libnet Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceNetEpollData

epoll user data union

Definition

```
#include <net.h>
typedef union SceNetEpollData {
        void *ptr;
        int fd;
        SceUInt32 u32;
        SceUInt64 u64;
        struct SceNetEpollDataExt {
              SceNetId id;
              SceUInt32 u32;
        }ext;
} SceNetEpollData;
```

Members

ptr, *fd*, *u32*, *u64*	BSD-compatible user data
ext	Extended user data

Description

This union is used in epoll user data.

Using a user data area of an event associated by `sceNetEpollControl()` makes it easier to describe processing when an output event occurs.

This area holds the contents set by applications. Therefore, for example, by setting a socket ID and a flag for applications to *ext.id* and *ext.u32* of the user data respectively, it will be possible for an output event to refer to the user data. In other words, whether to use this 64-bit area or not is an option and the decision depends on an application. In addition, the system will not perform any processing by referring to this area.

Applications need not to hold this area after associating the area by `sceNetEpollControl()`. In the case of associating an area with more than one socket by `sceNetEpollControl()`, `SceNetEpollEvent` can be reused.

See Also

`SceNetEpollEvent`

SceNetEpollEvent

epoll event structure

Definition

```
#include <net.h>
typedef struct SceNetEpollEvent {
        SceUInt32 events;
        SceUInt32 reserved;
        SceNetEpollSystemData system;
        SceNetEpollData data;
} SceNetEpollEvent;
```

Members

events	Events to be checked (input) or check result (output) events
reserved	Always 0
system	System data (Use with 0 clear, reference disabled)
data	User data

Description

This structure is used when setting check-target events (input) with `sceNetEpollControl()` or when receiving check result events (output) with `sceNetEpollWait()`.

The OR value of the following bit flags will be stored to *events*. To detect an output event, a corresponding input event needs to be specified beforehand. An event that does not require any input event, however, detects the event whenever an output event occurs.

Value (*events*)	Input	Output	Description
SCE_NET_EPOLLIN	●	●	The receive functions (`sceNetRecv()`, `sceNetRecvfrom()`, `sceNetRecvmsg()`, `sceNetAccept()`) can be called without entering the wait state.
SCE_NET_EPOLLOUT	●	●	The send functions (`sceNetSend()`, `sceNetSendto()`, `sceNetSendmsg()`) can be called without entering the wait state. Or `sceNetConnect()` has been completed.
SCE_NET_EPOLLERR	-	●	Socket error occurred. Details on the error can be obtained by a socket option `SCE_NET_SO_ERROR`.
SCE_NET_EPOLLHUP	-	●	Operation was aborted by application. (e.g. `sceNetSocketAbort()`)
SCE_NET_EPOLLDESCID	-	●	Event for DNS resolver occurred. (Input of `SCE_NET_EPOLLIN` must be specified to enter the event waiting state for DNS resolver. `SCE_NET_EPOLLIN` is output simultaneously)

*"-" in the input fields of the above table indicates where specification is not required.

data can refer to the input value given to `sceNetEpollControl()` as an output value of `sceNetEpollWait()` or `sceNetEpollWaitCB()`. Refer to the description of the `SceNetEpollData` structure for usage examples.

A `SCE_NET_EPOLLOUT` event will not occur when sending data with UDP or RAW.

`SCE_NET_EPOLLERR` and `SCE_NET_EPOLLHUP` are independent event of each other. In other words, `SCE_NET_EPOLLERR` does not include `SCE_NET_EPOLLHUP`, for example.

See Also

sceNetEpollControl(),sceNetEpollWait(),sceNetEpollWaitCB()

SceNetFdSet

Socket ID bit set structure

Definition

```
#include <net.h>
typedef SceUInt32 SceNetIdMask;
typedef struct SceNetFdSet {
        SceNetIdMask bits[8];
} SceNetFdSet;
```

Members

bits	Bits that represent a set of socket IDs

Description

This data structure specifies a set of socket IDs by bit flags.

The macros SCE_NET_FD_SET(), SCE_NET_FD_CLR(), SCE_NET_FD_ZERO(), and SCE_NET_FD_ISSET() are used for operation and evaluation, so it is not necessary to know the details of the structure.

See Also

SCE_NET_FD_SET(), SCE_NET_FD_CLR(), SCE_NET_FD_ZERO(), SCE_NET_FD_ISSET()

SceNetInAddr

IPv4 address structure

Definition

```
#include <net.h>
typedef SceUInt32 SceNetInAddr_t;
typedef struct SceNetInAddr {
        SceNetInAddr_t s_addr;
} SceNetInAddr;
```

Members

s_addr	IPv4 address

Description

This structure is used for holding the IPv4 address.

SceNetIovec

iovec structure

Definition

```
#include <net.h>
typedef struct SceNetIovec {
        void *iov_base;
        SceSize iov_len;
} SceNetIovec;
```

Members

iov_base	Base address (pointer)
iov_len	Size of area (in bytes) indicated by *iov_base*

Description

This structure is used by the scatter/gather array of the message header structure.

See Also

SceNetMsghdr

SceNetLinger

Linger structure

Definition

```
#include <net.h>
typedef struct SceNetLinger {
        int l_onoff;
        int l_linger;
} SceNetLinger;
```

Members

l_onoff	ON/OFF flag
l_linger	Linger time (seconds)

Description

This structure controls the termination process of a TCP connection. The linger time does not enable adjustment of the time for maintaining the TCP TIME_WAIT state.

SceNetIpMreq

IP multicast setting structure

Definition

```
#include <net.h>
typedef struct SceNetIpMreq {
        SceNetInAddr imr_multiaddr;
        SceNetInAddr imr_interface;
} SceNetIpMreq;
```

Members

imr_multiaddr	IP multicast group (network byte order)
imr_interface	Local IP address of interface (network byte order)

Description

This structure is used with SCE_NET_IP_ADD_MEMBERSHIP and SCE_NET_IP_DROP_MEMBERSHIP.

SceNetMsghdr

Message header structure

Definition

```
#include <net.h>
typedef struct SceNetMsghdr {
        void *msg_name;
        SceNetSocklen_t msg_namelen;
        SceNetIovec *msg_iov;
        int msg_iovlen;
        void *msg_control;
        SceNetSocklen_t msg_controllen;
        int msg_flags;
} SceNetMsghdr;
```

Members

msg_name	Pointer to address structure
msg_namelen	Size of address structure
msg_iov	Pointer to scatter/gather array
msg_iovlen	Number of elements in *msg_iov* array
msg_control	(unsupported)
msg_controllen	(unsupported)
msg_flags	(unsupported)

Description

This structure is used when sending and receiving data with sceNetSendmsg() and sceNetRecvmsg().

SceNetSockaddr

Socket address structure

Definition

```
#include <net.h>
typedef SceUChar8 SceNetSaFamily_t;
typedef struct SceNetSockaddr {
        SceUChar8 sa_len;
        SceNetSaFamily_t sa_family;
        char sa_data[14];
} SceNetSockaddr;
```

Members

sa_len	Address structure size
sa_family	Address family
sa_data	Protocol-dependent address

Description

This structure is used to pass a reference of the socket address structure for each protocol family.

SceNetSockaddrIn

Socket address structure

Definition

```
#include <net.h>
typedef SceUChar8 SceNetSaFamily_t;
typedef SceUShort16 SceNetInPort_t;
typedef struct SceNetSockaddrIn {
        SceUChar8 sin_len;
        SceNetSaFamily_t  sin_family;
        SceNetInPort_t sin_port;
        SceNetInAddr sin_addr;
        SceNetInPort_t sin_vport;
        SceChar8 sin_zero[6];
} SceNetSockaddrIn;
```

Members

sin_len	Address structure size
sin_family	Address family (`SCE_NET_AF_INET`)
sin_port	Port number (network byte order)
sin_addr	IPv4 address (network byte order)
sin_vport	v port number (network byte order)
sin_zero	Unused (always 0)

Description

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

Note that *sin_vport* is added.

The following is the relationship between port number and v port number.

Port number	TCP: TCP port number UDP: UDP port number UDPP2P: UDP port number TCP over UDPP2P: TCP port number
v port number	TCP: 0 UDP: 0 UDPP2P: Virtual port number TCP over UDPP2P: UDP port number

Datatypes (Extension)

SceNetDnsInfo

DNS address setting structure

Definition

```
#include <net.h>
#define SCE_NET_DNS_ADDR_MAX 2
typedef struct SceNetDnsInfo {
        SceNetInAddr dns_addr[SCE_NET_DNS_ADDR_MAX];
} SceNetDnsInfo;
```

Members

dns_addr	DNS address (network byte order)

Description

This structure is used to specify the DNS address on the application side.

When this feature is used, the DNS address used by the application is changed. Note that the application no longer uses the DNS address used by the system software for communication. The use of this function is not usually required.

Use libnetctl to obtain the DNS address.

See Also

`sceNetSetDnsInfo()`

SceNetEmulationParam

Network emulation parameter structure

Definition

```
#include <net.h>
typedef struct SceNetEmulationParam {
        SceUShort16 version;
        SceUShort16 option_number;
        SceUShort16 current_version;
        SceUShort16 result;
        SceUInt32 flags;
#define SCE_NET_EMULATION_PARAM_FLAGS_BPS_LIMIT_SHORT_TIME 0x00000001
#define SCE_NET_EMULATION_PARAM_FLAGS_API                  0x00000100
#define SCE_NET_EMULATION_PARAM_FLAGS_DEBUG                0x00000200
#define SCE_NET_EMULATION_PARAM_FLAGS_HOSTTOOL             0x00000400
        SceUInt32 reserved1;
        SceNetEmulationData send;
        SceNetEmulationData recv;
        SceUInt32 seed;
        SceUChar8 reserved[44];
} SceNetEmulationParam;

typedef struct SceNetEmulationData {
        SceUShort16 drop_rate;
        SceUShort16 drop_duration;
        SceUShort16 pass_duration;
        SceUShort16 delay_time;
        SceUShort16 delay_jitter;
        SceUShort16 order_rate;
        SceUShort16 order_delay_time;
        SceUShort16 duplication_rate;
        SceUInt32 bps_limit;
        SceUShort16 lower_size_limit;
        SceUShort16 upper_size_limit;
        SceUInt32 system_policy_pattern;
        SceUInt32 game_policy_pattern;
        SceUShort16 policy_flags[64];
        SceUChar8 reserved[64];
} SceNetEmulationData;
```

Members

See "Emulation Parameters" in the "Network Emulation" chapter of the "libnet Overview" document.

The following values are set to *flags* when the network emulation parameters are obtained.

Value	Description
SCE_NET_EMULATION_PARAM_FLAGS_API	Set with sceNetEmulationSet()
SCE_NET_EMULATION_PARAM_FLAGS_DEBUG	Set with ★**Debug Settings**
SCE_NET_EMULATION_PARAM_FLAGS_HOSTTOOL	Set with psp2ctrl utility

Description

This structure is used to set and obtain the network emulation parameters.

See Also

`sceNetEmulationSet(),sceNetEmulationGet()`

SceNetEtherAddr

Ethernet address structure

Definition

```
#include <net.h>
#define SCE_NET_ETHER_ADDR_LEN 6
typedef struct SceNetEtherAddr {
        SceUChar8 data[SCE_NET_ETHER_ADDR_LEN];
} SceNetEtherAddr;
```

Members

data	Ethernet address data

Description

This structure indicates the Ethernet address.

See Also

`sceNetEtherStrton(),sceNetEtherNtostr()`

SceNetInitParam

Initialization parameter structure

Definition

```
#include <net.h>
typedef struct SceNetInitParam {
        void *memory;
        int size;
        int flags;
} SceNetInitParam;
```

Members

memory	Memory address to be used by libnet
size	Memory size to be used by libnet (Specify 16 KiB (16,384 bytes) or more; 32 KiB (32,768 bytes) or more is recommended. 48 KiB (49,152 bytes) or more is recommended for the ad hoc communication mode.)
flags	Flag (always 0)

Description

This structure stores information passed to the initialization function `sceNetInit()`.

The memory is used for internal dynamic memory allocation, and it is mainly broken down as follows. Determine the minimum available memory (check with `sceNetShowIfconfig()`) while leaving about 6 KiB.

- 4 KiB for future compatibility (consumed after calling the initialization function)
- 3 KiB for basic features of libnet
- Max. 4 KiB DNS cache
- 128 bytes/socket
- 1,300 bytes/DNS resolver

SceNetResolverParam

DNS resolver parameter structure

Definition

```
#include <net.h>
typedef void *(*SceNetResolverFunctionAllocate)(
        SceSize size,
        SceNetId rid,
        const char *name,
        void *user
);
typedef void (*SceNetResolverFunctionFree)(
        void *ptr,
        SceNetId rid,
        const char *name,
        void *user
);
typedef struct SceNetResolverParam {
        SceNetResolverFunctionAllocate allocate;
        SceNetResolverFunctionFree free;
        void *user;
} SceNetResolverParam;
```

Members

size	Size required for memory allocation
rid	Resolver ID (hint)
name	Name (hint)
ptr	Address of memory to be freed
allocate	Dynamic memory allocation function
free	Free memory allocated with *allocate*
user	User data

Description

This structure stores information passed to `sceNetResolverCreate()`.

Use `allocate()` to allocate the memory including the specified *size*, and have the address (pointer) indicating the allocated memory be the return value. If the memory cannot be allocated, have NULL be the return value. In addition, use a 4-byte alignment for the address. *rid* passes the resolver ID established when `sceNetResolverCreate()` ends normally. This *rid* assists in associating the memory allocated by the host side application with the resolver context, and *rid* cannot be used to operate the resolver functions at this time. *name* passes the *name* of `sceNetResolverCreate()` as a hint.

`free()` is used to pass the address specified with `allocate()` to *ptr*, so free the memory. *rid* is similar to *name* in that it is a hint, and *rid* cannot be used to operate the resolver functions at this time.

