversion is prohibited
scelnetN2AF_NoHosts	0x02	(Reserved)
scelnetN2AF_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 scelnetN2AF_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 scelnetN2AF_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)
- | | |
|--------------|--|
| <i>num8</i> | Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 8 bits |
| <i>num16</i> | Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 16 bits |
| <i>num24</i> | Octal, decimal, or hexadecimal number in the range that can be represented by unsigned 24 bits |
| <i>num32</i> | 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, `scelNETE_OK (=0)` is returned. When an error occurs, an error code (negative value) is returned.

scelnetOpen

Establish a Connection

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetOpen(
```

<code>int cid,</code>	Connection ID
<code>int ms);</code>	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 Connection is established (=Established).
- An error occurred, excluding a timeout.
- The specified timeout interval (*ms*) elapsed (scelNETE_TIMEOUT).

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

Return value

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

scelnetRecv

Receive data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetRecv(
```

<code>int cid,</code>	Connection ID
<code>void *ptr,</code>	Buffer address for storing receive data
<code>int count,</code>	Data size to be received (bytes)
<code>int *pflags,</code>	Address of variable for storing status flags
<code>int ms);</code>	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
scelNETF_URG	0x02	A segment with an Urgent flag was received
scelNETF_FIN	0x04	A segment with a Fin flag was received
scelNETF_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. [scelNETF_FIN]
5. A segment with an Urgent flag was received. [scelNETF_URG]
6. The timeout interval specified by *ms* elapsed. [scelNETF_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.

scelnetRecvFrom

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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetRecvFrom(
```

<code>int cid,</code>	Connection ID
<code>void *ptr,</code>	Buffer address for storing reception data
<code>int count,</code>	Data size to be received (bytes)
<code>int *pflags,</code>	Address of variable for storing status flags
<code>scelnetAddress_t *iadr,</code>	Address of variable for storing IP address of sender
<code>int *port,</code>	Address of variable for storing port
<code>int ms);</code>	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 `scelnetRecv()`, but it also stores the IP address of the sender in **iadr* and the port of the sender in **port*. This function can only be used for a UDP connection. If it is used for a TCP connection, an error (`scelNETE_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 `scelnetParam` or `remote_port` is set to `scelNETP_ANY` when the Connection is created. `scelnetRecvFrom()` 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.

scelnetSend

Send data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetSend(
```

<code>int cid,</code>	Connection ID
<code>void *ptr,</code>	Buffer address where send data was stored
<code>int count,</code>	Size of send data (bytes)
<code>int *pflags,</code>	Address of variable for storing status flags
<code>int ms);</code>	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 `scelNETF_URG` bit can be specified in **pflags*. When the `scelNETF_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, `scelnetSend()` returns the number of transmitted bytes. Whether or not a timeout occurred, can be determined by the `scelNETF_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 `scelNETCC_CODE_SET_FLAGS` control code. For details, see `scelnetControl()`.

Return value

The number of bytes that were sent is returned.

