PlayStation®2 EE Library Reference
Release 2.4.2

Network Libraries

Summary Table of Contents

About This Manual

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

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

Changes Since Last Release

Chapter 2: HTTP Library

- New

Chapter 3: Network Socket Library

- Error code details have been added to the return value of the sceInsockErrno() function.

Chapter 4: General-Purpose Network Wrapper API (netglue)

- New

Chapter 5: Network Configuration GUI Library

- An explanation has been added to the peer_name member of the sceNetGuiCnfEnvData structure.

Related Documentation

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

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

Typographic Conventions

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

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

Developer Support

Sony Computer Entertainment America (SCEA)

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

Order Information	Developer Support
In North America:	*In North America:*
Attn: Developer Tools Coordinator 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: dev9 Reference (for networks)
Table of Contents

Functions

sceDevctl

Special operations on device

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

Syntax

#include <sifdev.h>

int sceDevctl(

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

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

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

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

Errors that are common to each command are as follows:

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

devctl Commands

DDIOC_MODEL

Flush disk cache

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

Syntax

arg	Reserved. Specify NULL.
arglen	arg size.
bufp	Reserved. Specify NULL.
buflen	bufp size.

Description

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

Return value

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

DDIOC_OFF

Power off device

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

Syntax

arg	Reserved. Specify NULL.
arglen	arg size.
bufp	Reserved. Specify NULL.
buflen	bufp size.

Description

This command powers off the entire dev9 device.

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

Return value

When processing succeeds, 0 is returned.

Chapter 2: HTTP Library
Table of Contents

Structures

sceHTTPAuth_t

Authentication structure

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

Structure

typedef struct sceHTTPAuth {	
int *type***;**	Authentication type
char ****realm*;**	realm string
char *****domains*;**	Domain string array
char ****uri*;**	URI string
char ****nonce*;**	nonce string
char ****opaque*;**	opaque string
int *stale***;**	stale value
int *algorithm***;**	Algorithm
int *qop***;**	QOP value
} sceHTTPAuth_t;	

Description

This structure represents the authentication challenge from the server.

The authentication type is represented as an integer, and the following constant definitions indicate basic authentication and digest authentication, respectively.

sceHTTPAuth_BASIC	0
sceHTTPAuth_DIGEST	1

For basic authentication, only the *type* and *realm* fields are used. For digest authentication, the meanings of the various fields are the same as those specified for the WWW-Authenticate header in RFC2617.

The value of the domain parameter is represented by a string array because it generally contains multiple domain names. The element following the last domain name in this array is a NULL pointer.

The *stale* value is represented by the following constant definitions. A value of 0 means that there is no *stale* parameter.

sceHTTPDigestStale_TRUE	1
sceHTTPDigestStale_FALSE	2

The algorithm value is represented by the following constant definitions. A value of 0 value means that there is no *algorithm* parameter.

sceHTTPDigestAlg_MD5	1
sceHTTPDigestAlg_MD5SESS	2

The QOP value is represented by the logical OR of the following bit flags.

sceHTTPDigestQOP_AUTH	1
sceHTTPDigestQOP_AUTHINT	2

See also

sceHTTPFreeAuthList(), sceHTTPParseAuth()

sceHTTPAuthInfo_t

Digest authentication information structure

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

Structure

typedef struct sceHTTPAuthInfo {	
char **nextnonce*;	nextnonce string
char **rspauth*;	rspauth string array
char **cnonce*;	cnonce string
int *nc*;	nc (nonce count) value
int *qop*;	QOP value
} sceHTTPAuthInfo_t;	

Description

This structure represents authentication confirmation information when digest authentication is used.

The meanings of the various fields are the same as those specified for the Authentication-Info header in RFC2617.

For *qop*, please refer to the description of sceHTTPAuth_t.

See also

sceHTTPParseAuthInfo(), sceHTTPAuth_t

sceHTTPAuthList_t

Authentication list structure

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

Structure

typedef struct sceHTTPAuthList {	
struct sceHTTPAuthList **forw*, **back*;	forw: Forward link back:Backward link
struct sceHTTPAuth *auth*;	Authentication challenge structure
} sceHTTPAuthList_t;	

Description

This structure represents a doubly-linked list of authentication challenges.

See also

sceHTTPFreeAuthList(), sceHTTPParseAuth()

sceHTTPClient_t

HTTP client structure

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

Structure

typedef struct sceHTTPClient {

char **name*;	User agent name
int *http_ver*;	HTTP protocol version
int *http_rev*;	HTTP protocol revision
int *rtimeout*;	Response timeout (seconds)
int *ttimeout*;	Data transfer timeout (seconds)
int *laptime*;	Input/output lap time
int *prot*;	Protocol
int *state*;	Transaction state
int *errnum*;	Error number
int *net_errno*;	Network error number
int *reloading*;	Reload flag
int *keep_alive*;	Connection hold time (seconds)
int *keep_count*;	Connection hold count
int *non_blocking*;	Non-blocking mode
int *abort_req*;	Abort request flag
int *t_stacksize*;	Transaction thread stack size
int *t_priority*;	Transaction thread priority
int *t_thread*;	Transaction thread ID
void **t_stack*;	Transaction thread stack
int *t_rtn*;	Transaction termination code
void (**t_notify*)**(int flags);**	Transaction termination notification callback function
unsigned int *max_olength*;	Maximum response data length
struct sceHTTPParsedURI_t **proxy*;	Parsed proxy URI
sceHTTPMethod_t *method*;	HTTP request method
struct sceHTTPParsedURI_t **puri*;	Parsed URI
sceHTTPHeaderList_t **iheaders*;	Request header
char **idata*;	Request data
int *ilength*;	Request data length
int *iflags*;	Request data flag
sceHTTPResponse_t *response*;	Response structure
void (**chunkf*)**(struct sceHTTPClient *, unsigned char *, unsigned int);**	Chunk receive notification callback function
int *recv_thread*;	Receive thread ID
int *send_thread*;	Send thread ID
void **io_rstack*;	Receive thread stack
void **io_sstack*;	Send thread stack

int *io_desc***;**	Socket descriptor
char * *io_buf***;**	Input/output buffer
int *io_len***;**	Input/output buffer length
int *io_rtn***;**	Input/output return value
int *io_timer***;**	Input/output timer ID
int *io_rwait***,** *io_rdone***;**	io_rwait: Receive request semaphore ID io_rdone: Receive complete semaphore ID
int *io_swait***,** *io_sdone***;**	io_swait: Send request semaphore ID io_sdone: Send complete semaphore ID
int *io_flags***;**	Input/output flag
int *io_tcount***;**	Input/output timer counter

} sceHTTPClient_t;

Description

This structure is used by the HTTP client to perform transactions.

