PlayStation®2 IOP Library Reference
Release 2.4

Common Network Configuration Library

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

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

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

Summary Table of Contents

About This Manual

This is the Runtime Library Release 2.4 version of the *PlayStation®2 IOP Library Reference - Common Network Configuration Library* manual.

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

Changes Since Last Release

Chapter 1: Common Network Configuration Library

- In the "Description" section of the sceNetCnfInterface structure, descriptions of the values set to the {want,allow}.auth and force_chap_type members have been added. Arguments of keywords corresponding to the members with value descriptions and a correspondence table of the values have been added.
- In the "Description" section of sceNetCnfLoadEntry(), error correction has been made to the names of the members of the sceNetCnfEnv structure.
- In the "Return Value" section of the following functions, a description of sceNETCNF_IO_ERROR (I/O error occurrence) has been added.

  sceNetCnfAddEntry()
  sceNetCnfDeleteAll()
  sceNetCnfDeleteEntry()
  sceNetCnfEditEntry()
  sceNetCnfGetCount()
  sceNetCnfGetList()
  sceNetCnfLoadConf()
  sceNetCnfLoadDial()
  sceNetCnfLoadEntry()
  sceNetCnfSetLatestEntry()

Related Documentation

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

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

Typographic Conventions

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

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

Developer Support

Sony Computer Entertainment America (SCEA)

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

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

Sony Computer Entertainment Europe (SCEE)

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

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

Chapter 1: Common Network Configuration Library
Table of Contents

Configuration File Structures

sceNetCnfAddress

Internal format IP address

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

Structure

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

Description

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

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

sceNetCnfCommand

Routing configuration information

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

Structure

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

Description

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

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

Table 1-1

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

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

See also

sceNetCnfInterface

sceNetCnfCtl

Configuration control information

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

Structure

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

Description

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

sceNetCnfDial

Dialing definition information

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

Structure

typedef struct sceNetCnfDial {

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

} sceNetCnfDial_t;

Description

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

sceNetCnfEnv

Load/save environment

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

Structure

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

Description

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

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

See also

sceNetCnfLoadEntry(), sceNetCnfAddEntry()

sceNetCnfInterface

Configuration information for each interface

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

Structure

typedef struct sceNetCnfInterface {

int *type;*

u_char **vendor;*

u_char **product;*

u_char **location;*

u_char *dhcp;*

u_char **dhcp_host_name;*

u_char *dhcp_host_name_null_terminated;*

u_char *dhcp_release_on_stop;*

u_char **address;*

u_char **netmask;*

u_char **chat_additional;*

int *redial_count;*

int *redial_interval;*

u_char **outside_number;*

u_char **outside_delay;*

u_char **phone_numbers*
[sceNetCnf_MAX_PHONE_NUMBERS];

u_char *answer_mode;*

int *answer_timeout;*

int *dialing_type;*

u_char **chat_login;*

u_char **auth_name;*

u_char **auth_key;*

u_char **peer_name;*

u_char **peer_key;*

int *lcp_timeout;*

int *ipcp_timeout;*

int *idle_timeout;*

int *connect_timeout;*

struct {

u_char *mru_nego;*

u_char *accm_nego;*

u_char *magic_nego;*

u_char *prc_nego;*

u_char *acc_nego;*

u_char *address_nego;*

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

Description

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

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

Table 1-2

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

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

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

The following represent the entries in the Data Type column.

Table 1-3

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

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

type can have any of the following values.

Table 1-4

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

dialing_type can have any of the following values.

Table 1-5

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

phy_config can have any of the following values.

Table 1-6

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

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

Table 1-7

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

force_chap_type can have any of the following values.

Table 1-8

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

See also

sceNetCnfCtl, sceNetCnfPair, scenetCnfEnv, sceNetCnfInitIFC()

sceNetCnfList

Configuration management file data

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

Structure

typedef struct sceNetCnfList {

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

} sceNetCnfList_t;

Description

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

See also

sceNetCnfGetList()

sceNetCnfPair
interface keyword information

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

Structure

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

Description

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

sceNetCnfRoot

NET_CNF information

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

Structure

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

Description

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

sceNetCnfRoutingEntry

Routing control table entry

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

Structure

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

Description

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

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

Table 1-9

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

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

See also

sceNetCnfAddress

sceNetCnfUnknown

Undefined keyword and argument data

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

Structure

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

Description

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

See also

sceNetCnfUnknownList

sceNetCnfUnknownList

Undefined keyword and argument list

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

Structure

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

Description

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

See also

sceNetCnfInterface

Configuration File Functions

sceNetCnfAddEntry

Add entry to configuration management file

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

Syntax

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

Calling conditions

Can be called from a thread.

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

Description

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

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

Combination"index"

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

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

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

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

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

Return value

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

sceNetCnfAddress2String

Conversion from internal-format IP address to dot format

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

Syntax

#include <netcnf.h>

int sceNetCnfAddress2String(

char **buf,*	Address of buffer where the conversion result will be stored
int *len,*	Buffer length (bytes)
sceNetCnfAddress_t **paddr);*	Internal-format IP address

Calling conditions

Can be called from a thread.

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

Description

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

This function is used for display and debugging.