You cannot set either `allocate()` or `free()` (only one of the two) to NULL.

During the execution of `allocate()` and `free()`, libnet functions from different threads or other library functions that use libnet enter the wait state, so ensure that `allocate()` and `free()` are completed within a reasonable amount of time.

Setting `allocate()` and `free()` and using these functions allows handling of the memory to be used internally by the DNS resolver. If both are set to NULL, the memory specified with `sceNetInit()` will be used.

SceNetSockInfo

Structure for receiving/passing connection information

Definition

```
#include <net.h>
#define SCE_NET_DEBUG_NAME_LEN_MAX 31
typedef struct SceNetSockInfo {
        char name[SCE_NET_DEBUG_NAME_LEN_MAX + 1];
        ScePid pid;
        SceNetId s;
        SceInt8 socket_type;
        SceInt8 policy;
        SceInt16 reserved16;
        int recv_queue_length;
        int send_queue_length;
        SceNetInAddr local_adr;
        SceNetInAddr remote_adr;
        SceNetInPort_t local_port;
        SceNetInPort_t remote_port;
        SceNetInPort_t local_vport;
        SceNetInPort_t remote_vport;
        int state;
        int flags;
        int reserved[8];
} SceNetSockInfo;
```

Members

name	(Unused)
pid	(Unused)
s	Socket ID, DNS resolver ID
socket_type	Socket type (`SCE_NET_SOCK_*`) (Note) The value 11 indicates I/O multiplexing (epoll ID). It is not an operable socket; however, the member *s* can be referenced to check the created epoll ID for debug purposes.
policy	Policy number to be used in network emulation
reserved16	Reserved
recv_queue_length	Number of data bytes in receive buffer
send_queue_length	Number of data bytes in send buffer
local_adr	Local address (network byte order)
remote_adr	Remote address (network byte order)
local_port	Local port number (network byte order)
remote_port	Remote port number (network byte order)
local_vport	Local v port number (network byte order)
remote_vport	Remote v port number (network byte order)
state	Connection state
flags	Flags
reserved	Reserved

Description

This structure is used to obtain information regarding connections.

See the description of `SceNetSockaddrIn` for the relationship between port number and v port number.

One of the following values will be stored for *`state`*.

Value	(Number)	Description (Corresponding Protocol)
`SCE_NET_SOCKINFO_STATE_UNKNOWN`	0	State unknown
`SCE_NET_SOCKINFO_STATE_CLOSED`	1	Closed
`SCE_NET_SOCKINFO_STATE_OPENED`	2	Opened
`SCE_NET_SOCKINFO_STATE_LISTEN`	3	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_SYN_SENT`	4	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_SYN_RECEIVED`	5	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_ESTABLISHED`	6	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_FIN_WAIT_1`	7	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_FIN_WAIT_2`	8	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_CLOSE_WAIT`	9	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_CLOSING`	10	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_LAST_ACK`	11	Internal state of TCP and TCP over UDPP2P
`SCE_NET_SOCKINFO_STATE_TIME_WAIT`	12	Internal state of TCP and TCP over UDPP2P

The OR value of the following bit flags will be stored to *`flags`*.

Value	Description
`SCE_NET_SOCKINFO_F_SELF`	Socket created by own process
`SCE_NET_SOCKINFO_F_KERNEL`	Socket created by kernel
`SCE_NET_SOCKINFO_F_OTHERS`	Socket created by separate process
`SCE_NET_SOCKINFO_F_RECV_WAIT`	Receive wait socket
`SCE_NET_SOCKINFO_F_SEND_WAIT`	Send wait socket
`SCE_NET_SOCKINFO_F_RECV_EWAIT`	`sceNetEpollWait()` and `sceNetEpollWaitCB()` receive wait socket
`SCE_NET_SOCKINFO_F_SEND_EWAIT`	`sceNetEpollWait()` and `sceNetEpollWaitCB()` send wait socket
`SCE_NET_SOCKINFO_F_WAKEUP_SIGNAL`	Intermittent connection wakeup signal

SceNetStatisticsInfo

Statistics information structure

Definition

```
#include <net.h>
typedef struct SceNetStatisticsInfo {
        int kernel_mem_free_size;
        int kernel_mem_free_min;
        int packet_count;
        int packet_qos_count;
        int libnet_mem_free_size;
        int libnet_mem_free_min;
} SceNetStatisticsInfo;
```

Members

kernel_mem_free_size	Number of currently free memory bytes (kernel)
kernel_mem_free_min	Minimum number of currently free memory bytes (kernel)
packet_count	Total number of packets held in kernel
packet_qos_count	Total number of QoS packets held in kernel
libnet_mem_free_size	Number of currently free memory bytes (library)
libnet_mem_free_min	Minimum number of currently free memory bytes (library)

Description

This structure is used to obtain statistics information.

Other Datatypes

SceNetIcmpHeader

ICMP header

Definition

```
#include <net.h>
typedef struct SceNetIcmpHeaderEcho {
        SceUShort16 id;
        SceUShort16 sequence;
} SceNetIcmpHeaderEcho;

typedef struct SceNetIcmpHeaderFrag {
        SceUShort16 unused;
        SceUShort16 mtu;
} SceNetIcmpHeaderFrag;

typedef union SceNetIcmpHeaderUnion {
        SceNetIcmpHeaderEcho echo;
        unsigned int gateway;
        SceNetIcmpHeaderFrag frag;
} SceNetIcmpHeaderUnion;

typedef struct SceNetIcmpHeader {
        SceUChar8 type;
#define SCE_NET_ICMP_TYPE_ECHO_REPLY           0
#define SCE_NET_ICMP_TYPE_DEST_UNREACH         3
#define SCE_NET_ICMP_TYPE_SOURCE_QUENCH        4
#define SCE_NET_ICMP_TYPE_REDIRECT             5
#define SCE_NET_ICMP_TYPE_ECHO_REQUEST         8
#define SCE_NET_ICMP_TYPE_TIME_EXCEEDED        11
#define SCE_NET_ICMP_TYPE_PARAMETER_PROBLEM    12
#define SCE_NET_ICMP_TYPE_TIMESTAMP_REQUEST    13
#define SCE_NET_ICMP_TYPE_TIMESTAMP_REPLY      14
#define SCE_NET_ICMP_TYPE_INFORMATION_REQUEST  15
#define SCE_NET_ICMP_TYPE_INFORMATION_REPLY    16
#define SCE_NET_ICMP_TYPE_ADDRESS_MASK_REQUEST 17
#define SCE_NET_ICMP_TYPE_ADDRESS_MASK_REPLY   18
        SceUChar8 code;
        /* DEST_UNREACH */
#define SCE_NET_ICMP_CODE_DEST_UNREACH_NET_UNREACH          0
#define SCE_NET_ICMP_CODE_DEST_UNREACH_HOST_UNREACH         1
#define SCE_NET_ICMP_CODE_DEST_UNREACH_PROTO_UNREACH        2
#define SCE_NET_ICMP_CODE_DEST_UNREACH_PORT_UNREACH         3
#define SCE_NET_ICMP_CODE_DEST_UNREACH_FRAG_AND_DF          4
#define SCE_NET_ICMP_CODE_DEST_UNREACH_SRC_HOST_FAILED      5
#define SCE_NET_ICMP_CODE_DEST_UNREACH_DST_NET_UNKNOWN      6
#define SCE_NET_ICMP_CODE_DEST_UNREACH_DST_HOST_UNKNOWN     7
#define SCE_NET_ICMP_CODE_DEST_UNREACH_SRC_HOST_ISOLATED    8
#define SCE_NET_ICMP_CODE_DEST_UNREACH_NET_ADMIN_PROHIBITED 9
#define SCE_NET_ICMP_CODE_DEST_UNREACH_NET_HOST_PROHIBITED  10
#define SCE_NET_ICMP_CODE_DEST_UNREACH_NET_TOS              11
#define SCE_NET_ICMP_CODE_DEST_UNREACH_HOST_TOS             12
#define SCE_NET_ICMP_CODE_TIME_EXCEEDED_TTL_EXCEEDED        0
#define SCE_NET_ICMP_CODE_TIME_EXCEEDED_FRT_EXCEEDED        1
        SceUShort16 checksum;
        SceNetIcmpHeaderUnion un;
} SceNetIcmpHeader;
```

Description

This is the ICMP header. For information on the members, refer to the reference material such as RFC for ICMP headers.

SceNetIpHeader

IP header

Definition

```
#include <net.h>
typedef struct SceNetIpHeaderIpVerHl {
        SceUChar8 hl:4;
        SceUChar8 ver:4;
} SceNetIpHeaderIpVerHl;

typedef union SceNetIpHeaderUnion {
        SceNetIpHeaderIpVerHl ip_ver_hl;
        SceUChar8 ver_hl;
} SceNetIpHeaderUnion;

typedef struct SceNetIpHeader {
        SceNetIpHeaderUnion un;
#define SCE_NET_IPVERSION   4
        SceUChar8 ip_tos;
        SceUShort16 ip_len;
        SceUShort16 ip_id;
        SceUShort16 ip_off;
#define SCE_NET_IP_RF      0x8000
#define SCE_NET_IP_DF      0x4000
#define SCE_NET_IP_MF      0x2000
#define SCE_NET_IP_OFFMASK 0x1fff
        SceUChar8 ip_ttl;
        SceUChar8 ip_p;
        SceUShort16 ip_sum;
        SceNetInAddr ip_src;
        SceNetInAddr ip_dst;
} SceNetIpHeader;
```

Description

This is the IP header. For information on the members, refer to the reference material such as RFC for IP headers.

Initialization / Termination Functions

sceNetInit

Initialize libnet

Definition

```
#include <net.h>
int sceNetInit(
        SceNetInitParam *param
)
```

Arguments

param Pointer to initialization parameter structure

Return Values

Value	Description
0	Normal termination
SCE_NET_ERROR_EBUSY	Initialization was not performed with the function call because libnet has already been initialized
SCE_NET_ERROR_EINVAL	Function called due to invalid argument or content

Other errors may be returned.

Description

This function initializes libnet.

It is not necessary to keep the actual structure indicated by *param* after completing the calling of this function. If libnet has not been initialized but a function requiring its initialization is called, that function will return SCE_NET_ERROR_ENOTINIT.

sce_net_errno is an invalid value.

Notes

This function is not multi-thread safe.

This function can be called again to re-initialize libnet after terminating it with sceNetTerm().

sceNetTerm

Terminate libnet

Definition

```
#include <net.h>
int sceNetTerm(void)
```

Arguments

None

Return Values

Value	Description
0	Normal termination
`SCE_NET_ERROR_ENOTINIT`	Not initialized

Description

This function terminates libnet.

`sce_net_errno` is an invalid value.

Notes

This function is not multi-thread safe.

Utility Functions

sceNetInetNtop

Convert address structure into numeric string address

Definition

```
#include <net.h>
const char *sceNetInetNtop(
        int af,
        const void *src,
        char *dst,
        SceNetSocklen_t size
);
```

Arguments

af	Address family (`SCE_NET_AF_INET`)
src	Pointer to area for storing address in network format (network byte order)
dst	Pointer to area for storing address expressed as a numeric string
size	Size of area pointed to by *dst* (`SCE_NET_INET_ADDRSTRLEN` (=16, NULL end) bytes or more)

Return Values

Value	Description
dst	Normal termination
NULL	Error

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

Value	Description
`SCE_NET_EAFNOSUPPORT`	Invalid address family

Description

This function converts the address information specified with *src* to a numerically expressed string in the address family specified with *af*. The conversion result is stored to the area specified with *dst* and *size*.

sceNetInetPton

Convert numeric string address into address structure

Definition

```
#include <net.h>
int sceNetInetPton(
        int af,
        const char *src,
        void *dst
);
```

Arguments

af	Address family (`SCE_NET_AF_INET`)
src	Address expressed as a string (only decimal addresses are valid)
dst	Pointer to area for storing address in network format (network byte order)

Return Values

Value	Description
Positive number	Normal termination
0	Invalid character string
Negative number	Error

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

Value	Description
`SCE_NET_EAFNOSUPPORT`	Invalid address family

Description

This function converts the address expressed as a string, *src*, to an address in network format, based on the address family *af*.

Sufficient space must be allocated to store a `SceNetInAddr` structure in *dst*.

sceNetHtonll

Convert 64-bit value byte order (from host to network)

Definition

```
#include <net.h>
SceUInt64 sceNetHtonll(
        SceUInt64 host64
);
```

Arguments

host64	Value for byte order conversion

Return Values

Value converted to byte order is returned.

Description

This function converts 64-bit value data from host byte order to network byte order.

sceNetHtonl

Convert 32-bit value byte order (from host to network)

Definition

```
#include <net.h>
SceUInt32 sceNetHtonl(
        SceUInt32 host32
);
```

Arguments

host32	Value for byte order conversion

Return Values

Value converted to byte order is returned.

Description

This function converts 32-bit value data from host byte order to network byte order.

sceNetHtons

Convert 16-bit value byte order (from host to network)

Definition

```
#include <net.h>
SceUInt16 sceNetHtons(
        SceUInt16 host16
);
```

Arguments

host16	Value for byte order conversion

Return Values

Value converted to byte order is returned.

Description

This function converts 16-bit value data from host byte order to network byte order.

sceNetNtohll

Convert 64-bit value byte order (from network to host)

Definition

```
#include <net.h>
SceUInt64 sceNetNtohll(
        SceUInt64 net64
);
```

Arguments

net64	Value for byte order conversion

Return Values

Value converted to byte order is returned.

Description

This function converts 64-bit value data from network byte order to host byte order.

sceNetNtohl

Convert 32-bit value byte order (from network to host)

Definition

```
#include <net.h>
SceUInt32 sceNetNtohl(
        SceUInt32 net32
);
```

Arguments

net32	Value for byte order conversion

Return Values

Value converted to byte order is returned.