The user cannot directly access the members of this structure.

See also

sceHTTPCreate(), sceHTTPDestroy(), sceHTTPRequest(), sceHTTPGetResponse(), sceHTTPGetOption(), sceHTTPSetOption(), sceHTTPOpen(), sceHTTPClose(), sceHTTPGetClientError(), sceHTTPParsedURI_t, sceHTTPHeaderList_t

sceHTTPCookie_t

Cookie structure

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

Structure

typedef struct sceHTTPCookie {

char **name*;	Name
char **value*;	Value
char **domain*;	Valid domain
char **path*;	Valid path
int *expires*;	Expiration
int *secure*;	Secure flag
int *version*;	Version
int *maxage*;	Expiration (version 1 type)

} sceHTTPCookie_t;

Description

This structure represents a cookie.

The value of the *expires* member is the time in seconds since January 1, 1970 in GMT.

A non-zero secure flag value means that communication with the server must be performed securely when this cookie is used.

The *maxage* member, which is used when this cookie is of the type specified in RFC2109 (the *version* is 1), contains the number of seconds from the current time.

See also

sceHTTPsceHTTPAddCookieList(), sceHTTPParseCookie(), sceHTTPSetCookie()

sceHTTPCookieList_t

Cookie list structure

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

Structure

typedef struct sceHTTPCookieList {	
struct sceHTTPCookieList **forw*, **back*;	forw: Forward link
	back: Backward link
struct sceHTTPCookie *cookie*;	Cookie structure
} sceHTTPCookieList_t;	

Description

This structure represents a doubly-linked list of cookies.

See also

sceHTTPsceHTTPAddCookieList(), sceHTTPFreeCookieList(), sceHTTPParseCookie(), sceHTTPSetCookie(), sceHTTPCookie_t()

sceHTTPDigest_t

Digest authentication request structure

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

Structure

typedef struct sceHTTPDigest {

char **username*;	User name string
char **realm*;	realm string
char **password*;	Password string
char **uri*;	URI string
char **nonce*;	nonce string
char **cnonce*;	cnonce string
char **opaque*;	opaque string
int *algorithm*;	Algorithm
int *nc*;	Count (integer)
int *qop*;	QOP value
int *method*;	HTTP method
char **entity*;	Pointer to data byte string
int *length*;	Data string length

} sceHTTPDigest_t;

Description

This structure is used for digest authentication requests.

The meanings of the fields other than *method*, *entity*, and *length* are the same as those specified in RFC2617. Also, for *algorithm* and *qop*, the same constants are defined as those described for sceHTTPAuth.

The *entity* and *length* fields are used only when the QOP value is sceHTTPDigestQOP_AUTHINT.

See also

sceHTTPSetDigestAuth(), sceHTTPAuth_t()

sceHTTPHeaderList_t

Header list structure

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

Structure

typedef struct sceHTTPHeaderList {	
struct sceHTTPHeaderList **forw*, **back*;	forw: Forward link back: Backward link
char **name*;	Name
char **value*;	String value
} sceHTTPHeaderList_t;	

Description

This is a doubly-linked list of name / value (string) pairs.

It is used for keeping header information in HTTP requests and responses.

See also

sceHTTPAddHeaderList(), sceHTTPFreeHeaderList(), sceHTTPNextHeader()

sceHTTPMimeFilter_t

MIME filter structure

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

Structure

typedef struct sceHTTPMimeFilter {	
struct sceHTTPMimeFilter **next*;	Pointer to multipart lower level filter
struct sceHTTPMimeFilter **prev*;	Pointer to multipart higher level filter
int *itype*;	Input type
int *idesc*;	Input file descriptor during file input
unsigned char **ibuf*;	Input buffer
unsigned int *ibuflen*;	Input buffer length
unsigned char **iptr*;	Input pointer
int *idesc_eof*;	End-of-input-file flag
int *otype*;	Output type
int *odesc*;	Output file descriptor during file output
unsigned char **obuf*;	Output buffer
unsigned int *obuflen*;	Output buffer length
unsigned char **optr*;	Output point
sceHTTPHeaderList_t **headers*;	Header list
int *dflags*;	Decoding flag
int (**decoder*)**(const char *, char *, int);**	Decoding function pointer
unsigned char **dbuf*;	Decoding buffer
int *multipart*;	Multipart flag
char **boundary*;	Boundary string
} sceHTTPMimeFilter_t;	

Description

This structure is used for MIME processing. The user cannot directly access the members of this structure.

See also

sceHTTPMimeFilterCreate(), sceHTTPMimeFilterFree(), sceHTTPMimeFilterParseHeaders(), sceHTTPMimeFilterApply(), sceHTTPMimeFilterGetMultipartType(), sceHTTPMimeFilterChangeOutput(), sceHTTPMimeFilterGetStringOutput(), sceHTTPMimeFilterGetHeaderList()

sceHTTPParsedURI_t

Parsed URI structure

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

Structure

typedef struct sceHTTPParsedURI {	
char **scheme*;	URI protocol scheme name ("http")
char **username*;	URI user name
char **password*;	URI password
char **hostname*;	URI hostname
int *port*;	URI port number
char **filename*;	URI file pathname
char **search*;	URI search part
} sceHTTPParsedURI_t;	

Description

This structure is used to keep a parsed URI.