scelnetSendTo

Send data to specified IP address and port number

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetSendTo(
```

<code>int <i>cid</i>,</code>	Connection ID
<code>void *<i>ptr</i>,</code>	Buffer address where send data was stored
<code>int <i>count</i>,</code>	Size of send data (bytes)
<code>int <i>flags</i>,</code>	Status flags
<code>scelnetAddress_t *<i>iadr</i>,</code>	Address of variable for storing IP address of send destination
<code>int <i>port</i>,</code>	Port
<code>int <i>ms</i>);</code>	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 `scelnetSend()`, 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 (`scelNETE_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 `scelNETP_ANY` in `scelnetParam` when `scelnetCreate()` is called. `scelnetSendTo()` 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 `scelNETP_ANY`, an error (`scelNETE_INVALID_ARGUMENT`) will occur.

Return value

The number of bytes that were sent is returned.

Control Codes for Connection Control

scelnetC_CODE_GET_FLAGS

Get control flags

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
int scelnetControl(  
    int cid,                                Connection ID for which operation is to be performed  
    scelNETC_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 scelnetC_CODE_SET_FLAGS.

Return value

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

scelnetC_CODE_GET_INFO

Get Connection information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

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

Description

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

scelnetC_CODE_SET_FLAGS

Set control flags

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
int scelnetControl(
    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

scelnetCC_GetAddress

Get IP address

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetInterfaceControl(
```

```
    int interface_id,           Network interface ID for which operation is to be
                                performed
```

```
    scelnetCC_GetAddress, // 0x00000009
```

```
    void *ptr,                 Starting address of data area
```

```
    int len)                   Size of data area (sizeof(scelnetAddress_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(scelnetAddress_t), an error will occur.

Return value

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

scelnetCC_GetBroadcast

Get broadcast address

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,                Network interface ID for which operation is to be
                                     performed
    scelnetCC_GetBroadcast, // 0x0000000b
    void *ptr,                       Starting address of data area
    int len)                         Size of data area (sizeof(scelnetAddress_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(scelnetAddress_t), an error will occur.

Return value

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

scelnetCC_GetBusLoc

Get device location information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax**#include <inetctl.h>****int scelnetInterfaceControl(**

int *interface_id*, Network interface ID for which operation is to be performed

scelnetCC_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 scelnetBus_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetBusType

Get bus type

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,                Network interface ID for which operation is to be
                                     performed
    scelnetCC_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 scelnetBus_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
scelnetBus_USB	1	USB ethernet
scelnetBus_NIC	5	HDD ethernet

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

Return value

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

scelnetCC_GetDeviceName

Get device name

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <netinet.h>
```

```
int scelnetInterfaceControl(
```

int <i>interface_id</i> ,	Network interface ID for which operation is to be performed
----------------------------------	---

scelnetCC_GetDeviceName, // 0x00000003

void *ptr,	Starting address of data area
-------------------	-------------------------------

int <i>len</i>)	Size of data area (bytes)
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

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.

scelnetCC_GetDHCPFlags

Get DHCP control flags

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
int interface_id,
                                Network interface ID for which operation is to be
                                performed
scelnetCC_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
scelnetDevDHCP_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.

scelnetCC_GetDHCPHostName

Get DHCP host name

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax**#include <inetctl.h>****int scelnetInterfaceControl(****int interface_id,**

Network interface ID for which operation is to be performed

scelnetCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetFlags

Get flags

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,           Network interface ID for which operation is to be
                                performed
    scelnetCC_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 scelnetDevOps 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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetHWaddr

Get hardware address

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
```

```
int scelnetInterfaceControl(
```

int <i>interface_id</i> ,	Network interface ID for which operation is to be performed
----------------------------------	---

scelnetCC_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.

scelnetCC_GetImpl

Get implementation version

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,           Network interface ID for which operation is to be
                                performed
    scelnetCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetInterfaceName

Get interface name

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <netinet.h>
```

```
int scelnetInterfaceControl(
```

int <i>interface_id</i> ,	Network interface ID for which operation is to be performed
----------------------------------	---

```
scelnetCC_GetInterfaceName, // 0x00000000
```

void *ptr,	Starting address of data area
-------------------	-------------------------------

int <i>len</i>	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.

scelnetCC_GetModuleName

Get module name

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,                Network interface ID for which operation is to be
                                     performed
    scelnetCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetMTU

Get MTU value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax**#include <inetctl.h>****int scelnetInterfaceControl(**

int *interface_id*, Network interface ID for which operation is to be performed

scelnetCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetNetmask

Get subnet mask

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,           Network interface ID for which operation is to be
                                performed
    scelnetCC_GetNetmask, // 0x0000000a
    void *ptr,                 Starting address of data area
    int len)                   Size of data area (sizeof(scelnetAddress_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(scelnetAddress_t), an error will occur.