Description

This function converts 32-bit value data from network byte order to host byte order.

sceNetNtohs

Convert 16-bit value byte order (from network to host)

Definition

```
#include <net.h>
SceUInt16 sceNetNtohs(
        SceUInt16 net16
);
```

Arguments

net16	Value for byte order conversion

Return Values

Value converted to byte order is returned.

Description

This function converts 16-bit value data from network byte order to host byte order.

Utility Functions (Extension)

sceNetEtherStrton

Convert string address into 48-bit Ethernet address

Definition

```
#include <net.h>
int sceNetEtherStrton(
        const char *str,
        SceNetEtherAddr *n
);
```

Arguments

str	String expressing an Ethernet address
n	Pointer to Ethernet address structure

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function converts a string expressing an Ethernet address (xx:xx:xx:xx:xx:xx) to a 48-bit Ethernet address.

sceNetEtherNtostr

Convert 48-bit Ethernet address into string address

Definition

```
#include <net.h>
int sceNetEtherNtostr(
        const SceNetEtherAddr *n,
        char *str,
        SceSize len
);
```

Arguments

n	Pointer to Ethernet address structure
str	Pointer to area for storing Ethernet address
len	Size of area for storing Ethernet address (SCE_NET_ETHER_ADDRSTRLEN (=18) bytes or more)

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content

Description

This function converts a 48-bit Ethernet address to a string expression (xx:xx:xx:xx:xx:xx).

sceNetGetMacAddress

Get MAC address

Definition

```
#include <net.h>
int sceNetGetMacAddress(
        SceNetEtherAddr *addr,
        int flags
);
```

Arguments

addr	Pointer to Ethernet address structure
flags	Flag (always 0)

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function gets the MAC address. This API can be called at any time if libnet has been prepared, and always returns the MAC address of Wi-Fi (wlan0). Therefore, note that this MAC address differs from the MAC address that is actually used for communication when performing development with USB ethernet, for example.

Furthermore, note that it is not guaranteed that a specific hardware can be roughly identified by, for example, using a value range of the MAC address, that is to say OUI (the vendor code indicated by the first three bytes of the address) for the title release, etc.

See Also

"libnetctl Reference", "OpenPSID Overview", "OpenPSID Reference"

Network Communication Functions

sceNetAccept

Get socket for which TCP connection was established

Definition

```
#include <net.h>
SceNetId sceNetAccept(
        SceNetId s,
        SceNetSockaddr *addr,
        SceNetSocklen_t *paddrlen
);
```

Arguments

s	Listening socket (sceNetBind() and sceNetListen() completed socket)
addr	Pointer to area for storing destination address structure
paddrlen	Pointer to area for storing size of *addr*

Return Values

If a normal termination occurs, the socket ID for the new client is returned.

If an error occurs, a negative value is returned.

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetSocketAbort()
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EWOULDBLOCK	Established connection does not exist (for non-blocking)
SCE_NET_EFAULT	Invalid argument
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EOPNOTSUPP	Invalid call for that socket
SCE_NET_ECONNABORTED	Connection was aborted
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_ECANCELED	Close processing was called for a socket that is in the wait condition and being executed
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function takes out the existing connection from the client while running as a TCP server and obtains the socket ID. At the same time, the address structure of the client is stored in **addr*, and the size is stored in **paddrlen*.

Before calling this function, store the size of the area pointed to by *addr* to **paddrlen*.

The timeout time can be set using one of the following two methods.

- Use the SCE_NET_SO_RCVTIMEO socket option
- Use sceNetEpollWait() / sceNetEpollWaitCB()

Notes

Note that you cannot wait for a connection from the other party with the disconnected state because of the introduction of intermittent connection (refer to the "Intermittent Connection and Intermittent Disconnection (Internet Communication Mode)" chapter in the "Network Overview" document).

Measures to reduce the number of warnings of cast to `SceNetSockaddr` are implemented in the actual header of this function.

sceNetBind

Bind address to socket

Definition

```
#include <net.h>
int sceNetBind(
        SceNetId s,
        const SceNetSockaddr *addr,
        SceNetSocklen_t addrlen
);
```

Arguments

s	Socket ID to which local address is be bound
addr	Pointer to local address structure
addrlen	Size of local address structure

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EACCES`	Attempted to use the port reserved by the system
`SCE_NET_EFAULT`	Invalid argument specified
`SCE_NET_EINVAL`	Function called due to invalid argument or content Socket already bound
`SCE_NET_EAFNOSUPPORT`	`((SceNetSockaddrIn*)`*`addr`*`)>`*`sin_family`* for `sceNetBind()` is invalid
`SCE_NET_EADDRINUSE`	`sceNetBind()` called for local port being used
`SCE_NET_EADHOC`	UDP or TCP was attempted in the ad hoc communication mode
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function binds a local address (IP address and port number) to a socket.

The local address specified with *`addr`* and *`addrlen`* is bound to socket *`s`*.

Notes

Measures to reduce the number of warnings of cast to `SceNetSockaddr` are implemented in the actual header of this function.

sceNetConnect

Connect to destination

Definition

```
#include <net.h>
int sceNetConnect(
        SceNetId s,
        const SceNetSockaddr *addr,
        SceNetSocklen_t addrlen
);
```

Arguments

s	Socket ID used for connection
addr	Pointer to local address structure
addrlen	Size of local address structure

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINTR`	Aborted by `sceNetSocketAbort()`
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EACCES`	Attempted to establish a connection to an invalid address (such as a broadcast address)
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EINPROGRESS`	Attempting to establish a connection (for non-blocking)
`SCE_NET_EALREADY`	Socket is already in use
`SCE_NET_EAFNOSUPPORT`	Address family of specified address is invalid
`SCE_NET_EADDRINUSE`	Specified address already in use
`SCE_NET_EADDRNOTAVAIL`	Invalid address specified
`SCE_NET_EISCONN`	Attempted to open an established connection (including the case that the function is called again in a non-blocking state)
`SCE_NET_EWOULDBLOCK`	Timeout occurred when establishing connection
`SCE_NET_ETIMEDOUT`	TCP resend timeout occurred
`SCE_NET_ECONNREFUSED`	Connection refused by destination
`SCE_NET_ERETURN`	libnetctl error was returned
`SCE_NET_EADHOC`	UDP or TCP was attempted in the ad hoc communication mode
`SCE_NET_EOPNOTSUPP`	Connection was attempted to a listening socket
`SCE_NET_ECANCELED`	Close processing was called for a socket that is in the wait condition and being executed
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to connect to a server on the client side.

Socket *s* is used to connect to the address indicated by *addr* and *addrlen*. For TCP, this is used to establish a connection. For UDP, the destination is specified, so the behavior is as though a connection were established.

The timeout time can be set using one of the following two methods.

- Use the SCE_NET_SO_RCVTIMEO socket option
- Use sceNetEpollWait() / sceNetEpollWaitCB()

The basic concept for the latter method is to wait for a multiplex I/O event through sceNetEpollWait() / sceNetEpollWaitCB() following the execution of sceNetConnect(). Similar to this method, when event waiting of multiplex I/O is executed on another thread first, the socket is linked to the multiplex I/O after confirming that sceNetConnect() is executed.

Notes

Measures to reduce the number of warnings of cast to SceNetSockaddr are implemented in the actual header of this function.

sceNetGetpeername

Get destination information of socket

Definition

```
#include <net.h>
int sceNetGetpeername(
        SceNetId s,
        SceNetSockaddr *addr,
        SceNetSocklen_t *paddrlen
);
```

Arguments

s	Socket ID for which information is to be obtained
addr	Pointer to area for storing address structure of destination host
paddrlen	Pointer to area for storing size of `addr`

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to obtain the destination address information of a socket.

The address structure of the destination host of socket *s* is stored to the area specified with *addr* and *paddrlen*.

Notes

Measures to reduce the number of warnings of cast to `SceNetSockaddr` are implemented in the actual header of this function.

sceNetGetsockname

Get local information of socket

Definition

```
#include <net.h>
int sceNetGetsockname(
        SceNetId s,
        SceNetSockaddr *addr,
        SceNetSocklen_t *paddrlen
);
```

Arguments

s	Socket ID for which information is to be obtained
addr	Pointer to area for storing local address structure of socket
paddrlen	Pointer to area for storing size of local address structure of socket

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to obtain the local address information of a socket.

The local address structure of socket *s* is stored to the area specified with *addr* and *paddrlen*.

Notes

Measures to reduce the number of warnings of cast to `SceNetSockaddr` are implemented in the actual header of this function.

sceNetGetsockopt

Get socket option

Definition

```
#include <net.h>
int sceNetGetsockopt(
        SceNetId s,
        int level,
        int optname,
        void *optval,
        SceNetSocklen_t *optlen
);
```

Arguments

s	Socket ID for which socket option is to be obtained
level	Socket option level
optname	Socket option name
optval	Pointer to area for storing socket option value
optlen	Pointer to area for storing size of socket option value

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EINVAL`	Invalid value specified
`SCE_NET_ENOPROTOOPT`	Invalid combination of specified level (*level*) and option (*optname*)
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to obtain the socket options, such as error information and buffer size.

The value of the socket option specified with *level* and *optname* of the socket specified with *s* is stored to the area specified with *optval* and *optlen*.

sceNetListen

Accept TCP connection

Definition

```
#include <net.h>
int sceNetListen(
        SceNetId s,
        int backlog
);
```

Arguments

s	Socket ID for which to perform TCP connection wait
backlog	Size of queue for accepting connections (number of pending connections)

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EOPNOTSUPP`	Socket type cannot accept connections
`SCE_NET_EADHOC`	UDP or TCP was attempted in the ad hoc communication mode
`SCE_NET_ECANCELED`	Close processing was called for a socket that is in the wait condition and being executed
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

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

backlog indicates the maximum length of the queue for accepting connections.

sceNetRecv

Receive data

Definition

```
#include <net.h>
int sceNetRecv(
        SceNetId s,
        void *buf,
        SceSize len,
        int flags
);
```

Arguments

s	ID of socket to receive data
buf	Pointer to area for storing receive data
len	Size of data to be received (bytes)
flags	Flags

The following values can be set to *flags*.

Value	Description
SCE_NET_MSG_DONTWAIT	Calls as non-blocking
SCE_NET_MSG_PEEK	Leaves receive data unchanged in receive buffer
SCE_NET_MSG_WAITALL	Blocks until specified buffer size is received
SCE_NET_MSG_PEEKLEN	Obtains size of received data

Return Values

Value	Description
0	FIN received (TCP)
Positive number	Size of received data
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by `sceNetSocketAbort()`
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	Socket is in blocking state (when non-blocking) Timeout occurred (when `SCE_NET_SO_RCVTIMEO` option is specified)
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_ENOTCONN	Connection not established
SCE_NET_ECONNABORTED	Connection was aborted
SCE_NET_ECANCELED	Close processing was called for a socket that is in the wait condition and being executed
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to receive data from a connected socket.

len bytes are received from socket *s*, and the received data is stored to the area specified with *buf*.

This function does not timeout implicitly.

When SCE_NET_MSG_WAITALL is specified to *flags*, the function returns a value when one of the following conditions is true.

- Data of the number of bytes specified with *len* has been received
- FIN is received (TCP only)
- A timeout, etc. has occurred

If SCE_NET_MSG_WAITALL has not been specified with TCP, the function may return a value when the size of the receive data has not reached the number of bytes specified with *len*. For UDP, the function will return a value regardless of the number of specified bytes as long as one packet is received.

When SCE_NET_MSG_PEEKLEN is specified to *flags*, specify NULL to *buf* and the maximum length to *len*. The point at which the size of the receive data is obtained is the same as for SCE_NET_MSG_PEEK. In other words, a buffer of the maximum size to be received with SCE_NET_MSG_PEEK is required in order to obtain the data size, but SCE_NET_MSG_PEEKLEN does not require this buffer.

SCE_NET_MSG_PEEK cannot be used together with SCE_NET_MSG_WAITALL.

sceNetRecvfrom

Receive data (with sender address)

Definition

```
#include <net.h>
int sceNetRecvfrom(
        SceNetId s,
        void *buf,
        SceSize len,
        int flags,
        SceNetSockaddr *addr,
        SceNetSocklen_t *paddrlen
);
```

Arguments

s	ID of socket to receive data
buf	Pointer to area for storing receive data
len	Size of data to be received (bytes)
flags	Flags
addr	Pointer to area for storing address structure of sending host
paddrlen	Pointer to area for storing size of address structure of sending host

The following values can be set to *flags*.

Value	Description
SCE_NET_MSG_DONTWAIT	Calls as non-blocking
SCE_NET_MSG_PEEK	Leaves receive data unchanged in receive buffer
SCE_NET_MSG_WAITALL	Blocks until specified buffer size is received
SCE_NET_MSG_PEEKLEN	Obtains size of received data

Return Values

Value	Description
Positive number	Size of received data
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetSocketAbort()
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	Socket is in blocking state (when non-blocking) Timeout occurred (when SCE_NET_SO_RCVTIMEO option is specified)
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_ENOTCONN	Connection not established
SCE_NET_ECANCELED	Close processing was called for a socket that is in the wait condition and being executed
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function receives data from a socket and, at the same time, obtains the address information of the sending host.

Data is received from socket *`s`* in the form of *`len`* bytes, and the received data is stored to the area specified with *`buf`*.

The address structure of the sending host is stored to the area specified with *`addr`* and *`paddrlen`*.

This function does not timeout implicitly.

For details on *`flags`*, Refer to the `sceNetRecv()` description.

Notes

Measures to reduce the number of warnings of cast to `SceNetSockaddr` are implemented in the actual header of this function.

sceNetRecvmsg

Receive data using message header structure

Definition

```
#include <net.h>
int sceNetRecvmsg(
        SceNetId s,
        SceNetMsghdr *msg,
        int flags
);
```

Arguments

s	ID of socket to receive data
msg	Pointer to message header structure for storing data
flags	Flags

The following values can be set to *flags*.