See also

secsceHTTPParseURI(), sceHTTPFreeURI(), sceHTTPCloneURI(), sceHTTPUnparseURI()

sceHTTPResponse_t

HTTP response structure

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

Structure

typedef struct sceHTTPResponse {	
int *http_ver***;**	Server HTTP version
int *http_rev***;**	Server HTTP revision
sceHTTPStatusCode_t *code***;**	Server response status code (integer type)
char **reason***;**	Server response result phrase
int *server_prot***;**	Server protocol
sceHTTPHeaderList_t **headers***;**	Server response header list
unsigned char **entity***;**	Server response data
unsigned int *length***;**	Server response data length (bytes)
int *interrupted***;**	Transaction interruption flag
int *date***;**	Time server responded
} sceHTTPResponse_t;	

Description

This structure keeps the HTTP response from the server.

The *code* parameter indicates the code specified in RFC2616.

The following value is defined as a constant for *server_prot*.

sceHTTPProt_HTTP	0

The *date* parameter is expressed as elapsed seconds since January 1, 1970 in GMT.

See also

sceHTTPGetResponse(), sceHTTPCleanUpResponse(), sceHTTPErrorString(), sceHTTPHeaderList_t

Functions

sceBASE64Encoder

Perform BASE64 encoding

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

Syntax

int sceBASE64Encoder(	
unsigned const char **in*,	Pointer to input byte string
unsigned char **out*,	Pointer to output byte string
int *ilen***);**	Input length

Calling conditions

Can be called from a thread

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

Description

This function performs BASE64 encoding of the input byte string specified by *in* and *ilen*, and outputs the result to the memory area specified by *out*.

The size of the output memory area must be at least (*ilen* + 2)/3*4.

Return value

Output byte count

sceBASE64LineDecoder

Decode BASE64 line

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

Syntax

int sceBASE64LineDecoder(

unsigned const char **in*,	Pointer to input byte string
unsigned char **out*,	Pointer to output byte string
int *ilen***);**	Input length

Calling conditions

Can be called from a thread

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

Description

This function decodes the input byte string specified by *in* and *ilen* (one line of data that was BASE64 encoded) and outputs the result to the memory area specified by *out*. The input byte string must have a length of 76 or less, not including the RFC822 newline (consecutive CR and LF) at the end. If a larger value is set for *ilen*, it is ignored.

The size of the output memory area must be at least 60 bytes.

Return value

Output byte count	Normal termination
-1	Error occurred

sceHTTPAbortRequest

Abort HTTP request

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

Syntax

int sceHTTPAbortRequest(	
sceHTTPClient_t ****client*);**	Pointer to HTTP client object

Calling conditions

Can be called from a thread

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

Description

This function aborts an HTTP transaction request.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t

sceHTTPAddCookieList

Add cookie list

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

Syntax

sceHTTPCookieList_t *sceHTTPAddCookieList(	
sceHTTPCookieList_t ****p*,**	Cookie list
sceHTTPCookie_t ****cp*);**	Pointer to a cookie

Calling conditions

Can be called from a thread

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

Description

This function adds a new element to a cookie list.

The cookie given by *cp* and its element is duplicated and added.

Return value

Header list after addition	Normal termination
Null	Error occurred

See also

sceHTTPCookieList_t, sceHTTPCookie_t

sceHTTPAddHeaderList

Add header list

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

Syntax

sceHTTPHeaderList_t *sceHTTPAddHeaderList(	
**sceHTTPHeaderList_t ****p*	Header list
**const char ****name*	Name (attribute) part of header to be added
const char ****value*);**	Value part of header to be added

Calling conditions

Can be called from a thread

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

Description

This function adds a new element to the end of the header list specified by *p*. The element, which is represented by a pair of strings indicating a *name* (attribute) and *value*, corresponds to the name:value format of an HTTP header.

Return value

Header list after addition	Normal termination
NULL	Error occurred

See also

sceHTTPHeaderList_t

sceHTTPCleanUpResponse

Return to initial state of HTTP response

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

Syntax

int sceHTTPCleanUpResponse(	
sceHTTPClient_t ****client*);**	Pointer to HTTP client object

Calling conditions

Can be called from a thread

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

Description

This function returns to its initial state, the structure used to hold an HTTP response.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t

sceHTTPCloneURI

Clone parsed URI

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

Syntax

sceHTTPParsedURI_t *sceHTTPCloneURI(	
sceHTTPParsedURI_t ****puri*);**	Pointer to parsed URI structure to be cloned

Calling conditions

Can be called from a thread

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

Description

This function creates a clone of **puri*. The members are also cloned.

Return value

When processing completes normally, a pointer to the parsed URI structure that was cloned is returned. When an error occurs, NULL is returned.

See also

sceHTTPParsedURI_t

sceHTTPClose

Close HTTP connection

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

Syntax

int sceHTTPClose(	
sceHTTPClient_t ****client*);**	Pointer to HTTP client object to be closed

Calling conditions

Can be called from a thread

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

Description

This function closes the HTTP connection with the server.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t

sceHTTPCreate

Create HTTP client object

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

Syntax

sceHTTPClient_t *sceHTTPCreate(void);

Calling conditions

Can be called from a thread

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

Description

This function gets a structure for performing new HTTP client transactions and returns a pointer to that structure.

Return value

When processing completes normally, a pointer (non-zero value) to the HTTP client that was created is returned. When an error occurs, NULL is returned.

See also

sceHTTPClient_t

sceHTTPDestroy

Free HTTP client object

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

Syntax

int sceHTTPDestroy(	
sceHTTPClient_t ****client*);**	Pointer to HTTP client object to be freed

Calling conditions

Can be called from a thread

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

Description

This function frees the structure used to perform HTTP client transactions.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t

sceHTTPErrorString

Create HTTP response status string

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

Syntax

const char *sceHTTPErrorString(	
sceHTTPStatusCode_t *error***);**	HTTP response code

Calling conditions

Can be called from a thread

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

Description

This function returns a string explaining the HTTP response code as defined in RFC2616. The string is statically allocated.

Return value

When processing completes normally, an explanatory string corresponding to *error* is returned. When an error occurs, NULL is returned.