Return value

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

scelnetCC_GetProt

Get protocol version

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,                Network interface ID for which operation is to be
                                     performed
    scelnetCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetCC_GetVendorName

Get vendor name

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
inet	2.2	April 16, 2001

Syntax

```
#include <inetctl.h>
int scelnetInterfaceControl(
    int interface_id,                Network interface ID for which operation is to be
                                     performed
    scelnetCC_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, scelNETE_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	3-3
sceModemOps	3-3
Modem Control Functions	3-5
sceModemRegisterDevice	3-5
sceModemUnregisterDevice	3-6
Modem Driver Layer Functions	3-7
control	3-7
recv	3-9
send	3-10
start	3-11
stop	3-12
Modem Driver Layer Common Control Codes	3-13
sceModemCC_FLUSH_RXBUF	3-13
sceModemCC_FLUSH_TXBUF	3-14
sceModemCC_GET_DIALCONF	3-15
sceModemCC_GET_IF_TYPE	3-16
sceModemCC_GET_RX_COUNT	3-17
sceModemCC_GET_THPRI	3-18
sceModemCC_GET_TX_COUNT	3-19
sceModemCC_SET_THPRI	3-20
Modem Driver Layer Common Control Codes (for serial devices)	3-21
sceModemCC_GET_BO_COUNT	3-21
sceModemCC_GET_FE_COUNT	3-22
sceModemCC_GET_LINE	3-23
sceModemCC_GET_OE_COUNT	3-24
sceModemCC_GET_PARAM	3-25
sceModemCC_GET_PE_COUNT	3-28
sceModemCC_SET_BREAK	3-29
sceModemCC_SET_LINE	3-30
sceModemCC_SET_PARAM	3-31

Modem Information Structure

sceModemOps

Modem information structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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
<i>sceModemBus_Unknown</i>	0	Unknown. (This setting is not recommended)
<i>sceModemBus_USB</i>	1	USB device
<i>sceModemBus_1394</i>	2	(Reserved)
<i>sceModemBus_PCMCIA</i>	3	(Reserved)
<i>sceModemBus_PSEUDO</i>	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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 - 0xdfffffff	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.

Receive function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
modem	2.2	March 26, 2000

```
#include <modem.h>
```

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

Can be called from a thread.

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

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.

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.

The number of transferred bytes is returned.

send

Send function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
modem	2.2	March 26, 2000

Syntax

```
#include <modem.h>
```

```
int start(
```

```
    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 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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 receive buffer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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
<code>sceINETE_INVALID_ARGUMENT</code>	Invalid argument
<code>sceINETE_INVALID_CALL</code>	Invalid control code
-600 to -649	Modem layer-dependent error code

sceModemCC_GET_IF_TYPE

Get device type

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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
<code>sceModemIFT_GENERIC</code>	0x00000000	Other than <code>sceModemIFT_SERIAL</code>
<code>sceModemIFT_SERIAL</code>	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
<code>sceINETE_INVALID_ARGUMENT</code>	Invalid argument
<code>sceINETE_INVALID_CALL</code>	Invalid control code
-600 to -649	Modem layer-dependent error code

sceModemCC_GET_RX_COUNT

Get receive data size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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
<code>sceINETE_INVALID_ARGUMENT</code>	Invalid argument
<code>sceINETE_INVALID_CALL</code>	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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 (<i>len</i> 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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 (<i>len</i> 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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 (<i>len</i> is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to -649	Modem layer-dependent error code

sceModemCC_GET_PARAM

Get serial communication parameter

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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	0x003ffff	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_PAREN	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
<code>sceModemPARAM_STOP0</code>	0 stop bits
<code>sceModemPARAM_STOP1</code>	1 stop bit
<code>sceModemPARAM_STOP1H</code>	1.5 stop bits
<code>sceModemPARAM_STOP2</code>	2 stop bits

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

Table 3-19

Constant	Meaning
<code>sceModemPARAM_CS5</code>	5 bits/char
<code>sceModemPARAM_CS6</code>	6 bits/char
<code>sceModemPARAM_CS7</code>	7 bits/char
<code>sceModemPARAM_CS8</code>	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	<code>sceModemPARAM_PARODD</code>	<code>sceModemPARAM_PARENb</code>
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 (<i>len</i> 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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 (<i>len</i> is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to -649	Modem layer-dependent error code

sceModemCC_SET_BREAK

Send break

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 (<i>len</i> 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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 (<i>len</i> is not 4)
sceINETE_INVALID_CALL	Invalid control code
-600 to -649	Modem layer-dependent error code

sceModemCC_SET_PARAM

Set serial communication parameter

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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
<code>sceINETE_INVALID_ARGUMENT</code>	Invalid argument (<code>len</code> is not 4, a parameter that cannot be set was assigned)
<code>sceINETE_INVALID_CALL</code>	Invalid control code
-600 to -649	Modem layer-dependent error code