Value	Description
SCE_NET_MSG_DONTWAIT	Calls as non-blocking
SCE_NET_MSG_PEEK	Leaves receive data unchanged in receive buffer
SCE_NET_MSG_WAITALL	Blocks until specified buffer size is received

Return Values

Value	Description
Positive number	Size of received data
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetSocketAbort()
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	Socket is in blocking state (when non-blocking) Timeout occurred (when SCE_NET_SO_RCVTIMEO option is specified)
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EMSGSIZE	Size of *msg_iovlen* is too large
SCE_NET_ENOTCONN	Connection not established
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_ECANCELED	Close processing was called for a socket that is in the wait condition and being executed
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to receive data from a socket and store it to the message header structure.

Data is received from socket *s*, and the received data and address structure of the sender are stored to the message header structure specified with *msg*.

To obtain the address structure of the sending host, specify a pointer to the area for storing the address structure in *msg* -> *msg_name*, and specify the size in *msg* -> *msg_namelen*.

The receive data is stored sequentially from the beginning of the scatter/gather structure array. The maximum value of array elements that can be specified with *msg* -> *msg_iov* is 1024.

SCE_NET_MSG_PEEKLEN cannot be specified to *flags*. For details on *flags*, Refer to the sceNetRecv() description.

sceNetSend

Send data

Definition

```
#include <net.h>
int sceNetSend(
        SceNetId s,
        const void *buf,
        SceSize len,
        int flags
);
```

Arguments

s	ID of socket to send data
buf	Pointer to send data
len	Size of data to be sent (bytes)
flags	Flags

The following values can be set to *flags*.

Value	Description
SCE_NET_MSG_DONTWAIT	Calls as non-blocking
SCE_NET_MSG_USECRYPTO	Encrypts send data (Valid only when socket type is SCE_NET_SOCK_DGRAM_P2P)
SCE_NET_MSG_USESIGNATURE	Appends signature to send data (Valid only when socket type is SCE_NET_SOCK_DGRAM_P2P)

Return Values

Value	Description
0 or higher	Size of sent data
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetSocketAbort()
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	Socket is in blocking state (when non-blocking) Timeout occurred (when SCE_NET_SO_SNDTIMEO option is specified)
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EPIPE	Writing side of socket already closed
SCE_NET_EMSGSIZE	Message size is too large
SCE_NET_EDESTADDRREQ	Invalid send request (sceNetSendto() should be used)
SCE_NET_EHOSTDOWN	Other side is down and unreachable
SCE_NET_EHOSTUNREACH	Network unreachable
SCE_NET_ENETDOWN	Interface is down
SCE_NET_ENETUNREACH	Destination is unreachable
SCE_NET_ECONNRESET	Connection was reset (TCP only)
SCE_NET_ENOTCONN	Connection not established
SCE_NET_ERETURN	libnetctl error was returned

Value	Description
`SCE_NET_EADHOC`	UDP or TCP was attempted in the ad hoc communication mode
`SCE_NET_ECANCELED`	Close processing was called for a socket that is in the wait condition and being executed
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function sends data.

Data in the area specified with *buf* is sent to socket *s* in the form of *len* bytes.

Notes

When sending data to a blocking socket, the TCP socket will perform blocking until data of the specified send data size is sent.

The maximum size of data that can be sent by UDP/UDPP2P is 9216 bytes (default value).

The maximum size of data that can be sent by a RAW socket is always 8192 bytes.

sceNetSendto

Send data (specify receiving host)

Definition

```
#include <net.h>
int sceNetSendto(
        SceNetId s,
        const void *buf,
        SceSize len,
        int flags,
        const SceNetSockaddr *addr,
        SceNetSocklen_t addrlen
);
```

Arguments

s	ID of socket to send data
buf	Pointer to send data
len	Size of data to be sent (bytes)
flags	Flags
addr	Pointer to address structure of receiving host
addrlen	Size of address structure of receiving host

The following values can be set to *flags*.

Value	Description
SCE_NET_MSG_DONTWAIT	Calls as non-blocking
SCE_NET_MSG_USECRYPTO	Encrypts send data (Valid only when socket type is SCE_NET_SOCK_DGRAM_P2P)
SCE_NET_MSG_USESIGNATURE	Appends signature to send data (Valid only when socket type is SCE_NET_SOCK_DGRAM_P2P)

Return Values

Value	Description
0 or higher	Size of sent data
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetSocketAbort()
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EACCES	Attempted to send data to a broadcast address without specifying the SCE_NET_SO_BROADCAST socket option
SCE_NET_EISCONN	Specified connection is already established
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	Socket is in blocking state (when non-blocking) Timeout occurred (when SCE_NET_SO_SNDTIMEO option is specified)
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EPIPE	Writing side of socket already closed
SCE_NET_EMSGSIZE	Message size is too large
SCE_NET_EHOSTDOWN	Other side is down and unreachable

Value	Description
SCE_NET_EHOSTUNREACH	Network unreachable
SCE_NET_ENETDOWN	Interface is down
SCE_NET_ENETUNREACH	Destination is unreachable
SCE_NET_ENOTCONN	Connection not established
SCE_NET_ERETURN	libnetctl error was returned
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_ECANCELED	Close processing was called for a socket that is in the wait condition and being executed
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to send data by specifying the destination.

Data is sent from socket *s* in the form of *len* bytes. Specify the send data with *buf* and the address structure of the receiving host with *addr* and *addrlen*.

For RAW sockets, the IP header area is not included in the send data. By specifying SCE_NET_IP_HDRINCL with sceNetSetsockopt(), the IP header area can be edited as needed. Also refer to sceNetSend().

Notes

Measures to reduce the number of warnings of cast to SceNetSockaddr are implemented in the actual header of this function.

sceNetSendmsg

Send data using message header structure

Definition

```
#include <net.h>
int sceNetSendmsg(
        SceNetId s,
        const SceNetMsghdr *msg,
        int flags
);
```

Arguments

s	ID of socket to send data
msg	Pointer to message header structure for send data
flags	Flags

The following values can be set to `flags`.

Value	Description
SCE_NET_MSG_DONTWAIT	Calls as non-blocking
SCE_NET_MSG_USECRYPTO	Encrypts send data (Valid only when socket type is SOCK_DGRAM_P2P)
SCE_NET_MSG_USESIGNATURE	Appends signature to send data (Valid only when socket type is SOCK_DGRAM_P2P)

Return Values

Value	Description
0 or higher	Size of sent data
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by `sceNetSocketAbort()`
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	Socket is in blocking state (when non-blocking) Timeout occurred (when SCE_NET_SO_SNDTIMEO option is specified)
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EPIPE	Writing side of socket already closed
SCE_NET_EHOSTDOWN	Other side is down and unreachable
SCE_NET_EHOSTUNREACH	Network unreachable
SCE_NET_ENETDOWN	Interface is down
SCE_NET_ENETUNREACH	Destination is unreachable
SCE_NET_ECONNRESET	Connection was reset (TCP only)
SCE_NET_EMSGSIZE	Size of *msg_iovlen* is too large
SCE_NET_ENOTCONN	Connection not established
SCE_NET_ERETURN	libnetctl error was returned
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_ECANCELED	Close processing was called for a socket that is in the wait condition and being executed

Value	Description
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to send data contained in the message header structure.

Data of the message header structure specified with *msg* is sent to socket *s*. When specifying the destination, specify the address structure and size of the destination host with *msg* -> *msg_name* and *msg* -> *mag_namelen*.

The send data is sent sequentially from the beginning of the scatter/gather structure array.

The maximum value of array elements that can be specified with *msg* -> *msg_iov* is 1024. Also refer to `sceNetSend()`.

sceNetSetsockopt

Set socket options

Definition

```
#include <net.h>
int sceNetSetsockopt(
        SceNetId s,
        int level,
        int optname,
        const void *optval,
        SceNetSocklen_t optlen
);
```

Arguments

s	Socket ID for which socket option is to be set
level	Socket option level
optname	Socket option name
optval	Pointer to area for storing socket option value
optlen	Size of socket option value

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EINVAL	Invalid value specified
SCE_NET_ENOPROTOOPT	Invalid combination of specified level (*level*) and option (*optname*)
SCE_NET_EADDRNOTAVAIL	Invalid address specified
SCE_NET_ETOOMANYREFS	Too many multicast addresses specified
SCE_NET_ECONNRESET	Connection had already been reset for TCP related settings
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to set the socket options.

The value specified with *optval* and *optlen* is set to the socket option specified with *level* and *optname* of the socket specified with *s*.

Refer to the "List of Socket Options" section for the supported socket options.

sceNetShutdown

Shut down socket

Definition

```
#include <net.h>
int sceNetShutdown(
        SceNetId s,
        int how
);
```

Arguments

s	ID of socket to be shut down
how	Shutdown method

The following values can be specified with *how*.

Value	Description
SCE_NET_SHUT_RD	Shuts down reading
SCE_NET_SHUT_WR	Shuts down writing
SCE_NET_SHUT_RDWR	Shuts down reading and writing

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EPIPE	Writing side of socket already closed
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function shuts down a socket.

It prohibits communication over socket *s* in part or in full. When `SCE_NET_SHUT_RD` is specified for *how*, subsequent receiving is prohibited. When `SCE_NET_SHUT_WR` is specified for *how*, subsequent sending is prohibited. When `SCE_NET_SHUT_RDWR` is specified for *how*, subsequent sending and receiving is prohibited.

The socket resources are not freed, so `sceNetSocketClose()` must be called.

sceNetSocket

Create socket

Definition

```
#include <net.h>
SceNetId sceNetSocket(
        const char *name,
        int family,
        int type,
        int protocol
);
```

Arguments

name	Debugging Name
family	Address family of socket to be created (SCE_NET_AF_INET)
type	Socket type
protocol	Protocol (valid for RAW socket)

The following values can be set to *type*.

Value	Description
SCE_NET_SOCK_STREAM	TCP socket
SCE_NET_SOCK_DGRAM	UDP socket
SCE_NET_SOCK_DGRAM_P2P	UDPP2P socket
SCE_NET_SOCK_RAW	RAW socket
SCE_NET_SOCK_STREAM_P2P	TCP over UDPP2P socket

Return Values

The ID of the created socket (0 or higher) is returned.

If an error occurs, a negative value is returned.

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EMFILE	Insufficient space in socket ID table
SCE_NET_EPROTONOSUPPORT	Invalid socket type or protocol family
SCE_NET_EADHOC	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_EPROTOTYPE	Unsupported protocol type was specified

Description

This function creates a socket.

It creates a socket with the address family indicated by *family* and the socket type indicated by *type*, and returns the descriptor for that socket.

sceNetSocketClose

Close socket

Definition

```
#include <net.h>
int sceNetSocketClose(
        SceNetId s
);
```

Arguments

s ID of socket to be closed

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Value	Description
`SCE_NET_EBADF`	Invalid socket ID specified

Description

This function ends the use of a socket and frees its resources.

If communication of the socket specified with *s* has not been closed with `sceNetShutdown()`, communication is closed, and then the termination process is performed.

The socket ID becomes invalid when this function is called. Thereafter, do not perform processes with this socket ID.

`sceNetSocketClose()` does not perform blocking unless the linger option is specified.

sceNetSocketAbort

Abort socket processing

Definition

```
#include <net.h>
int sceNetSocketAbort(
        SceNetId s,
        int flags
);
```

Arguments

s	ID of socket to be aborted
flags	Flags

The following values can be set to *flags*.

Value	Description
SCE_NET_SOCKET_ABORT_FLAG_RCV_PRESERVATION	Saves abort processing to receive functions (sceNetRecv(), sceNetRecvfrom(), sceNetRecvmsg(), sceNetAccept())
SCE_NET_SOCKET_ABORT_FLAG_SND_PRESERVATION	Saves abort processing to send functions (sceNetSend(), sceNetSendto(), sceNetSendmsg(), sceNetConnect())

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid socket ID specified
SCE_NET_ENOTBLK	Function called for socket not in wait state
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function aborts processing for a socket.

It unblocks sockets blocked by functions such as sceNetSend() and sceNetRecv() specified with *s*. Namely, the applicable functions return an error as sce_net_errno = SCE_NET_EINTR.

If *flags* is not specifically specified, calling this function for a socket that is not in wait state returns an error and the abort process is not performed. To change this action and have the abort process performed when the next receive or send function is called, set the conditions in *flags*. When the target receive or send function is performed at this time and it is determined that the abort process is saved, an error is returned immediately as sce_net_errno = SCE_NET_EINTR even if sending or receiving can be performed. Moreover, each condition being independent, no influence can be exerted on one of the *flags* conditions.

This socket only performs the abort process. Following the release of a block, send/receive and other processing can be resumed for that socket. Moreover, to close a socket, `sceNetSocketClose()` must be called separately.

Notes

Regarding TCP and TCP over UDP socket transmission during block operation, when there is no available space in the socket send buffer, the socket processing is blocked until transmission of all the data has been completed. If abort process is performed at this time, in the case where there is any data that was successfully sent up to that time, the target send function returns the size of that data and on other hands, `SCE_NET_ERROR_EINTR` is returned.

The UDP, UDPP2P and RAW sockets are not blocked, but if the send function is called in the state where the abort process has been saved, `SCE_NET_ERROR_EINTR` is returned in the same way as for the TCP and TCP over UDP sockets.

Network Communication Functions (Multiplex I/O)

sceNetEpollCreate

Create multiplex I/O

Definition

```
#include <net.h>
SceNetId sceNetEpollCreate(
        const char *name,
        int flags
);
```

Arguments

name	Debugging Name
flags	Flag (always 0)

Return Values

The ID of the created multiplex I/O (epoll ID, 0 or higher) is returned.