See also

sceHTTPStatusCode_t

sceHTTPFindAbsoluteURI

Make absolute URI

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

Syntax

char *sceHTTPFindAbsoluteURI(	
**const char ****uri*	URI string
const char ****base*);**	Base URI string

Calling conditions

Can be called from a thread

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

Description

This function makes **uri* into an absolute URI based on **base*.

Return value

When processing completes normally, the absolute URI string is returned. When an error occurs, NULL is returned.

sceHTTPFreeAuthList

Free authentication challenge list

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

Syntax

int sceHTTPFreeAuthList(	
sceHTTPAuthList_t ***p*****);**	Authentication challenge list

Calling conditions

Can be called from a thread

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

Description

This function frees all of the elements of the authentication challenge list specified by *p*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPAuthList_t

sceHTTPFreeCookieList

Free cookie list

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

Syntax

int sceHTTPFreeCookieList(	
sceHTTPCookieList_t ****p*);**	Cookie list

Calling conditions

Can be called from a thread

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

Description

This function frees all of the elements of the cookie list specified by *p*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPCookieList_t

sceHTTPFreeHeaderList

Free header list

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

Syntax

int sceHTTPFreeHeaderList(	
sceHTTPHeaderList_t ****p*);**	Header list

Calling conditions

Can be called from a thread

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

Description

This function frees all of the elements of the header list specified by *p*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPHeaderList_t

sceHTTPFreeLocations

Free redirection location array

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

Syntax

int sceHTTPFreeLocations(	
char *****locations*);**	Pointer to location array

Calling conditions

Can be called from a thread

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

Description

This function frees both the array of URI strings obtained by sceHTTPParseLocations() as well as the array elements.

Return value

0	Normal termination
-1	Error

See also

sceHTTPParseLocations()

sceHTTPFreeURI

Free parsed URI

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

Syntax

int sceHTTPFreeURI(	
sceHTTPParsedURI_t ****puri*);**	Pointer to parsed URI structure to be freed

Calling conditions

Can be called from a thread

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

Description

This function frees the parsed URI structure specified by *puri*.

Return value

0	Normal termination
-1	Error

See also

sceHTTPParsedURI_t

sceHTTPGetClientError

Get HTTP client internal error code

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

Syntax

int sceHTTPGetClientError(

sceHTTPClient_t ***client*****);**	Pointer to HTTP client object

Calling conditions

Can be called from a thread

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

Description

This function returns a code indicating the reason when an internal error such as an insufficient memory condition occurs. The internal error codes are currently defined as follows.

Table 2-1

Macro definition	Value	Meaning
sceHTTPError_KERNEL	-1001	Kernel call failed
sceHTTPError_NOMEM	-1002	Insufficient memory
sceHTTPError_IO	-1003	IO failed
sceHTTPError_INVAL	-1004	Invalid numeric value detected
sceHTTPError_TIMEOUT	-1005	Timeout
sceHTTPError_RESOLV	-1006	Host name resolution failed
sceHTTPError_SOCKET	-1007	Socket acquisition failed
sceHTTPError_CONNECT	-1008	Connection failed
sceHTTPError_SSL	-1009	SSL error
sceHTTPError_NOTYET	-1010	Non-existent function called
sceHTTPError_INTR	-1011	Interrupted

Return value

Internal error code of HTTP client specified by *client*

See also

sceHTTPClient_t

sceHTTPGetOption

Get HTTP option

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

Syntax

int sceHTTPGetOption(	
sceHTTPClient_t **client*,	Pointer to HTTP client object
sceHTTPOption_t *opt*,	Option number
...);	Takes a number of additional arguments depending on the option number

Calling conditions

Can be called from a thread

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

Description

This function gets the settings of various options for HTTP transactions. For *opt*, specify an sceHTTPOption_t type enum value representing the option to be obtained, and specify appropriate additional arguments for receiving the settings. For example, to get request data and its byte length, specify sceHTTPO_RequestEntity for *opt*. Since this option takes two additional arguments, call this function with the following form.

```
char *data;
int length;

sceHTTPGetOption(client, sceHTTPO_RequestEntity, &data, &length);
```

The values that can be specified for *opt* and the corresponding additional arguments are as follows.

Get user agent name

opt	sceHTTPO_ClientName
char ***namep*	Pointer to variable for storing the user agent name that was obtained

Get HTTP revision

opt	sceHTTPO_HTTPRevision
int ***revisionp*	Pointer to integer variable for storing the HTTP revision that was obtained

Get HTTP method

opt	sceHTTPO_Method
sceHTTPMethod_t **mtdp*	Pointer to variable for storing the HTTP method constant that was obtained. For details about HTTP method constants, see the description of sceHTTPMethod_t.

Get parsed URI

opt	sceHTTPO_ParsedURI
sceHTTPParsedURI_t ***urip*	Pointer to variable for storing the parsed URI that was obtained

Get parsed proxy URI

opt	sceHTTPO_ProxyURI
sceHTTPParsedURI_t ***pxyp*	Pointer to variable for storing the parsed proxy URI that was obtained

Get request header list

opt	sceHTTPO_RequestHeaders
sceHTTPHeaderList_t ***hdp*	Pointer to variable for storing the request header list that was obtained

Get request data and its byte length

opt	sceHTTPO_RequestEntity
char ***datap*	Pointer to variable for storing the request data that was obtained
unsigned int **lengthp*	Pointer to integer variable for storing the size of the request data that was obtained

Get response header acquisition timeout value

opt	sceHTTPO_ResponseTimeout
int **timoutp*	Pointer to integer variable for storing the timeout value (seconds) that was obtained

Get response data acquisition timeout value

opt	sceHTTPO_TransferTimeout
int **timoutp*	Pointer to integer variable for storing the timeout value (seconds) that was obtained

Get blocking mode

opt	sceHTTPO_BlockingMode
int **blkmodep*	Pointer to integer variable for storing the blocking mode that was obtained

Get callback function called when transaction completes

opt	sceHTTPO_EndOfTransactionCB
void ***funcp*	Pointer to variable for storing pointer to function that was obtained

Get callback function called when chunk is received

opt	sceHTTPO_ReceiveChunkCB
void ***funcp*	Pointer to variable for storing pointer to function that was obtained