Chapter 4: Network Device Library

Table of Contents

Structures	4-4
scelnetDevOps_t	4-4
scelnetPkt_t	4-8
scelnetPktQ_t	4-10
Library Functions	4-11
scelnetAllocMem	4-11
scelnetAllocPkt	4-12
scelnetFreeMem	4-13
scelnetFreePkt	4-14
scelnetPktDeQ	4-15
scelnetPktEnQ	4-16
scelnetPrintf	4-17
scelnetRand	4-18
scelnetRegisterNetDevice	4-19
scelnetUnregisterNetDevice	4-20
Functions Implemented in the NETDEV Layer	4-21
control	4-21
start	4-22
stop	4-23
xmit	4-24
NETDEV Layer Common Control Codes	4-25
scelnetNDCC_GET_IF_TYPE	4-25
scelnetNDCC_GET_RX_BYTES	4-26
scelnetNDCC_GET_RX_DROPPED	4-27
scelnetNDCC_GET_RX_ERRORS	4-28
scelnetNDCC_GET_RX_PACKETS	4-29
scelnetNDCC_GET_THPRI	4-30
scelnetNDCC_GET_TX_BYTES	4-31
scelnetNDCC_GET_TX_DROPPED	4-32
scelnetNDCC_GET_TX_ERRORS	4-33
scelnetNDCC_GET_TX_PACKETS	4-34
scelnetNDCC_SET_THPRI	4-35
PPP Layer Control Codes	4-36
scePPPCC_GetAllowAccmNego	4-36
scePPPCC_GetAllowAccmValue	4-37
scePPPCC_GetAllowAccNego	4-38
scePPPCC_GetAllowAddressNego	4-39
scePPPCC_GetAllowAuth	4-40
scePPPCC_GetAllowDNS1	4-41
scePPPCC_GetAllowDNS1Nego	4-42
scePPPCC_GetAllowDNS2	4-43
scePPPCC_GetAllowDNS2Nego	4-44
scePPPCC_GetAllowIpAddress	4-45
scePPPCC_GetAllowIpMask	4-46
scePPPCC_GetAllowMagicNego	4-47

scePPPC_GetAllowMruNego	4-48
scePPPC_GetAllowMruValue	4-49
scePPPC_GetAllowPrcNego	4-50
scePPPC_GetAllowVjcompNego	4-51
scePPPC_GetAnyDial	4-52
scePPPC_GetAuthKey	4-53
scePPPC_GetAuthName	4-54
scePPPC_GetChapType	4-55
scePPPC_GetChatDial	4-56
scePPPC_GetChatLogin	4-57
scePPPC_GetConnectTimeout	4-58
scePPPC_GetCurrentStatus	4-59
scePPPC_GetDialingType	4-60
scePPPC_GetHisAddress	4-61
scePPPC_GetHisMask	4-62
scePPPC_GetIdleTimeout	4-63
scePPPC_GetIpcpTimeout	4-64
scePPPC_GetLcpEchoTimeout	4-65
scePPPC_GetLcpTimeout	4-66
scePPPC_GetLogFlags	4-67
scePPPC_GetOmitEmptyFrame	4-68
scePPPC_GetPeerKey	4-69
scePPPC_GetPeerName	4-70
scePPPC_GetPhoneNumber	4-71
scePPPC_GetPulseDial	4-72
scePPPC_GetToneDial	4-73
scePPPC_GetWantAccmNego	4-74
scePPPC_GetWantAccmValue	4-75
scePPPC_GetWantAccNego	4-76
scePPPC_GetWantAddressNego	4-77
scePPPC_GetWantAuth	4-78
scePPPC_GetWantDNS1	4-79
scePPPC_GetWantDNS1Nego	4-80
scePPPC_GetWantDNS2	4-81
scePPPC_GetWantDNS2Nego	4-82
scePPPC_GetWantIpAddress	4-83
scePPPC_GetWantIpMask	4-84
scePPPC_GetWantMagicNego	4-85
scePPPC_GetWantMruNego	4-86
scePPPC_GetWantMruValue	4-87
scePPPC_GetWantPrcNego	4-88
scePPPC_GetWantVjcompNego	4-89