If an error occurs, a negative value is returned.

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EMFILE	Insufficient space in epoll ID table

Description

This function creates an ID for multiplex I/O.

sceNetEpollDestroy

Destroy multiplex I/O

Definition

```
#include <net.h>
int sceNetEpollDestroy(
        SceNetId eid
);
```

Arguments

eid	epoll ID

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EBADF`	Invalid epoll ID specified

Description

This function destroys the target epoll ID.

Operations for this epoll ID can no longer be performed after this function is called.

sceNetEpollControl

Required operation for event waiting of multiplex I/O

Definition

```
#include <net.h>
int sceNetEpollControl(
        SceNetId eid,
        int op,
        SceNetId id,
        SceNetEpollEvent *event
);
```

Arguments

eid	epoll ID
op	Operation type
id	libnet ID associated with epoll ID
event	Pointer to area for storing associated event

The following values can be set to *op*. (OR cannot be specified.)

Value	Description
SCE_NET_EPOLL_CTL_ADD	Associates to *eid* of *id*
SCE_NET_EPOLL_CTL_MOD	Resets associated event
SCE_NET_EPOLL_CTL_DEL	Deletes association from *eid* of *id* (*event* is always NULL)

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid epoll ID or libnet ID specified
SCE_NET_EBUSY	(Does not occur after SDK 0.990)
SCE_NET_EEXIST	SCE_NET_EPOLL_CTL_ADD specified to previously associated libnet ID
SCE_NET_EINACTIVEDISABLED	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function sets, resets and deletes a libnet ID that is waiting for an event to a target epoll ID.

The supported event delivering method is only the delivered level trigger operation in either the readable or the writable states.

The libnet IDs that can wait for epoll are socket IDs and DNS resolver IDs. A separate epoll ID or dump ID cannot be specified. The area of *event* does not need to be held after calling the function. Refer to the description of the SceNetEpollEvent structure for associated events.

Events can be set, re-set, and canceled from a different thread for *`eid`* waiting for an event with `sceNetEpollWait()` or `sceNetEpollWaitCB()`. However, `sceNetEpollAbort()` must be used for block release of `sceNetEpollWait()` or `sceNetEpollWaitCB()`.

sceNetEpollWait, sceNetEpollWaitCB

Event waiting of multiplex I/O

Definition

```
#include <net.h>
int sceNetEpollWait(
        SceNetId eid,
        SceNetEpollEvent *events,
        int maxevents,
        int timeout_us
);
int sceNetEpollWaitCB(
        SceNetId eid,
        SceNetEpollEvent *events,
        int maxevents,
        int timeout_us
);
```

Arguments

eid	epoll ID
events	Pointer to area for storing usable event
maxevents	Number of events of area for storing events (1 or higher)
timeout_us	Timeout (-1 (negative value) means infinite timeout, microseconds)

Return Values

Value	Description
0	Timeout occurred while there were no events
Positive number	Number of libnet IDs of which usable events have occurred
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINTR`	Blocking canceled
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EBADF`	Invalid epoll ID specified

Description

This function waits for multiplex I/O and checks libnet IDs associated by `sceNetEpollControl()` for the state that can be used for input/output. Using this function enables the simultaneous processing of multiple libnet IDs that can wait for epoll on a single thread.

The sufficient maximum number of *maxevents* is the number of libnet IDs associated with epoll IDs (*eid*) by `sceNetEpollControl()`. When the input events of `SCE_NET_EPOLLIN` and `SCE_NET_EPOLLOUT` are set to a socket ID, and then these events are output simultaneously, the return value is 1.

Note that the timeout time unit is the microsecond.

Notes

For the events that occur, refer to SceNetEpollEvent.

If the abort process is performed for libnet IDs that can wait for epoll associated with an epoll ID (sceNetSocketAbort(), for example), an event occurs for the target epoll ID, and SCE_NET_EPOLLHUP is notified to the libnet IDs.

sceNetEpollWaitCB() is a function that can wait for CB. Refer to the kernel feature for use of CB waiting.

sceNetEpollAbort

Destroy multiplex I/O

Definition

```
#include <net.h>
int sceNetEpollAbort(
        SceNetId eid,
        int flags
);
```

Arguments

eid	epoll ID of target
flags	Flag

The following values can be set to *flags*.

Value	Description
SCE_NET_EPOLL_ABORT_FLAG_PRESERVATION	Saves abort process

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid epoll ID specified

Description

This function aborts multiplex I/O that is being executed with the target epoll ID. Given this feature, the multiplex I/O target to singularly cancel a wait for can be aborted even if the libnet ID that can wait for epoll and associated with epoll waiting changes.

If *flags* is not specifically specified, calling this function for an epoll ID that is not in wait state returns an error and the abort process is not performed. To change this action and have the abort process performed when the next epoll event waiting is called, set the conditions in *flags*. When the target epoll event waiting is performed at this time and it is determined that the abort process is saved, an error is returned immediately as sce_net_errno = SCE_NET_EINTR even in the state where the event exists.

This function only performs the abort process. Following block release, the processing for that epoll ID can be resumed. Moreover, to terminate epoll, sceNetEpollDestroy() must be called separately.

Network Communication Functions (Extension)

sceNetGetSockInfo

Get socket information

Definition

```
#include <net.h>
int sceNetGetSockInfo(
        SceNetId s;
        SceNetSockInfo *p,
        int n,
        int flags
);
```

Arguments

s	Socket ID
p	Pointer to buffer for storing obtained socket information
n	Maximum number of entries to obtain
flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EBADF`	Invalid socket ID specified
`SCE_NET_EINACTIVEDISABLED`	Network disconnection occurred owing to intermittent disconnection or system suspend.

Description

This function is used to obtain information about a socket.

When a socket ID is specified to *s*, information about that socket is stored to the buffer indicated with *p*. The required buffer size is `sizeof(SceNetSockInfo)` bytes.

When -1 (negative value) is specified to *s*, information about all sockets is stored to the buffer indicated with *p*. *n* is the number of information items about a socket, so specify the value dividing the size of the provided buffer with `sizeof(SceNetSockInfo)`. Specifying NULL to *p* returns the number of current sockets, so it is possible to use this to find out the appropriate value of *n* and the buffer size.

For information of sockets already closed, -1 is returned for socket ID *s* of the `SceNetSockInfo` structure.

See Also

`SceNetSockInfo`, `sceNetGetpeername()`, `sceNetGetsockname()`

sceNetGetSockIdInfo

Get socket ID bit string

Definition

```
#include <net.h>
int sceNetGetSockIdInfo(
        SceNetFdSet *fds,
        int sockinfo_flags,
        int flags
);
```

Arguments

fds	Pointer to socket bit set
sockinfo_flags	Condition flags for which to search
flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Normal termination (number of times target socket ID bit is set to 1)
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function is used to obtain the bit string of a socket ID.

The same flags of the *flags* member of the `SceNetSockInfo` structure can be specified to *sockinfo_flags*. For example, when specifying `SCE_NET_SOCKINFO_F_RECV_WAIT`, the target bit is set to 1 for the receive wait socket ID.

This function can be used to obtain information for a socket that does not require dynamic memory allocation of an application. Namely, this function can be used to find out information of all required socket IDs by calling `sceNetGetSockInfo()` for each socket ID for which the target bit is set to 1.

sceNetGetStatisticsInfo

Get statistics information

Definition

```
#include <net.h>
int sceNetGetStatisticsInfo(
        SceNetStatisticsInfo *info,
        int flags
)
```

Arguments

info	Pointer to area for storing statistics information
flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function obtains statistics information.

sceNetDumpCreate

Start log acquisition

Definition

```
#include <net.h>
SceNetId sceNetDumpCreate(
        const char *name,
        int len,
        int flags
);
```

Arguments

name	Debugging Name
len	Maximum log buffer length (2048 or more)
flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Log ID
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EMFILE`	Insufficient space in log ID table

Description

This function starts log (tcpdump format log) acquisition.

sceNetDumpDestroy

End log acquisition

Definition

```
#include <net.h>
int sceNetDumpDestroy(
        SceNetId id
);
```

Arguments

id Log ID

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid log ID specified

Description

This function ends use of the target log ID.

Operations for this log ID can no longer be performed after this function is called.

sceNetDumpRead

Get log

Definition

```
#include <net.h>
int sceNetDumpRead(
        SceNetId id,
        void *buf,
        int len,
        int *pflags
);
```

Arguments

id	Log ID
buf	Area for saving log
len	Maximum length of area for saving log
pflags	Flags

The following values can be set to *flags*.

Value	Description
SCE_NET_DUMP_DONTWAIT	Calls as non-blocking (input side)
SCE_NET_DUMP_PEEK	Leaves receive data unchanged in receive buffer (input side)
SCE_NET_DUMP_OVERFLOW	Blocks until specified buffer size is received (output side)

Return Values

Value	Description
0 or higher	Normal termination (log size)
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid log ID specified

Description

This function obtains the started log.

When SCE_NET_DUMP_PEEK is specified for an input level flag, the log remains in the log buffer.

When SCE_NET_DUMP_DONTWAIT is specified but there is no data in the log buffer, the function returns from the call without blocking. When not using this function, perform initialization with 0.

When the SCE_NET_DUMP_OVERFLOW bit for the output level flag is 1, this indicates that the log data did not fit in the buffer specified at the time the log ID was created.

sceNetDumpAbort

Stop log acquisition

Definition

```
#include <net.h>
SceNetId sceNetDumpAbort(
        SceNetId rid,
        int flags
)
```

Arguments

rid	Target log ID
flags	Flag

The following values can be set to *flags*.

Value	Description
SCE_NET_DUMP_ABORT_FLAG_PRESERVATION	Saves abort process

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid log ID specified

Description

This function stops log acquisition that is being performed with the target log ID.

To end log acquisition, sceNetDumpDestroy() must be called separately.

If *flags* is not specifically specified, calling this function for a log ID that is not in wait state returns an error and the abort process is not performed. To change this action and have the abort process performed when the next log obtaining function is called, set the conditions in *flags*. The target function is executed at this time and if it is determined that the abort process was saved, sce_net_errno = SCE_NET_EINTR is returned as an error.

This function performs only the abort process, so the processing for that log ID can be resumed following block release. To terminate log acquisition, sceNetDumpDestroy() must be called.

sceNetSetDnsInfo

Set DNS address

Definition

```
#include <net.h>
int sceNetSetDnsInfo(
        SceNetDnsInfo *info,
        int flags
)
```

Arguments

info	Pointer to area for storing DNS address
flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function sets the DNS address required by the application.

A valid address must be set to *info*->*dns_addr*[0]. To clear the DNS address setting, specify NULL for *info*.

The use of this function is not usually required.

See Also

`SceNetDnsInfo`

sceNetClearDnsCache

Clear DNS cache

Definition

```
#include <net.h>
int sceNetClearDnsCache(
        int flags
)
```

Arguments

flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function clears the DNS cache.

The use of this function is not usually required.

DNS Resolver Functions

sceNetResolverCreate

Create DNS resolver ID

Definition

```
#include <net.h>
SceNetId sceNetResolverCreate(
        const char *name,
        SceNetResolverParam *param,
        int flags
)
```

Arguments

name	Debugging Name
param	Pointer to DNS resolver parameters
flags	Flag (always 0)

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content

Description

This function obtains the DNS resolver ID, which is the libnet ID. The same DNS resolver ID can be repeated to resolve the name.

When NULL is specified for the DNS resolver parameters, the function operates as though the parameters are not set.

Saving the DNS resolver parameters can be safely omitted, with the exception of the user data pointer destination, following execution of this function.

Refer to the `SceNetResolverParam` structure for the meanings of the parameters.

(The memory required for DNS resolver is as described in the description of the `SceNetInitParam` structure, so it is recommended to use the memory of libnet after estimating the maximum usage amount, instead of using dynamic memory allocation via the DNS resolver parameters.)

sceNetResolverDestroy

Destroy DNS resolver ID

Definition

```
#include <net.h>
SceNetId sceNetResolverDestroy(
        SceNetId rid
)
```

Arguments

rid Target DNS resolver ID

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EBADF`	Invalid DNS resolver ID specified
`SCE_NET_RESOLVER_EBUSY`	Specified DNS resolver ID already in use (target ID not destroyed)

Description

This function ends use of the target DNS resolver ID.

This DNS resolver ID can no longer be operated after this function is called and ends normally.

`SCE_NET_RESOLVER_EBUSY` indicates that operations of the target DNS resolver ID, such as calling `sceNetResolverAbort()`, must be stopped.

sceNetResolverStartNtoa

Perform forward lookup name resolution

Definition

```
#include <net.h>
SceNetId sceNetResolverStartNtoa(
        SceNetId rid,
        const char *hostname,
        SceNetInAddr *addr,
        int timeout_us,
        int retry,
        int flags
)
```

Arguments

rid	DNS resolver ID
hostname	Host name of name resolution target
addr	Pointer to area for storing IP address (network byte order) corresponding to host name
timeout_us	Inquiry resend interval (microseconds)
retry	Inquiry resend count
flags	Flags

Specify the following values to *flags*.

Value	Description
0	Default operation
SCE_NET_RESOLVER_ASYNC	Non-blocking operation
SCE_NET_RESOLVER_START_NTOA_DISABLE_IPADDRESS	Operation that returns an error when IP address is specified for host name

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetResolverAbort()
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid DNS resolver ID specified
SCE_NET_EMFILE	Insufficient space in socket ID table
SCE_NET_EHOSTDOWN	Did not reach other side
SCE_NET_EHOSTUNREACH	Network unreachable
SCE_NET_ERETURN	libnetctl error was returned
SCE_NET_RESOLVER_EBUSY	Specified DNS resolver ID already in use
SCE_NET_RESOLVER_ENOSPACE	Insufficient memory (library)

Refer to Notes in sceNetResolverGetError() for other errors.