Get stack size and priority of non-blocking-mode transaction execution thread

opt	sceHTTPO_ThreadValue
int **stacksize*	Pointer to variable for storing stack size that was obtained
int **priority*	Pointer to variable for storing priority that was obtained

Get connection hold parameters

opt	sceHTTPO_KeepAlive
int **timeout*	Pointer to variable for storing connection hold time (seconds) that was obtained
int **maxcount*	Pointer to variable for storing maximum connection hold count that was obtained

int *priority*	Priority (default is 63)

Notes

The sceHTTPO_KeepAlive option value is used only by sceHTTPGetOption(). When the connection hold time and maximum connection hold count are explicitly indicated during a response but their values are not explicitly indicated, this option gets their default values according to the protocol. When 0 is returned, it indicates that no connection is being held, and when -1 is returned, it indicates that a connection is held but the time or count is uncertain. This option is provided only for compatibility with some clients and servers that support HTTP/1.0.

Return value

0	Normal termination
-1	Error

See also

sceHTTPClient_t, sceHTTPOption_t, sceHTTPMethod_t, sceHTTPURI_t, sceHTTPHeaderList_t

sceHTTPGetResponse

Get HTTP response

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

Syntax

sceHTTPResponse_t *sceHTTPGetResponse(	
sceHTTPClient_t ***client*);**	Pointer to HTTP client object

Calling conditions

Can be called from a thread

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

Description

This function returns a pointer to the structure that holds the HTTP response.

Return value

Pointer to obtained HTTP response structure

See also

sceHTTPResponse_t, sceHTTPClient_t

sceHTTPGetSocketError

Get socket error code

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

Syntax

int sceHTTPGetSocketError(	
sceHTTPClient_t ***client*****);**	Pointer to HTTP client object

Calling conditions

Can be called from a thread

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

Description

This function gets the detailed error code that is returned by the socket layer when processing by sceHTTPOpen() for getting the socket or connecting fails.

Notes

sceHTTPOpen() simply returns -1 when an error occurs. However, you can determine whether processing failed while getting the socket or connecting according to the value returned by sceHTTPGetClientError().

Return value

Error code returned by socket layer

See also

sceHTTPClient_t, sceHTTPGetClientError()

sceHTTPInit

Initialize library (for HTTP)

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

Syntax

int sceHTTPInit(void);

Calling conditions

Can be called from a thread

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

Description

This function initializes libhttp. It must be called before using other libhttp functions.

Return value

0	Normal termination
-1	Abnormal termination

sceHTTPIsAbsoluteURI

Check for absolute URI

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

Syntax

int sceHTTPIsAbsoluteURI(	
const char ****uri*);**	URI string

Calling conditions

Can be called from a thread

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

Description

This function checks whether or not **uri* is an absolute URI. **uri* must be a URI with a valid format.

Return value

1	Absolute URI
0	Relative URI

sceHTTPMimeFilterApply

Perform MIME filter processing

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

Syntax

int sceHTTPMimeFilterApply(	
sceHTTPMimeFilter_t ****p*,**	Pointer to MIME filter
int ****closep*);**	Pointer to variable for setting part termination state

Calling conditions

Can be called from a thread

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

Description

This function processes a MIME part entity and advances to the next part, if one exists. If the processed part was the last part of a multipart entity and the *closep* argument is not 0, this function returns a non-zero value for **closep*.

Consequently, if *p* encounters a part termination in a low-level filter, *p* is freed within this function. Once freed, *p* cannot be subsequently referenced.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPMimeFilter_t

sceHTTPMimeFilterChangeOutput

Change MIME filter output destination

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

Syntax

int sceHTTPMimeFilterChangeOutput(	
**sceHTTPMimeFilter_t ****p*	Pointer to MIME filter
int *otype***,**	Output type
void ****oarg*);**	Pointer indicating output

Calling conditions

Can be called from a thread

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

Description

This function changes the output destination of the MIME filter specified by *p* so that it is the destination specified by *otype* and *oarg*.

For information about how to specify *otype* and *oarg*, see the description of sceHTTPMimeFilterCreate().

If the previously specified output destination was memory, the memory area of the output destination is automatically freed since it was allocated internally by the library. If the previous output destination was a file, the file is not automatically closed.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPMimeFilter_t, sceHTTPMimeFilterCreate()

sceHTTPMimeFilterCreate

Create MIME filter

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

Syntax

sceHTTPMimeFilter_t *sceHTTPMimeFilterCreate(

int *itype***,**	Input type
void ****iarg*,**	Pointer representing input
int *ilen***,**	Input length
int *otype***,**	Output type
void ****oarg*);**	Pointer representing output

Calling conditions

Can be called from a thread

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

Description

This function specifies the input and output for a MIME filter and performs processing to create the filter.

The following two input/output types can be specified for *itype* and *otype*.

Table 2-2

Macro Definition	Input/Output Destination
sceHTTPMimeFilter_FILE	File
sceHTTPMimeFilter_STRING	Memory

When a file is specified (*itype*/*otype* is sceHTTPMimeFilter_FILE), *iarg*/*oarg* is set to the file descriptor number cast to (void *). When input is from a file (*itype* is sceHTTPMimeFilter_FILE), the *ilen* argument is ignored.

When input from memory is specified (*itype* is sceHTTPMimeFilter_STRING), the byte string in memory, which is to be the input, is specified for *iarg* and its length is specified for *ilen*.

When output to memory is specified (*otype* is sceHTTPMimeFilter_STRING), the output destination memory area will be automatically allocated by the library, so the result and its length can be obtained with sceHTTPMimeFilterGetStringOutput(). (The output result byte string is not zero-terminated.)

Return value

When processing completes normally, a pointer to the MIME filter that was created is returned. When an error occurs, NULL is returned.