Structures

scelnetDevOps_t

Network interface structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Structure

```
struct scelnetDevOps {
    struct scelnetDevOps *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 scelnetPktQ rcvq; /* receive queue */
    struct scelnetPktQ 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 */
} scelnetDevOps_t;
```

Description

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

It is set to "ethX" for an Ethernet interface or "pppX" 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 4-1

Constant	Value	Meaning
scelnetBus_Unknown	0	Unknown (This setting is not recommended)
scelnetBus_USB	1	USB device
scelnetBus_1394	2	(Reserved)
scelnetBus_PCMCIA	3	(Reserved)
scelnetBus_PSEUDO	4	Pseudo device
scelnetBus_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 4-2

Constant	Value	Meaning
scelNetDevF_Up	0x0001	Interface is up
scelNetDevF_Running	0x0002	Interface is available
scelNetDevF_Broadcast	0x0004	Broadcast address has been explicitly set
scelNetDevF_ARP	0x0010	Module is an Ethernet module
scelNetDevF_DHCP	0x0020	DHCP is to be used
scelNetDevF_PPP	0x0040	Module is a PPP module
scelNetDevF_NIC	0x0080	(Reserved)
scelNetDevF_Error	0x0100	Error
scelNetDevF_PPPE	0x0200	(Reserved)

The only bit among these that the INET layer must set is scelNetDevF_ARP for Ethernet or scelNetDevF_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 4-3

Constant	Value	Meaning
scelNetDevEFP_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.
scelNetDevEFP_PlugOut	0x00000002	Notification that data cannot be exchanged between the NETDEV layer and the device because the device was unplugged from the connector.
scelNetDevEFP_Recv	0x00000004	Notification that the received packet was added to the receive packet queue.
scelNetDevEFP_Error	0x00000010	Notification that an error (excluding a timeout) occurred for which send/receive could not continue for a reason other than a PlugOut.
scelNetDevEFP_TimeOut	0x00000020	Notification that a timeout error occurred for which send/receive could not continue for a reason other than a PlugOut.
scelNetDevEFP_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 scelNetPktQ and scelNetPkt.

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 4-4

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

scelnetRegisterNetDevice()