Description

This function performs forward lookup name resolution for the target DNS resolver ID.

This function inquires as to the IP address corresponding to the host name specified with *hostname*, and stores that information in the area specified with *addr*. For default operation, IP address can be specified for the host name. If name resolution ends normally with non-blocking operation, the result is stored in the area specified by *addr* as same as with blocking operation.

Supplement

When 0 is specified to *timeout_us*, this is handled as though specified at the default value of 1 second. When 0 is specified to *retry*, this is handled as though specified at the default value of 5 times. When the value specified for *timeout_us* is less than 2 seconds, *timeout_us* is set to the value of 2 seconds, and correction is made by only subtracting 1 so that 0 is not specified to *retry*. Then, the actual timeout time is calculated as follows based on *timeout (=timeout_us* (seconds)*)* and *retry*.

Timeout time when primary DNS only is set

Retry	Timeout time
First	*timeout* seconds
Second	*timeout* x 2 seconds
Third	*timeout* x 4 seconds
:	:
retry time	*timeout* x (2 ^ (*retry* -1)) seconds

When the default values are specified to both *timeout_us* and *retry*, the maximum wait time for a function call is calculated as follows:
2+4+8+16 = 30 seconds

Timeout time when both primary and secondary DNS are set

Retry	Timeout Time
First (primary DNS)	*timeout* seconds
First (secondary DNS)	*timeout* seconds
Second (primary DNS)	*timeout* seconds
Second (secondary DNS)	*timeout* seconds
Third (primary DNS)	*timeout* x 2 seconds
Third (secondary DNS)	*timeout* x 2 seconds
:	:
retry time (primary DNS)	*timeout* x (2 ^ (*retry* -2)) seconds
retry time (secondary DNS)	*timeout* x (2 ^ (*retry* -2)) seconds

When the default values are specified to both *timeout_us* and *retry*, the maximum wait time for a function call is calculated as follows:
(2+2)+(2+2)+(4+4)+(8+8) = 32 seconds

If *flags* is not specifically specified, this function is blocked until name resolution ends normally or an error is returned.

When `SCE_NET_RESOLVER_ASYNC` is specified to *flags* for non-blocking operation, non-blocking operation is performed, and if this function ends normally, this indicates that name resolution has started. The calling of `sceNetEpollWait()` or `sceNetEpollWaitCB()` is required for name resolution to proceed, and this is performed using the called thread context. In other words, just the normal termination of this function does not mean the name resolution is completed.

An arbitrary value can be specified for the timeout value. Also, if name resolution execution is not carried out in one go, the timeout time is extended proportionally to the time during which name resolution is not executed.

Also the ending of name resolution is determined by calling either `sceNetEpollWait()` or `sceNetEpollWaitCB()`. Thereafter, use `sceNetResolverGetError()` to determine whether name resolution ended normally or an error occurred.

sceNetResolverStartAton

Perform reverse lookup name resolution

Definition

```
#include <net.h>
SceNetId sceNetResolverStartAton(
        SceNetId rid,
        const SceNetInAddr *addr,
        char *hostname,
        int hostname_len,
        int timeout_us,
        int retry,
        int flags
)
```

Arguments

rid	DNS resolver ID
addr	Pointer to area for storing IP address (network byte order) for inquiry
hostname	Pointer to area for storing host name
hostname_len	Size of area for storing host name (size including NULL end characters)
timeout_us	Inquiry resend interval (microseconds)
retry	Inquiry resend count
flags	Flags

Specify the following values to *flags*.

Value	Description
0	Default operation
SCE_NET_RESOLVER_ASYNC	Non-blocking operation

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with sce_net_errno.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINTR	Aborted by sceNetResolverAbort()
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid DNS resolver ID specified
SCE_NET_EMFILE	Insufficient space in socket ID table
SCE_NET_EHOSTDOWN	Did not reach other side
SCE_NET_EHOSTUNREACH	Network unreachable
SCE_NET_ERETURN	libnetctl error was returned
SCE_NET_RESOLVER_EBUSY	Specified DNS resolver ID already in use
SCE_NET_RESOLVER_ENOSPACE	Insufficient memory (library) Insufficient size of area for storing host name

Refer to Notes in sceNetResolverGetError() for other errors.

Description

This function performs reverse lookup name resolution for the target DNS resolver ID.

This function inquires as to the host name corresponding to the IP address specified with *`addr`*, and stores the host name in the area specified with *`hostname`*. If *`hostname_len`* is insufficient for the reply host name, an error is returned. Therefore, it is recommended that (`SCE_NET_RESOLVER_HOSTNAME_LEN_MAX + 1`) be specified for the area size of *`hostname`*. If name resolution ends normally with non-blocking operation, the result is stored in the area specified by *`hostname`* as same as with blocking operation.

Supplement

The same supplement to `sceNetResolverStartNtoa()` applies to other features.

sceNetResolverGetError

Get name resolution execution result

Definition

```
#include <net.h>
SceNetId sceNetResolverGetError(
        SceNetId rid,
        int *result
)
```

Arguments

rid	DNS resolver ID
result	Pointer to area for storing execution result

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
`SCE_NET_EINVAL`	Function called due to invalid argument or content
`SCE_NET_EBADF`	Invalid DNS resolver ID specified

Description

This function obtains the result of the name resolution previously performed for the target DNS resolver ID. This is used to check errors of name resolution for which normal non-blocking operation was performed.

The following are the meanings of the `*result` value. The error value is the Sce error code as same as to the return value.

Value	Description
0	Name resolution ends normally
Negative number	Name resolution ends with an error.

Notes

The following results are stored in `*result`.

Value	Description
`SCE_NET_ERROR_EINTR`	Aborted by `sceNetResolverAbort()`
`SCE_NET_ERROR_RESOLVER_ENOSPACE`	Insufficient memory (library)
`SCE_NET_ERROR_RESOLVER_EPACKET`	Invalid DNS response
`SCE_NET_ERROR_RESOLVER_ENODNS`	DNS server not specified
`SCE_NET_ERROR_RESOLVER_ETIMEDOUT`	Timeout occurred
`SCE_NET_ERROR_RESOLVER_ENOSUPPORT`	Unsupported feature requested by server
`SCE_NET_ERROR_RESOLVER_EFORMAT`	Invalid response from server
`SCE_NET_ERROR_RESOLVER_ESERVERFAILURE`	Temporary error from server
`SCE_NET_ERROR_RESOLVER_ENOHOST`	Inquired host name does not exist
`SCE_NET_ERROR_RESOLVER_ENOTIMPLEMENTED`	Inquired feature is not implemented
`SCE_NET_ERROR_RESOLVER_ESERVERREFUSED`	Inquiry denied
`SCE_NET_ERROR_RESOLVER_ENORECORD`	Inquired record does not exist

sceNetResolverAbort

Stop name resolution

Definition

```
#include <net.h>
SceNetId sceNetResolverAbort(
        SceNetId rid,
        int flags
)
```

Arguments

rid	Target DNS resolver ID
flags	Flag

The following values can be set to *flags*.

Value	Description
SCE_NET_RESOLVER_ABORT_FLAG_NTOA_PRESERVATION	Saves abort process of lookup name resolution execution
SCE_NET_RESOLVER_ABORT_FLAG_ATON_PRESERVATION	Saves abort process of reverse lookup name resolution execution

Return Values

Value	Description
0 or higher	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_EINVAL	Function called due to invalid argument or content
SCE_NET_EBADF	Invalid DNS resolver ID specified

Description

This function stops name resolution that is being performed with the target DNS resolver ID. This function also stops name resolution being performed with non-blocking.

To destroy the DNS resolver ID, `sceNetResolverDestroy()` must be called separately.

If *flags* is not specifically specified, calling this function for a DNS resolver ID that is not in wait state returns an error and the abort process is not performed. To change this action and have the abort process performed when the next name resolution function is called, set the conditions in *flags*. The target name resolution is executed at this time, and if it is determined that the abort process was saved, `sce_net_errno` = `SCE_NET_EINTR` is returned as an error even when send/receive is enabled. Moreover, each condition being independent, no influence can be exerted on one of the *flags* conditions.

Since this function only performs the abort process, the processing can be resumed that DNS resolver following block release. To terminate a DNS resolver ID, `sceNetResolverDestroy()` must be called.

Functions Exclusively for Developing Programs

sceNetShowIfconfig

Display interface state

Definition

```
#include <net.h>
int sceNetShowIfconfig(void);
```

Arguments

None

Return Values

Value	Description
0	Normal termination

Description

This function displays the interface states and name server information. Refer to the "libnet Overview" document for the display information.

As TTY output control during debugging, `sceNetGetStatisticsInfo()` can be used when one needs to know just the network memory state.

sceNetShowNetstat

Display socket information

Definition

```
#include <net.h>
int sceNetShowNetstat(void);
```

Arguments

None

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Value	Description
`SCE_NET_ENOLIBMEM`	Insufficient memory (library)

Description

This function displays the socket and resolver ID states. Refer to the "libnet Overview" document for the display information.

As TTY output control during debugging, `sceNetGetSockInfo()` can be used when one needs to know the state of a specific socket or resolver ID.

sceNetShowRoute

Display routing information

Definition

```
#include <net.h>
int sceNetShowRoute(void);
```

Arguments

None

Return Values

Value	Description
0	Normal termination

Description

This function displays routing information.

sceNetEmulationSet

Set network emulation parameters

Definition

```
#include <net.h>
int sceNetEmulationSet(
        SceNetEmulationParam *param,
        int flags
);
```

Arguments

param	Parameters to be set
flags	Flag

The following values can be set to *flags*.

Value	Description
SCE_NET_EMULATION_FLAG_ETH0	USB Ethernet interface
SCE_NET_EMULATION_FLAG_WLAN0	Wireless interface

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_ENODEV	Target device does not exist
SCE_NET_EINVAL	Invalid argument or flag has been specified
SCE_NET_ENOSPC	Attempted to set an invalid parameter value
SCE_NET_ENOENT	*param*->*version* value is invalid
SCE_NET_ENOTSUP	This call is invalid (SDK 2.000 or later)

Description

This function sets the network emulation parameters. Refer to the "libnet Overview" document for the meanings of the parameters.

One physical interface to which the network emulation parameters are set must be specified for *flags*. The parameters are applied to this function only during the "development mode". Note that after your title is released, the emulation operation is not executed even if the values of parameters are valid, while the function is terminated normally.

this indicates that the emulation operation was not executed owing to the condition described in the "Network Emulation" chapter of the "libnet Overview" document.

When the return value is `SCE_NET_ERROR_ENOSPC` error, the parameter whose setting value is not correct can be identified by calling `sceNetEmulationGet()` and then confirming the position of the member of *param* indicated by the *param*->*result* value. For instance, if the *param*->*result* value is 24, it means *param*->*send.delay_jitter* is wrong.

Notes

This function can be used only for development purposes. This function cannot be used for master packages.

sceNetEmulationGet

Get network emulation parameters

Definition

```
#include <net.h>
int sceNetEmulationGet(
        SceNetEmulationParam *param,
        int flags
);
```

Arguments

param	Parameter storage destination
flags	Flag

The following values can be set to *flags*.

Value	Description
SCE_NET_EMULATION_FLAG_ETH0	USB Ethernet interface
SCE_NET_EMULATION_FLAG_WLAN0	Wireless interface

Return Values

Value	Description
0	Normal termination
Negative number	Error

Details of the error can be obtained with `sce_net_errno`.

Other than the error codes below, libnet common error codes [common] described in the "Error Codes" section may return.

Value	Description
SCE_NET_ENODEV	Target device does not exist
SCE_NET_EINVAL	Invalid argument or flag has been specified

Description

This function obtains the currently set network emulation parameters. Refer to the "libnet Overview" document for the meanings of the parameters.

One physical interface from which the network emulation parameters are obtained must be specified for *flags*.

Notes

This function can be used only for development purposes. This function cannot be used for master packages.

Socket ID Set Operations

SCE_NET_FD_CLR

Clear socket ID

Definition

```
#include <net.h>
SCE_NET_FD_CLR(n, p)
```

Arguments

n	Socket ID
p	Pointer to `SceNetFdSet` structure

Description

This macro removes certain socket IDs from a socket ID set.

It sets the bit corresponding to the socket ID specified with *n* of the `SceNetFdSet` structure specified with *p* to 0.

SCE_NET_FD_ISSET

Examine socket ID

Definition

```
#include <net.h>
SCE_NET_FD_ISSET(n, p)
```

Arguments

n	Socket ID
p	Pointer to `SceNetFdSet` structure

Return Values

Value	Description
Not 0	Socket ID *n* is set (bit is 1)
0	Socket ID *n* is not set (bit is 0)

Description

This macro checks whether or not a certain socket ID is set.

It returns whether or not the bit corresponding to the socket ID specified with *n* of the `SceNetFdSet` structure specified with *p* is 1.

SCE_NET_FD_SET

Set socket ID

Definition

```
#include <net.h>
SCE_NET_FD_SET(n, p)
```

Arguments

n	Socket ID
p	Pointer to `SceNetFdSet` structure

Description

This macro adds certain socket IDs to a socket ID set.

It sets the bit corresponding to the socket ID specified with *n* of the `SceNetFdSet` structure specified with *p* to 1.

SCE_NET_FD_ZERO

Initialize socket ID set with 0

Definition

```
#include <net.h>
SCE_NET_FD_ZERO(p)
```

Arguments

p Pointer to `SceNetFdSet` structure

Description

This macro initializes the socket ID set.

It sets all of the bits of the `SceNetFdSet` structure specified with *p* to 0.