See also

sceHTTPMimeFilter_t, sceHTTPMimeFilterGetStringOutput()

sceHTTPMimeFilterFree

Free MIME filter

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

Syntax

int sceHTTPMimeFilterFree(	
sceHTTPMimeFilter_t ****p*);**	Pointer to MIME filter

Calling conditions

Can be called from a thread

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

Description

This function frees the MIME filter specified by *p*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPMimeFilter_t

sceHTTPMimeFilterGetHeaderList

Get MIME headers

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

Syntax

sceHTTPHeaderList_t *sceHTTPMimeFilterGetHeaderList(	
sceHTTPMimeFilter_t ***_p_);**	Pointer to MIME filter

Calling conditions

Can be called from a thread

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

Description

This function returns the MIME part headers that were parsed by sceHTTPMimeFilterParseHeader as a header list.

Return value

Pointer to header list	Normal termination
NULL	Error occurred

See also

sceHTTPHeaderList_t, sceHTTPMimeFilter_t, sceHTTPMimeFilterParseHeader()

sceHTTPMimeFilterGetMultipartType

Send MIME multipart type inquiry

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

Syntax

int sceHTTPMimeFilterGetMultipartType(	
sceHTTPMimeFilter_t ****p*);**	Pointer to MIME filter

Calling conditions

Can be called from a thread

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

Description

This function returns 0 if the part being processed is not multipart and returns its type if it is multipart.

The multipart type is determined by the Multipart/type string contained in the Content-Type header for that part. The following constants representing those strings have been defined.

sceHTTPMultipart_MIXED
sceHTTPMultipart_BYTERANGES
sceHTTPMultipart_ALTERNATIVE

Return value

0	Type is not multipart
Other	Integer value indicating multipart type

See also

sceHTTPMimeFilter_t

sceHTTPMimeFilterGetStringOutput

Get output and length for MIME memory

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

Syntax

int sceHTTPMimeFilterGetStringOutput(

sceHTTPMimeFilter_t **p*	Pointer to MIME filter
char ***odatap*,	Pointer to variable for storing starting address of output
int **olenp*);	Pointer to variable for storing byte count of output

Calling conditions

Can be called from a thread

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

Description

When the output destination of the MIME filter **p* is memory, this function returns the output's starting address and length in the variables specified by *odatap* and *olenp*.

Since the memory area returned in **odatap* (byte string in which MIME filter was output) is not freed by sceHTTPMimeFilterFree(), the user must free it with the free() function.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPMimeFilter_t, sceHTTPMimeFilterCreate()

sceHTTPMimeFilterParseHeaders

Parse MIME headers

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

Syntax

int sceHTTPMimeFilterParseHeaders(	
sceHTTPMimeFilter_t ****p*);**	Pointer to MIME filter

Calling conditions

Can be called from a thread

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

Description

This function uses the MIME filter specified by *p* to parse the MIME part headers for the current level. If a Content-Type header indicating a multipart entity is detected, this function internally generates a new MIME filter for processing each part as a lower-level filter of the current filter.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPMimeFilter_t

sceHTTPNextHeader

Get next element in header list

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

Syntax

sceHTTPHeaderList_t *sceHTTPNextHeader(	
sceHTTPHeaderList_t ****p*);**	Header list

Calling conditions

Can be called from a thread

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

Description

This function returns a pointer to the next element in the header list.

Return value

Pointer to next element	Normal termination
NULL	When there is no next element

See also

sceHTTPHeaderList_t

sceHTTPOpen

Open HTTP connection

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

Syntax

int sceHTTPOpen(	
sceHTTPClient_t ****client*);**	Pointer to HTTP client object to be opened

Calling conditions

Can be called from a thread

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

Description

This function establishes an HTTP connection with the server. The sceHTTPSetOption() function must be used to set the parsed URL of the server or proxy before this function is called.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t, sceHTTPSetOption()

sceHTTPParseAuth

Parse authentication challenge in response

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

Syntax

sceHTTPAuthList_t *sceHTTPParseAuth(	
sceHTTPResponse_t ***rp*****);**	Pointer to structure that represents an HTTP response

Calling conditions

Can be called from a thread

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

Description

This function parses all WWW-Authenticate headers within **rp* and returns them as an authentication challenge list.

Return value

Pointer to authentication challenge list	Normal termination
NULL	Error occurred

See also

sceHTTPAuthList_t, sceHTTPResponse_t

sceHTTPParseAuthInfo

Parse authentication verification information in response

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

Syntax

sceHTTPAuthInfo_t *sceHTTPParseAuthInfo(	
sceHTTPResponse_t ****rp*);**	Pointer to structure that represents an HTTP response

Calling conditions

Can be called from a thread

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

Description

This function parses Authentication-Info headers within **rp*, generates an authentication verification information structure, and returns a pointer to that structure.

Return value

Pointer to authentication verification information structure	Normal termination
NULL	Error occurred

See also

sceHTTPAuthInfo_t, sceHTTPResponse_t

sceHTTPParseCookie

Parse cookie in response

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

Syntax

sceHTTPCookieList_t *sceHTTPParseCookie(

sceHTTPResponse_t ***rp*);**	Pointer to structure that represents an HTTP response

Calling conditions

Can be called from a thread

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

Description

This function parses all Set-Cookie headers within **rp* and returns them as a cookie list.

Return value

Pointer to cookie list	Normal termination
NULL	Error occurred

See also

sceHTTPCookieList_t, sceHTTPResponse_t

sceHTTPParseLocations

Parse redirection locations in response

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

Syntax

const char **sceHTTPParseLocations(	
sceHTTPResponse_t ****rp*);**	Pointer to structure that represents HTTP response

Calling conditions

Can be called from a thread

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

Description

This function parses all Location headers within **rp* and returns them as an array of URI strings. The end of the array is indicated by a NULL element.

Return value

URI string array	Normal termination
NULL	Error occurred

See also

sceHTTPResponse_t

sceHTTPParseURI

Parse URI

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

Syntax

sceHTTPParsedURI_t *sceHTTPParseURI(	
const char ****uri*,**	String representing URI
int *pflag***);**	Parse option flag

Calling conditions

Can be called from a thread

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

Description

This function parses **uri* and divides it into a scheme, host name, port number, file path, search part, and other components, stores these in a structure that represents the parsed URI, and returns a pointer to this structure.

This structure must be freed with sceHTTPFreeURI().

pflag is a flag representing options for parsing. The following constants are currently defined as options. A bitwise logical OR can be used as necessary to specify multiple options.