scelnetPkt_t

Packet structure

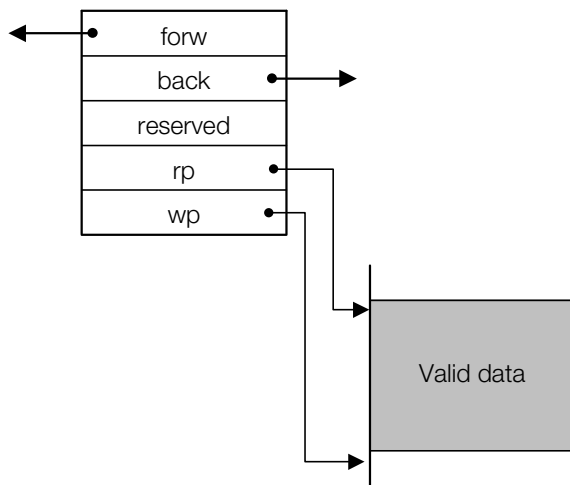
<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Structure

```
typedef struct scelnetPkt {
    struct scelnetPkt *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)
} scelnetPkt_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 4-1

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 `scelnetFreePkt()` 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()`

scelnetPktQ_t

Packet queue structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Structure

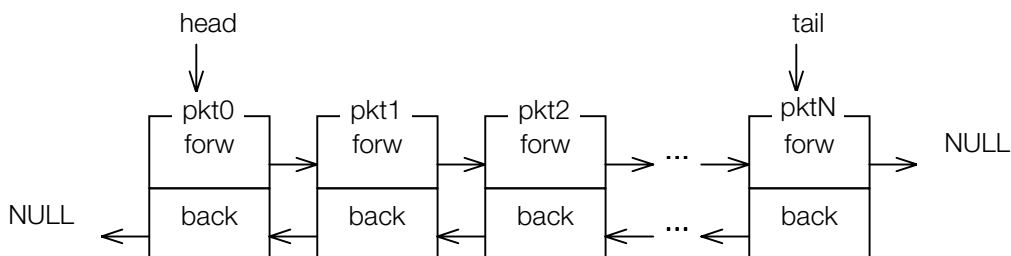
```
typedef struct scelnetPktQ {
    struct scelnetPkt *head;           Pointer to starting packet
    struct scelnetPkt *tail;           Pointer to final packet
} scelnetPktQ_t;
```

Description

Multiple packets are queued within the NETDEV layer with a packet queue having forward and backward links. scelnetPktQ 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 4-2

Library Functions

scelnetAllocMem

Allocate memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
```

```
void *scelnetAllocMem(
```

```
    struct scelnetDevOps *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.

scelnetAllocPkt() and scelnetFreePkt() should be used to allocate and free packet areas.

Besides this scelnetAllocMem() 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

scelnetAllocPkt

Allocate packet memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

#include <inet.h>

struct scelnetPkt *scelnetAllocPkt(**struct scelnetDevOps *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.

scelnetFreeMem

Free memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
```

```
void scelnetFreeMem(
```

```
    struct scelnetDevOps *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 scelnetAllocMem().

Return value

None.

scelnetFreePkt

Free packet memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
```

```
void scelnetFreePkt(
```

```
    struct scelnetDevOps *ops,           Pointer to network interface structure
```

```
    struct scelnetPkt *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 scelnetFreePkt() function, not the FreeSysMemory() or scelnetAllocMem() function.

Return value

None.

scelnetPktDeQ

Extract packet from packet queue

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
struct scelnetPkt *scelnetPktDeQ(
    struct scelnetPktQ *que);          Packet queue
```

Calling conditions

Can be called from a thread.

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

DescriptionThis 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.

scelnetPktEnQ

Add packet to packet queue

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

#include <inet.h>

void scelnetPktEnQ(

struct scelnetPktQ <i>*que</i> ,	Pointer to packet queue
struct scelnetPkt <i>*pkt</i>);	Pointer to packet structure

Calling conditions

Can be called from a thread.

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

DescriptionThis 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.

scelnetPrintf

Record log

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetPrintf(
```

```
    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 scelnetGetLog().

Return value

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

scelnetRand

Generate random number

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
u_int scelnetRand(
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.

scelnetRegisterNetDevice

Register network interface

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetRegisterNetDevice(
```

```
    struct scelnetDevOps *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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetUnregisterNetDevice

Delete network interface registration

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
#include <inet.h>
```

```
int scelnetUnregisterNetDevice(
```

```
    struct scelnetDevOps *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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

Functions Implemented in the NETDEV Layer

control

Control function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 `scelnetInterfaceControl()` 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, `scelNETE_OK` (=0) is returned. When an error occurs, an error code (negative value) is returned.

start

Start use function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 `scelnetDevEFP_StartDone` in the `evfid` field of the `scelnetDevOps` 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, `scelNETE_OK` (=0) is returned. When an error occurs, an error code (negative value) is returned.

stop

Stop use function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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 `sclnetDevEFP_PlugOut` is reported from the NETDEV layer to the INET layer.

In the latter case, `sclnetUnregisterNetDevice()` 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, `sclNETE_OK` (=0) is returned. When an error occurs, an error code (negative value) is returned.

xmit

Start transmission function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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, `scelNETE_OK` (=0) is returned. When an error occurs, an error code (negative value) is returned.