Socket Options

List of Socket Options

List of socket options

Socket Level	Option Name	get	set
SCE_NET_SOL_SOCKET	SCE_NET_SO_BROADCAST	get	set
	SCE_NET_SO_ERROR	get	-
	SCE_NET_SO_ERROR_EX	get	-
	SCE_NET_SO_KEEPALIVE	get	set
	SCE_NET_SO_LINGER	get	set
	SCE_NET_SO_RCVBUF	get	set
	SCE_NET_SO_SNDBUF	get	set
	SCE_NET_SO_RCVTIMEO	get	set
	SCE_NET_SO_SNDTIMEO	get	set
	SCE_NET_SO_REUSEADDR	get	set
	SCE_NET_SO_REUSEPORT	get	set
	SCE_NET_SO_TYPE	get	-
	SCE_NET_SO_NBIO	get	set
	SCE_NET_SO_ONESBCAST	get	set
	SCE_NET_SO_USECRYPTO	get	set
	SCE_NET_SO_USESIGNATURE	get	set
	SCE_NET_SO_TPPOLICY	get	set
	SCE_NET_SO_NAME	-	set
SCE_NET_IPPROTO_IP	SCE_NET_IP_MULTICAST_IF	get	set
	SCE_NET_IP_MULTICAST_TTL	get	set
	SCE_NET_IP_MULTICAST_LOOP	get	set
	SCE_NET_IP_ADD_MEMBERSHIP	-	set
	SCE_NET_IP_DROP_MEMBERSHIP	-	set
	SCE_NET_IP_HDRINCL	get	set
	SCE_NET_IP_TTL	get	set
	SCE_NET_IP_TTLCHK	get	set
	SCE_NET_IP_MAXTTL	get	-
	SCE_NET_IP_DONTFRAG	get	set
	SCE_NET_IP_TOS	get	set
SCE_NET_IPPROTO_TCP	SCE_NET_TCP_NODELAY	get	set
	SCE_NET_TCP_MAXSEG	get	set
	SCE_NET_TCP_MSS_TO_ADVERTISE	get	set

SCE_NET_SO_BROADCAST

Allow sending of broadcast datagrams

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (allow)

Description

This option allows a socket to send broadcast datagrams. It is valid only when the socket type is `SCE_NET_SOCK_DGRAM` or `SCE_NET_SOCK_DGRAM_P2P`. Other socket types are not affected even if they are set.

SCE_NET_SO_ERROR

Get pending error

Definition

```
int
```

Value

Pending error value

Description

This option obtains a pending error value for the socket. (an error code obtained with `sce_net_errno`). In the case that no error occurs, 0 is obtained. Once an error value is obtained, the error is cleared to 0 for that socket.

An error from libnetctl is only expressed as `SCE_NET_ERETURN`. To obtain the details on the error, use `SCE_NET_SO_ERROR_EX` instead of `SCE_NET_SO_ERROR`.

Typical usage of this option is as follows:

- The error for the socket after the event of `SCE_NET_EPOLLERR` can be referenced.
- The result of the `sceNetConnect()` process after `sceNetConnect()` is returned with `sce_net_errno` = `SCE_NET_EINPROGRESS` can be referenced.

SCE_NET_SO_ERROR_EX

Get pending error (Extended)

Definition

```
int
```

Value

Pending error value

Description

This option obtains a pending Sce error code value for the socket. (refer to the "libnet Overview" document.)

In the case that no error occurs, 0 is obtained. Once an error value is obtained, the error is cleared to 0 for that socket. Also, the error obtained with `SCE_NET_SO_ERROR` is cleared to 0 at the same time.

The error codes obtained with this option include the error codes from libnetctl.

SCE_NET_SO_KEEPALIVE

Send TCP keep-alive probe

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (send keep-alive probe)

Description

This option relates to sending a keep-alive probe. It is valid only when the socket type is `SCE_NET_SOCK_STREAM` or `SCE_NET_SOCK_STREAM_P2P`. If this option is enabled and no data is sent or received over the socket within two hours, TCP automatically sends a keep-alive probe to the destination.

SCE_NET_SO_LINGER

Control TCP connection termination process

Definition

```
SceNetLinger
```

Value

`l_onoff`

0	Disable [default]
Non-zero	Enable (reference *`l_linger`*)

`l_linger`

0	Reset
>0	Linger time specification (seconds)

Description

This option controls the termination process of a TCP connection. It is valid only when the socket type is `SCE_NET_SOCK_STREAM` or `SCE_NET_SOCK_STREAM_P2P`.

If *`l_onoff`* is enabled and *`l_linger`* is 0, TCP immediately discards the data existing in the send buffer for a closed connection and sends RST.

If *`l_onoff`* is enabled and *`l_linger`* is a positive number, TCP performs linger operations for a closed connection. In other words, all of the data in the send buffer is sent and blocking is performed until an ACK is received from the destination or the linger time has elapsed. If blocking is required on a non-blocking socket, an error occurs and `sce_net_errno` = `SCE_NET_EWOULDBLOCK` is returned.

The linger time cannot control the time of the TCP TIME_WAIT state.

SCE_NET_SO_RCVBUF

Receive buffer size

Definition

```
int
```

Value

Receive buffer size (bytes)

Description

This option relates to the receive buffer size of a socket.

The receive buffer size of TCP or TCP over UDPP2P can be specified up to 512 KiB, and this size is used for window notification to the destination.

Up to 512KiB can be specified as the receive buffer size for UDP, UDPP2P and RAW.

For the default value, refer to the "Socket Buffer Sizes" section in the "Internal Operations" chapter of the "libnet Overview" document.

SCE_NET_SO_SNDBUF

Send buffer size

Definition

```
int
```

Value

Send buffer size (bytes)

Description

This option relates to the send buffer size of a socket.

The send buffer size of TCP or TCP over UDPP2P can be specified up to 512 KiB.

Setting this option for UDP, UDPP2P and RAW is meaningless.

For the default value, refer to the "Socket Buffer Sizes" section in the "Internal Operations" chapter of the "libnet Overview" document.

SCE_NET_SO_RCVTIMEO

Receive timeout time

Definition

```
int
```

Value

<=0	Disable [default]
>0	Timeout time (microseconds)

Description

This option relates to the receive timeout time for a blocking socket.

When 0 seconds is specified for the timeout time, no timeout occurs. Set the non-blocking mode with the `SCE_NET_SO_NBIO` socket option when wishing to execute the target function without entering the wait state.

The timeout time applies to `sceNetRecv()`, `sceNetRecvfrom()`, and `sceNetRecvmsg()`, and `sceNetAccept()`.

SCE_NET_SO_SNDTIMEO

Send timeout time

Definition

```
int
```

Value

<=0	Disable [default]
>0	Timeout time (microseconds)

Description

This option relates to the send timeout time for a blocking socket.

When 0 seconds is specified for the timeout time, no timeout occurs. Set the non-blocking mode with the `SCE_NET_SO_NBIO` socket option when wishing to execute the target function without entering the wait state.

The timeout time applies to `sceNetConnect()`, `sceNetSend()`, `sceNetSendto()`, and `sceNetSendmsg()`.

SCE_NET_SO_REUSEADDR

Allow duplicate bindings for the same port

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (allow)

Description

This option determines the behavior when `sceNetBind()` is executed to bind a port to a socket when that port is already bound to an existing socket.

When `SCE_NET_SO_REUSEADDR` is enabled, the execution of duplicate bindings of local IP addresses and wildcards is allowed. The duplicate binding of multiple sockets to the same multicast address or same port is also allowed.

Notes

For servers receiving connection waiting for a TCP connection, it is recommended to enable the `SCE_NET_SO_REUSEADDR` option before executing `sceNetBind()` for the waiting socket.

SCE_NET_SO_REUSEPORT

Allow duplicate bindings for the same address and same port

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (allow)

Description

This option allows a socket to be bound to the same address and port that are already bound to an existing socket when `sceNetBind()` is executed. However, the `SCE_NET_SO_REUSEPORT` option must be enabled for all target sockets.

This option is used to receive UDP broadcast or multicast datagrams over multiple sockets.

Notes

When the target address to be bound is a multicast address, using the `SCE_NET_SO_REUSEADDR` option achieves the same operation as when the `SCE_NET_SO_REUSEPORT` option is allowed.

SCE_NET_SO_TYPE

Get socket type

Definition

```
int
```

Value

Socket type

Description

This option obtains one of the following socket types.

- `SCE_NET_SOCK_STREAM`
- `SCE_NET_SOCK_DGRAM`
- `SCE_NET_SOCK_RAW`
- `SCE_NET_SOCK_STREAM_P2P`
- `SCE_NET_SOCK_DGRAM_P2P`

SCE_NET_SO_NBIO

Set non-blocking

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (non-blocking)

Description

This option determines the non-blocking operation for a socket. It is possible to change to blocking mode even after setting to non-blocking mode.

SCE_NET_SO_ONESBCAST

Handling of conversion of broadcast send address

Definition

```
int
```

Value

0	Disable (perform conversion) [default]
Non-zero	Enable (destination broadcast address is used as is)

Description

When this option is disabled and data is sent to address 255.255.255.255 with the send function, the address is converted to the broadcast address that was set for the interface.

To explicitly send data to the address 255.255.255.255, enable this option.

SCE_NET_SO_USECRYPTO

Encrypt and decrypt data

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (encrypt and decrypt)

Description

This option relates to whether or not to encrypt data when sending and decrypt data when receiving. It is valid only when the socket type is `SCE_NET_SOCK_DGRAM_P2P` or `SCE_NET_SOCK_STREAM_P2P`.

When the option is enabled, a 4-byte initial vector is assigned to each packet.

SCE_NET_SO_USESIGNATURE

Generate and verify data signatures

Definition

```
int
```

Value

0	Disable [default]
Non-zero	Enable (generate and verify signature)

Description

This option relates to whether or not to generate a signature for data when sending and verify a signature for data when receiving. It is valid only when the socket type is `SCE_NET_SOCK_DGRAM_P2P` or `SCE_NET_SOCK_STREAM_P2P`.

When the option is enabled, a 4-byte signature is assigned to each packet.

SCE_NET_SO_TPPOLICY

Policy number of socket

Definition

```
int
```

Value

Policy number (0 [default] - 31)

Description

This option is used with network emulation. Refer to the "libnet Overview" document for details.

SCE_NET_SO_NAME

Debug name of socket

Definition

```
char *
```

Value

Character string (maximum character count is 31 characters, not including termination character)

Description

This option sets the debug name of sockets , etc for which `sceNetAccept()` was executed. The length of the character string not including the termination character is passed.

SCE_NET_IP_MULTICAST_IF

Specify IPv4 multicast datagram send interface

Definition

```
SceNetInAddr
```

Value

`SCE_NET_INADDR_ANY`	Reference the route control table [default]
IPv4 address	Send interface address (network byte order)

Description

This option relates to the specification of the interface for sending an IPv4 multicast datagram. It is supported when the socket type is `SCE_NET_SOCK_DGRAM`.

SCE_NET_IP_MULTICAST_TTL

TTL value of IPv4 multicast datagram

Definition

`unsigned char` or `int`

Value

TTL (default value = 1, "within local network")

Description

This option relates to the TTL value when sending an IPv4 multicast datagram.

SCE_NET_IP_MULTICAST_LOOP

Existence of local loopback for IPv4 multicast datagram

Definition

`unsigned char` or `int`

Value

0	No loopback
1	Loopback [default]

Description

This option relates to whether or not there is a local loopback for an outgoing IPv4 multicast datagram. If this option is enabled and the send interface joins the destination multicast group of a send datagram, the send datagram is copied and processed as a receive datagram.

SCE_NET_IP_ADD_MEMBERSHIP

Join IPv4 multicast group

Definition

`SceNetIpMreq`

Value

imr_multiaddr	Address of multicast group to join
imr_interface	Receive interface address

Description

This option allows a specified interface to join an IPv4 multicast group.

The specification method of the receive interface is the same as that of the `SCE_NET_IP_MULTICAST_IF`.

SCE_NET_IP_DROP_MEMBERSHIP

Leave IPv4 multicast group

Definition

```
SceNetIpMreq
```

Value

imr_multiaddr	Address of IP multicast group to leave
imr_interface	Receive interface address

Description

This option allows a specified interface to leave an IPv4 multicast group.

The specification method of the receive interface is the same as that of `SCE_NET_IP_MULTICAST_IF`.

If a socket that belongs to a group is not explicitly withdrawn, the socket is withdrawn automatically when the socket is closed.

SCE_NET_IP_HDRINCL

Add send IP header when using RAW socket

Definition

```
int
```

Value

0	Do not include IP header in user payload [default]
Non-zero	Include IP header in user payload

Description

This option relates to whether or not to specify a send IP header when using a RAW socket. It is supported when the socket type is SCE_NET_SOCK_RAW. When the setting is enabled, data to be sent to the socket must start with an IP header. In addition, the IP header is changed according to the following conditions.

- IP header checksum: IP header is changed without exception.
- Sender IP address: in the case of '0', the IP header is changed to its own IP address.
- IP header ID: in the case of '0', the IP header is changed.

SCE_NET_IP_TTL

TTL value of IP header of send data

Definition

```
int
```

Value

TTL value

Description

This option relates to the TTL value of the IP header of send data. This applies to all sockets, but this does not have an effect when creating an IP header with `SCE_NET_IP_HDRINCL` for a RAW socket.

SCE_NET_IP_TTLCHK

Start and end recording of maximum TTL value of incoming packets

Definition

```
int
```

Value

0	End recording [default]
Non-zero	Start recording

Description

This option is for recording the maximum TTL value of incoming packets. It applies to all sockets except RAW sockets.

SCE_NET_IP_MAXTTL

Maximum TTL value of incoming packets

Definition

```
int
```

Value

TTL value

Description

This option obtains the maximum TTL value of incoming packets in an interval recorded with the `SCE_NET_IP_TTLCHK` option. The TTL value is initialized to 0 at the start of recording, and the TTL value is held after the end of recording. It applies to all sockets except RAW sockets.