Table 2-3

Macro Definition	Meaning
sceHTTPParseURI_FILENAME	Also parse file path
sceHTTPParseURI_SEARCHPART	Also parse search part

The scheme, host name, and port number are always parsed. If **uri* does not contain a port number, the port number is assumed to be 80. The constant sceHTTPProt_HTTP is defined as a constant representing the scheme.

Return value

When processing completes normally, a pointer to the parsed URI structure is returned. When an error occurs, NULL is returned.

See also

sceHTTPParsedURI_t

sceHTTPRequest

Execute HTTP transaction

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

Syntax

int sceHTTPRequest(	
sceHTTPClient_t ***_client_);**	Pointer to HTTP client object

Calling conditions

Can be called from a thread

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

Description

This function executes an HTTP transaction with the server.

Before this function is called, an HTTP connection with the server or proxy must have been established using sceHTTPOpen().

This function can be executed in blocking mode or non-blocking mode according to the option setting. (The default is blocking mode. See sceHTTPSetOption().) In blocking mode, the function does not return until the HTTP transaction ends. In non-blocking mode, the function returns immediately, and a callback function that was set by the user is invoked when the transaction ends.

Notes

The callback function mentioned above has an integer argument. When the transaction completes normally, the argument is set to 0, and when an error occurs, the argument is set to -1.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t, sceHTTPSetOption()

sceHTTPSetBasicAuth

Set basic authentication

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

Syntax

int sceHTTPSetBasicAuth(	
sceHTTPClient_t **client*,	Pointer to structure for performing HTTP transactions
const char **user*,	Pointer to user name
const char **passwd***);**	Pointer to password

Calling conditions

Can be called from a thread

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

Description

This function converts the user name and password specified by *user* and *passwd* to a basic-type Authorization header and adds it to the request header list of **client*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t

sceHTTPSetCookie

Set cookie

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

Syntax

int sceHTTPSetCookie(	
sceHTTPClient_t **client*,	Pointer to HTTP client structure
sceHTTPCookieList_t **p*);	Pointer to cookie list

Calling conditions

Can be called from a thread

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

Description

This function converts the cookie list given by *p* to Cookie headers and adds them to the request header list of **client*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t, sceHTTPCookieList_t

sceHTTPSetDigestAuth

Set digest authentication

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

Syntax

int sceHTTPSetDigestAuth(	
sceHTTPClient_t ****client*,**	Pointer to structure for performing HTTP transactions
sceHTTPDigest_t ****digest*);**	Pointer to digest request structure

Calling conditions

Can be called from a thread

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

Description

This function converts **digest* to a digest-type Authorization header and adds it to the request header list of **client*.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t, sceHTTPDigest_t

sceHTTPSetOption

Set HTTP option

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

Syntax

int sceHTTPSetOption(

sceHTTPClient_t **client***,**	Pointer to HTTP client object
sceHTTPOption_t *opt***,**	Option number
*...***);**	Takes a number of additional arguments depending on the option number

Calling conditions

Can be called from a thread

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

Description

This function sets various options for HTTP transactions.

For *opt*, specify an sceHTTPOption_t type enum value representing the option to be set, and specify the settings by using the additional arguments that correspond to each option.

For example, to set request data and the data length, specify sceHTTPO_RequestEntity for *opt*. Since this option takes three additional arguments, call this function with the following form.

```
char *data;
int length;

sceHTTPSetOption(client, sceHTTPO_RequestEntity, data, length, 0);
```

The values that can be specified for *opt* and the corresponding additional arguments are as follows.

Set user agent name

opt	sceHTTPO_ClientName
char **name*	Pointer to user agent name. The default value is "unknown (sceHTTPLib-X.X.X)," where the version number is entered for X.X.X. Since this is only a sample value, be sure to change it to an appropriate name when it is used in a title.

Set HTTP revision

opt	sceHTTPO_HTTPRevision
int *revision*	HTTP revision. Specify 0 or 1. The default is 0.

Set HTTP method

opt	sceHTTPO_Method
sceHTTPMethod_t *method*	HTTP method constant. For details, see the description of sceHTTPMethod_t.

Set parsed URI

opt	sceHTTPO_ParsedURI

sceHTTPParsedURI_t **uri*	Pointer to parsed URI

Set parsed proxy URI

opt	sceHTTPO_ProxyURI
sceHTTPParsedURI_t **proxy*	Pointer to parsed proxy URI

Set (add to) request header list

opt	sceHTTPO_RequestHeaders
sceHTTPHeaderList_t **hd*	Pointer to request header list
int *overwrite*	Whether or not to overwrite 0: Append 1: Overwrite (the old header list is deleted and freed)

Set request data and its byte length

opt	sceHTTPO_RequestEntity
char **data*	Pointer to request data
unsigned int *length*	Request data length
int *flags*	Flag

The *flags* argument has the following bit definitions. These two flags cannot be set at the same time.

Table 2-4

Macro Definition	Meaning
sceHTTPInputF_ESCAPE	The given request data is set after URL encoding. The encoded length is also set as the data length.
sceHTTPInputF_LINK	The given request data is used by linking it as is, and is not duplicated.

Set response header acquisition timeout value

opt	sceHTTPO_ResponseTimeout
int *timout*	Timeout value (seconds). The default is no timeout.

Set response data acquisition timeout value

opt	sceHTTPO_TransferTimeout
int *timout*	Timeout value (seconds). The default is no timeout.

Set blocking mode

opt	sceHTTPO_BlockingMode
int *blkmode*	Blocking mode 0: Non-blocking mode 1: Blocking mode (default)

Set callback function called when transaction completes

opt	sceHTTPO_EndOfTransactionCB
void (**func*)(int *flags*)	Pointer to callback function called when transaction completes. The *flags* argument for the callback function is 0 when the

	transaction completes normally and -1 when an error has occurred.