NETDEV Layer Common Control Codes

scelnetNDCC_GET_IF_TYPE

Get network interface type

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                                Network interface ID to be manipulated
    scelnetNDCC_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.

scelnetNDIFT_ETHERNET indicates that the NETDEV layer module is an Ethernet interface and, therefore, implicitly has the capability of returning Ethernet interface-dependent statistical information.

scelnetNDIFT_PPP indicates that the NETDEV layer module is a PPP interface. In the current specifications, no implicitly assumed capabilities are defined.

scelnetNDIFT_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 4-5

Constant	Value	Meaning
scelnetNDIFT_GENERIC	0x00000000	General
scelnetNDIFT_ETHERNET	0x00000001	Ethernet interface
scelnetNDIFT_PPP	0x00000002	PPP

scelnetNDCC_GET_RX_BYTES

Get number of receive bytes

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                                Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_RX_DROPPED

Get number of dropped receive packets

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                      Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_RX_ERRORS

Get number of receive errors

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                                Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_RX_PACKETS

Get number of receive packets

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                                Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_THPRI

Get thread priority

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                                Network interface ID to be manipulated
    scelnetNDCC_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, scelnetNDCC_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.

scelnetNDCC_GET_TX_BYTES

Get number of transmit bytes

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                      Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_TX_DROPPED

Get number of dropped transmit packets

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                      Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_TX_ERRORS

Get number of transmit errors

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                      Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_GET_TX_PACKETS

Get number of transmit packets

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                      Network interface ID to be manipulated
    scelnetNDCC_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, scelNETE_OK (=0) is returned. When an error occurs, an error code (negative value) is returned.

scelnetNDCC_SET_THPRI

Set thread priority

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.2	March 26, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                      Network interface ID to be manipulated
    scelnetNDCC_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 scelnetNDCC_SET_THPRI depends on the NETDEV layer module.

Return value

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

PPP Layer Control Codes

scePPPC_C_GetAllowAccmNego

Get allow ACCM negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(  
    int id,                                Network interface ID of target  
    scePPPC_C_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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowAccmValue

Get allow ACCM value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```

int scelnetInterfaceControl(
    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 scelnetAddress) (=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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```

int scelnetInterfaceControl(
    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 scelnetAddress) (=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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowIpMask

Get allow subnet mask

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowMagicNego

Get allow MAGIC negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPC_GetAllowMruNego

Get allow MRU negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    int id,                                Network interface ID of target
    scePPPC_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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetAllowMruValue

Get allow MRU value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
  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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetConnectTimeout

Get connection timeout value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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.

scePPPC_GetCurrentStatus

Get status string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    int id,                      Network interface ID of target
    scePPPC_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.

scePPPC_GetDialingType

Get dialing type

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
  int id,                      Network interface ID of target
  scePPPC_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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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_GetlpcpTimeout

Get IPCP timeout value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    int id,                                Network interface ID of target
    scePPPCC_GetlpcpTimeout, // 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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetLcpEchoTimeout

Get LCP Keep-Alive timeout value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
  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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPC_GetLogFlags

Get log display contents specification

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    int id,                      Network interface ID of target
    scePPPC_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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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.

scePPPC_GetPeerKey

Get peer_key string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    int id,                                Network interface ID of target
    scePPPC_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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetPeerName

Get peer_name string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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.

scePPPC_GetPulseDial

Get pulse dialing string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
  int id,                                Network interface ID of target
  scePPPC_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.

scePPPC_GetToneDial

Get tone dialing string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceInetInterfaceControl(
    int id,                                Network interface ID of target
    scePPPC_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.

scePPPC_GetWantAccmNego

Get request ACCM negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    int id,                                Network interface ID of target
    scePPPC_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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAccNego

Get request ACC negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
  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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantAuth

Get request authentication method

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    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 scelnetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPC_GetWantIpMask

Get request subnet mask

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int scelnetInterfaceControl(
    int id,                                Network interface ID of target
    scePPPC_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 scelnetAddress) (=16), an error will occur.

Return value

When processing terminates normally, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantMagicNego

Get request MAGIC negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantMruNego

Get request MRU negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
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

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

scePPPCC_GetWantVjcompNego

Get request VJCOMP negotiation flag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
netdev	2.4	October 11, 2001

Syntax

```
int sceNetInterfaceControl(
    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, sceNETE_OK (=0) is returned.

If an error occurs, an error code (negative value) is returned.