Return value

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

sceNetCnfAllocMem

Allocate memory area

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

Syntax

#include <netcnf.h>

void *sceNetCnfAllocMem(

sceNetCnfEnv_t **e,*	Load/save environment
int *size,*	Number of bytes of memory to be allocated
int *align***);**	Alignment of beginning of memory area to be allocated 0: Byte alignment 2: Word alignment

Calling conditions

Can be called from a thread.

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

Description

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

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

Return value

!= NULL	Allocation was successful
== NULL	Allocation failed

sceNetCnfDeleteAll

Delete all common network settings

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

Syntax

#include <netcnf.h>

int sceNetCnfDeleteAll(

char **dev***);**	Device name (only "mc?:" and "pfs?:" are supported)

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

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

sceNetCnfDeleteEntry

Delete entry from configuration management file

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

Syntax

#include < netcnf.h>

int sceNetCnfDeleteEntry(

char **fname,*	Pathname of configuration management file
int *type,*	File type 0: Connection environment configuration file 1: Connection configuration file 2: Modem configuration file
char **usr_name*);	Current configuration name

Calling conditions

Can be called from a thread.

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

Description

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

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

Combination"index"

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

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

Return value

0 <	Deletion was successful
sceNETCNF_INVALID_USR_NAME	*usr_name* was invalid

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

sceNetCnfEditEntry

Edit entry in configuration management file

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

Syntax

#include <netcnf.h>

int sceNetCnfEditEntry(

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

Calling conditions

Can be called from a thread.

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

Description

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

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

Combination"index"

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

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

“index” must not be between 1 and 10.

Other

“index” must not be between 1 and 1000.

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

Notes

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

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

Return value

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

Table 1-10

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

sceNetCnfGetCount

Get number of files

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

Syntax

#include <netcnf.h>

int sceNetCnfGetCount(

char **fname,*	Pathname of configuration management file
int *type***);**	File type 0: Connection environment configuration file 1: Connection configuration file 2: Modem configuration file

Calling conditions

Can be called from a thread.

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

Description

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

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

Return value

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

sceNetCnfGetList

Get file list

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

Syntax

#include <netcnf.h>

int sceNetCnfGetList(

char **fname,*	Pathname of configuration management file
int *type,*	File type 0: Connection environment configuration file 1: Connection configuration file 2: Modem configuration file
sceNetCnfList_t **p*);	Pointer to beginning of file list

Calling conditions

Can be called from a thread.

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

Description

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

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

Return value

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

sceNetCnfInitIFC

Initialize configuration information for each interface

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

Syntax

#include <netcnf.h>

int sceNetCnfInitIFC(

sceNetCnfInterface_t **ifc***);**	Pointer to configuration information for each interface to be initialized

Calling conditions

Can be called from a thread.

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

Description

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

Return value

Always 0.

sceNetCnfLoadConf

Load configuration file

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

Syntax

#include <netcnf.h>

int sceNetCnfLoadConf(

sceNetCnfEnv_t **e***);**	Load environment

Calling conditions

Can be called from a thread.

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

Description

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

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

Notes

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

Return value

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

Table 1-11

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

sceNetCnfLoadDial

Load dialing definition file

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

Syntax

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

Calling conditions

Can be called from a thread.

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

Description

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

Notes

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

Return value

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

Table 1-12

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

sceNetCnfLoadEntry

Load configuration file

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

Syntax

#include <netcnf.h>

int sceNetCnfLoadEntry(

char **fname,*	Pathname of configuration management file
int *type,*	File type 0: Connection environment configuration file 1: Connection configuration file 2: Modem configuration file
char **usr_name,*	Configuration name
sceNetCnfEnv_t **e***);**	Load environment

Calling conditions

Can be called from a thread.

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

Description

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

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

Combination"index"

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

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

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

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

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

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

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

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

Return value

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

sceNetCnfMergeConf

Merge configuration data

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

Syntax

#include <netcnf.h>

int sceNetCnfMergeConf(

sceNetCnfEnv_t **e***);**	Load environment

Calling conditions

Can be called from a thread.

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

Description

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

Notes

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

Return value

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

Table 1-13

Constant	Meaning
sceNETCNF_ALLOC_ERROR	Attempt to allocate memory failed

sceNetCnfName2Address

Convert internal-format IP address

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

Syntax

#include <netcnf.h>

int sceNetCnfName2Address(

sceNetCnfAddress_t **paddr,*	Address of structure variable for receiving internal-format IP address
char **name,* **);**	Dot-format IP address

Calling conditions

Can be called from a thread.

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

Description

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

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

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

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

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

Return value

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

sceNetCnfSetLatestEntry

Change list position in configuration management file

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

Syntax

#include <netcnf.h>

int sceNetCnfSetLatestEntry(

char **fname,*	Pathname of configuration management file
int *type,*	File type 0: Connection environment configuration file 1: Connection configuration file 2: Modem configuration file
char **usr_name***);**	Configuration name

Calling conditions

Can be called from a thread.

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

Description

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

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

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

mc?:/BWNETCNF/BWNETCNF

pfs?:/etc/network/net.db

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

Combination"index"

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

All common devices

"index" must be 5 digits or more.

PS2 Memory card

"index" must not be between 1 and 6.

Hard disk drive

"index" must not be between 1 and 10.

Other

"index" must not be between 1 and 1000.

Return value

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