SCE_NET_IP_DONTFRAG

IP header Don’t Fragment flag value

Definition

```
int
```

Value

0	Permit fragmenting with IP level [default]
Non-0	Prohibit fragmenting with IP level

Description

This option is related to the IP Don’t Fragment flag of send packets.

It is applied to UDP, UDPP2P, and RAW sockets. However, it does not have an effect for RAW sockets when creating an IP header with `SCE_NET_IP_HDRINCL`.

SCE_NET_IP_TOS

Type-Of-Service (TOS) field value

Definition

```
int
```

Value

TOS value

Description

This option relates to the IP TOS value of outgoing packets. It applies to all sockets. However, this does not have an effect when creating an IP header with `SCE_NET_IP_HDRINCL` for a RAW socket.

SCE_NET_TCP_NODELAY

Prohibit use of TCP Nagle algorithm

Definition

```
int
```

Value

0	Use Nagel algorithm [default]
Non-zero	Prohibit use

Description

This option determines whether or not to prohibit use of the TCP Nagle algorithm. It is valid only when the socket type is `SCE_NET_SOCK_STREAM` or `SCE_NET_SOCK_STREAM_P2P`. This option can be set at any time.

The Nagle algorithm prevents the sending of data smaller than the maximum segment size (MSS) if there is send data that has not received an ACK. In addition, when an ACK is sent to the receive data, a delayed ACK is sent after waiting a maximum of 200 milliseconds. As a result, when using the Nagle algorithm with an application that continuously sends and receives small packets over short periods of time, the response time may appear to become longer.

SCE_NET_TCP_MAXSEG

Maximum segment size (MSS)

Definition

```
int
```

Value

Maximum segment size (bytes)

Description

This option relates to the maximum segment size (MSS) of a TCP connection. With this option, the packet size sent by own socket is controlled so as not to exceed the specified MSS.

It is valid only when the socket type is `SCE_NET_SOCK_STREAM` or `SCE_NET_SOCK_STREAM_P2P`. This can be set after establishing a connection.

SCE_NET_TCP_MSS_TO_ADVERTISE

MSS value to be reported to destination by initial packet (MSS option value of SYN packet)

Definition

```
unsigned short
```

Value

0	MSS value of interface [default]
1 or higher	MSS value for reporting (bytes)

Description

This option relates to the maximum segment size (MSS) that is contained in the SYN packet when a TCP connection is established. With this option, the packet size sent by the other terminal is controlled so as not to exceed the specified MSS.

It is valid only when the socket type is `SCE_NET_SOCK_STREAM` or `SCE_NET_SOCK_STREAM_P2P`. This must be set before trying to establish a connection. For example, by setting this option to the listening socket that has called `sceNetAccept()`, the MSS value specified for the connected socket that has been obtained with `sceNetAccept()` is applied.

If the set MSS value is not an appropriate value, such as when it exceeds the MSS value of the interface, the value is adjusted internally.

Error Codes

sce_net_errno

Get or set network error value (network errno)

Definition

```
#include <net.h>
int *sceNetErrnoLoc(void);
#define sce_net_errno (*sceNetErrnoLoc())
```

Arguments

None

Return Values

Most recent libnet error value

Description

This macro obtains the value of the error that occurred most recently with libnet.

The value can be overwritten.

Error values are retained separately by thread.

Note that the value is updated only when an error occurs with libnet and that the value is not cleared when there are no errors.

Error Codes

Error codes obtained with sce_net_errno

Definition

The following are the common libnet error codes.

As a general rule, the error codes of errors that can occur with each function are listed with each function, but common error codes that can occur with all functions, such as that for insufficient memory (SCE_NET_ENOMEM), have been omitted from the function descriptions (indicated as [common]). [reserved] indicates error codes that are currently not returned.

Value	(Number)	Description
None [common]	-1	Internal error for protocol stack
SCE_NET_ENOENT	2	No resources are in wait state
SCE_NET_EINTR	4	Blocking canceled by abort function
SCE_NET_EBADF	9	Invalid libnet ID specified
SCE_NET_ENOMEM [common]	12	Insufficient memory (kernel)
SCE_NET_EACCES	13	Attempted to use an area reserved by the system Attempted to send to a broadcast address
SCE_NET_EFAULT	14	Invalid argument specified
SCE_NET_ENOTBLK	15	Abort process called while target is not in wait state
SCE_NET_EBUSY	16	libnet already initialized
SCE_NET_EEXIST	17	SCE_NET_EPOLL_CTL_ADD specified to previously associated libnet ID
SCE_NET_ENODEV	19	Target device does not exist
SCE_NET_EINVAL	22	Invalid argument specified
SCE_NET_EMFILE	24	Insufficient space in socket table
SCE_NET_ENOSPC	28	Size specified with *dst* is too small to store string
SCE_NET_EPIPE	32	Writing side of socket already closed
SCE_NET_EAGAIN SCE_NET_EWOULDBLOCK	35	Socket is in blocking state (when non-blocking) Timeout occurred (when SCE_NET_SO_SNDTIMEO or SCE_NET_SO_RCVTIMEO option is specified)
SCE_NET_EINPROGRESS	36	Attempting to establish a connection
SCE_NET_EALREADY	37	Socket is already in use
SCE_NET_EDESTADDRREQ	39	Invalid send request (sceNetSendto() should be used)
SCE_NET_EMSGSIZE	40	Message size is too large
SCE_NET_EPROTOTYPE	41	Unsupported protocol type was specified
SCE_NET_ENOPROTOOPT	42	Option is not supported
SCE_NET_EPROTONOSUPPORT	43	Invalid protocol family
SCE_NET_EOPNOTSUPP	45	Invalid call for that socket
SCE_NET_EPFNOSUPPORT [reserved]	46	Unsupported protocol family was specified
SCE_NET_EAFNOSUPPORT	47	Value of specified address family is not supported by socket protocol family
SCE_NET_EADDRINUSE	48	Attempted to bind to bound port
SCE_NET_EADDRNOTAVAIL	49	Invalid address specified
SCE_NET_ENETDOWN	50	Interface is down
SCE_NET_ENETUNREACH	51	Destination is unreachable
SCE_NET_ECONNABORTED	53	Connection was aborted
SCE_NET_ECONNRESET	54	Connection was reset
SCE_NET_ENOBUFS [common]	55	Memory limited (kernel work area is insufficient) (refer to "libnet Overview" document)

Value	(Number)	Description
SCE_NET_EISCONN	56	Specified connection is already established
SCE_NET_ENOTCONN	57	Specified connection does not exist
SCE_NET_ESHUTDOWN [reserved]	58	Shutdown in progress
SCE_NET_ETOOMANYREFS	59	Too many multicast addresses specified
SCE_NET_ETIMEDOUT	60	Timeout occurred (indicates a protocol timeout, unlike SCE_NET_EAGAIN)
SCE_NET_ECONNREFUSED	61	Connection request was denied
SCE_NET_EHOSTDOWN	64	Did not reach other side
SCE_NET_EHOSTUNREACH	65	Network unreachable
SCE_NET_ENOTSUP	86	This call is invalid (SDK 2.000 or later)
SCE_NET_ECANCELED	87	Close processing was called for a socket that is in the wait condition and being executed
SCE_NET_EADHOC	160	UDP or TCP was attempted in the ad hoc communication mode
SCE_NET_EDISABLEDIF [common]	161	(Internal error)
SCE_NET_ERESUME [reserved]	162	The sockets were recovered by the system between process suspend and process resume (sceNetSocketClose() must be called)
SCE_NET_EIPADDRCHANGED SCE_NET_EINACTIVEDISABLED	163	Network disconnection occurred owing to intermittent disconnection or system suspend. (sceNetSocketClose() must be called)
SCE_NET_ENOTINIT	200	libnet not initialized
SCE_NET_ENOLIBMEM	201	Insufficient memory (library)
SCE_NET_ECALLBACK	203	(Internal error)
SCE_NET_EINTERNAL	204	Fatal internal error
SCE_NET_ERETURN	205	libnetctl error was returned

The following are the error codes related to the DNS resolver.

Value	(Number)	Description
SCE_NET_RESOLVER_EINTERNAL	220	Fatal internal error
SCE_NET_RESOLVER_EBUSY	221	Resolver was in use
SCE_NET_RESOLVER_ENOSPACE	222	Insufficient memory (library)
SCE_NET_RESOLVER_EPACKET	223	Invalid DNS response
SCE_NET_RESOLVER_ENODNS	225	DNS server not specified
SCE_NET_RESOLVER_ETIMEDOUT	226	Timeout occurred
SCE_NET_RESOLVER_ENOSUPPORT	227	Unsupported feature requested by server
SCE_NET_RESOLVER_EFORMAT	228	Invalid response from server
SCE_NET_RESOLVER_ESERVERFAILURE	229	Temporary error from server
SCE_NET_RESOLVER_ENOHOST	230	Inquired host name does not exist
SCE_NET_RESOLVER_ENOTIMPLEMENTED	231	Inquired feature is not implemented
SCE_NET_RESOLVER_ESERVERREFUSED	232	Inquiry denied
SCE_NET_RESOLVER_ENORECORD	233	Inquired record does not exist
SCE_NET_RESOLVER_EALIGNMENT	234	Invalid alignment

Notes

Sce error codes correspond to the following values.

Value	(Number)
SCE_NET_ERROR_ENOENT	0x80410102
SCE_NET_ERROR_EINTR	0x80410104
SCE_NET_ERROR_EBADF	0x80410109
SCE_NET_ERROR_ENOMEM	0x8041010c
SCE_NET_ERROR_EACCES	0x8041010d
SCE_NET_ERROR_EFAULT	0x8041010e
SCE_NET_ERROR_ENOTBLK	0x8041010f
SCE_NET_ERROR_EBUSY	0x80410110
SCE_NET_ERROR_EEXIST	0x80410111
SCE_NET_ERROR_ENODEV	0x80410113
SCE_NET_ERROR_EINVAL	0x80410116
SCE_NET_ERROR_EMFILE	0x80410118
SCE_NET_ERROR_ENOSPC	0x8041011c
SCE_NET_ERROR_EPIPE	0x80410120
SCE_NET_ERROR_EAGAIN SCE_NET_ERROR_EWOULDBLOCK	0x80410123
SCE_NET_ERROR_EINPROGRESS	0x80410124
SCE_NET_ERROR_EALREADY	0x80410125
SCE_NET_ERROR_EDESTADDRREQ	0x80410127
SCE_NET_ERROR_EMSGSIZE	0x80410128
SCE_NET_ERROR_EPROTOTYPE	0x80410129
SCE_NET_ERROR_ENOPROTOOPT	0x8041012a
SCE_NET_ERROR_EPROTONOSUPPORT	0x8041012b
SCE_NET_ERROR_EOPNOTSUPP	0x8041012d
SCE_NET_ERROR_EPFNOSUPPORT	0x8041012e
SCE_NET_ERROR_EAFNOSUPPORT	0x8041012f
SCE_NET_ERROR_EADDRINUSE	0x80410130
SCE_NET_ERROR_EADDRNOTAVAIL	0x80410131
SCE_NET_ERROR_ENETDOWN	0x80410132
SCE_NET_ERROR_ENETUNREACH	0x80410133
SCE_NET_ERROR_ECONNABORTED	0x80410135
SCE_NET_ERROR_ECONNRESET	0x80410136
SCE_NET_ERROR_ENOBUFS	0x80410137
SCE_NET_ERROR_EISCONN	0x80410138
SCE_NET_ERROR_ENOTCONN	0x80410139
SCE_NET_ERROR_ESHUTDOWN	0x8041013a
SCE_NET_ERROR_ETOOMANYREFS	0x8041013b
SCE_NET_ERROR_ETIMEDOUT	0x8041013c
SCE_NET_ERROR_ECONNREFUSED	0x8041013d
SCE_NET_ERROR_EHOSTDOWN	0x80410140
SCE_NET_ERROR_EHOSTUNREACH	0x80410141
SCE_NET_ERROR_ENOTSUP	0x80410156
SCE_NET_ERROR_ECANCELED	0x80410157
SCE_NET_ERROR_EADHOC	0x804101a0
SCE_NET_ERROR_EDISABLEDIF	0x804101a1
SCE_NET_ERROR_ERESUME	0x804101a2
SCE_NET_ERROR_EIPADDRCHANGED SCE_NET_ERROR_EINACTIVEDISABLED	0x804101a3
SCE_NET_ERROR_ENOTINIT	0x804101c8

Value	(Number)
SCE_NET_ERROR_ENOLIBMEM	0x804101c9
SCE_NET_ERROR_ECALLBACK	0x804101cb
SCE_NET_ERROR_EINTERNAL	0x804101cc
SCE_NET_ERROR_ERETURN	0x804101cd
SCE_NET_ERROR_RESOLVER_EINTERNAL	0x804101dc
SCE_NET_ERROR_RESOLVER_EBUSY	0x804101dd
SCE_NET_ERROR_RESOLVER_ENOSPACE	0x804101de
SCE_NET_ERROR_RESOLVER_EPACKET	0x804101df
SCE_NET_ERROR_RESOLVER_ENODNS	0x804101e1
SCE_NET_ERROR_RESOLVER_ETIMEDOUT	0x804101e2
SCE_NET_ERROR_RESOLVER_ENOSUPPORT	0x804101e3
SCE_NET_ERROR_RESOLVER_EFORMAT	0x804101e4
SCE_NET_ERROR_RESOLVER_ESERVERFAILURE	0x804101e5
SCE_NET_ERROR_RESOLVER_ENOHOST	0x804101e6
SCE_NET_ERROR_RESOLVER_ENOTIMPLEMENTED	0x804101e7
SCE_NET_ERROR_RESOLVER_ESERVERREFUSED	0x804101e8
SCE_NET_ERROR_RESOLVER_ENORECORD	0x804101e9
SCE_NET_ERROR_RESOLVER_EALIGNMENT	0x804101ea