Set callback function called when chunk is received

opt	sceHTTPO_ReceiveChunkCB
void (**func*)(sceHTTPClient_t **client*, char **cdata*, int *clen*)	Pointer to callback function called when chunk is received

Set stack size and priority of non-blocking-mode transaction execution thread

opt	sceHTTPO_ThreadValue
int *stacksize*	Stack size (default is 8192)
int *priority*	Priority (default is 63)

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t, sceHTTPOption_t, sceHTTPMethod_t, sceHTTPURI_t, sceHTTPHeaderList_t

sceHTTPSetRedirection

Set redirection

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

Syntax

int sceHTTPSetRedirection(

sceHTTPClient_t **client*,	Pointer to structure for performing HTTP transactions
sceHTTPParsedURI_t **uri*,	Parsed URI of redirection destination
int *proxy***);**	Flag indicating proxy redirection

Calling conditions

Can be called from a thread

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

Description

This function performs processing required for performing HTTP transactions using redirection.

Return value

0	Normal termination
-1	Error occurred

See also

sceHTTPClient_t, sceHTTPParsedURI_t

sceHTTPUnparseURI

Create URI string from parsed URI

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

Syntax

char *sceHTTPUnparseURI(	
sceHTTPParsedURI_t ***puri*);**	Pointer to parsed URI structure

Calling conditions

Can be called from a thread

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

Description

This function creates a URI string representing the contents of **puri*.

Return value

URI string	Normal termination
NULL	Error occurred

See also

sceHTTPParsedURI_t

sceQPrintableEncoder

Perform quoted-printable encoding

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

Syntax

int sceQPrintableEncoder(	
unsigned const char **in*,	Pointer to input byte string
unsigned char **out*,	Pointer to output byte string
int *ilen***);**	Input length

Calling conditions

Can be called from a thread

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

Description

This function performs QPrintable encoding of the input byte string that was specified by *in* and *ilen*, and outputs the result to the memory area specified by *out*.

The size of the output memory area must be at least (*ilen* * 3 + ilen / 38 + 2).

Return value

Output byte count

sceQPrintableLineDecoder

Decode quoted-printable line

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

Syntax

int sceQPrintableLineDecoder(	
**unsigned const char ****in*,	Pointer to input byte string
**unsigned char ****out*,	Pointer to output byte string
int *ilen***);**	Input length

Calling conditions

Can be called from a thread

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

Description

This function decodes the input byte string that was specified by *in* and *ilen* (one line of quoted-printable encoded data) and outputs the result to the output memory area specified by *out*. The input byte string must have a length of 78 or less, including the terminating RFC822 newline (consecutive CR and LF). If a larger value is set for *ilen*, it is ignored. The size of the output memory area must be at least 78 bytes.

Return value

Output byte count	Normal termination
-1	Error occurred

sceURLEscape

Perform URL escape processing

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

Syntax

char *sceURLEscape(	
unsigned const char ****in*,);**	Pointer to string

Calling conditions

Can be called from a thread

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

Description

This function performs URL escape processing on the input string specified by *in* and returns the result as a new string.

Return value

Pointer to string	Normal termination
NULL	Error occurred

sceURLUnescape

Perform URL unescape processing

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

Syntax

char *sceURLUnescape(	
unsigned const char ****in*);**	Pointer to string

Calling conditions

Can be called from a thread

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

Description

This function performs URL unescape processing on the input string specified by *in* and returns the result as a new string.

Return value

Pointer to string	Normal termination
0	Error occurred

Global Variables

sceHTTPLibVersion

Library version

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

Syntax

const char *sceHTTPLibVersion;

Description

This variable maintains the libhttp version string, which is statically allocated.

The version string has a format consisting of three decimal numbers separated by dots such as "1.1.0".

Constant Definitions

sceHTTPMethod_t

HTTP method definition

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

Definition

```
typedef enum {
    sceHTTPM_OPTIONS,
    sceHTTPM_GET,
    sceHTTPM_HEAD,
    sceHTTPM_POST,
    sceHTTPM_PUT,
    sceHTTPM_DELETE,
    sceHTTPM_TRACE,
    sceHTTPM_CONNECT
} sceHTTPMethod_t;
```

Description

These constants represent HTTP 1.1 commands.

libhttp currently supports sceHTTPM_GET, sceHTTPM_HEAD, and sceHTTPM_POST.

See also

sceHTTPClient_t, sceHTTPSetOption(), sceHTTPGetOption()

sceHTTPOption_t

HTTP option definition

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

Definition

```
typedef enum {
   sceHTTPO_ClientName,
   sceHTTPO_HTTPRevision,
   sceHTTPO_Method,
   sceHTTPO_ParsedURI,
   sceHTTPO_ProxyURI,
   sceHTTPO_RequestHeaders,
   sceHTTPO_RequestEntity,
   sceHTTPO_ResponseTimeout,
   sceHTTPO_TransferTimeout,
   sceHTTPO_BlockingMode,
   sceHTTPO_EndOfTransactionCB,
   sceHTTPO_ReceiveChunkCB,
   sceHTTPO_ThreadValue,
   sceHTTPO_KeepAlive,
   sceHTTPO_SSLFlags,
} sceHTTPOption_t;
```

Description

These constants represent options that are used by secHTTPGetOption() and secHTTPSetOption().

See also

sceHTTPSetOption(), sceHTTPGetOption()

sceHTTPStatusCode_t

HTTP 1.1 response status

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

Definition

```
typedef enum {
    sceHTTPC_Continue                          = 100,
    sceHTTPC_SwitchProtocols                   = 101,
    sceHTTPC_OK                                = 200,
    sceHTTPC_Created                           = 201,
    sceHTTPC_Accepted                          = 202,
    sceHTTPC_NonAuthoritativeInfo              = 203,
    sceHTTPC_NoContent                         = 204,
    sceHTTPC_ResetContent                      = 205,
    sceHTTPC_PartialContent                    = 206,
    sceHTTPC_MultipleChoices                   = 300,
    sceHTTPC_MovedPermanently                  = 301,
    sceHTTPC_Found                             = 302,
    sceHTTPC_SeeOther                          = 303,
    sceHTTPC_NotModified                       = 304,
    sceHTTPC_UseProxy                          = 305,
    sceHTTPC_TemporaryRedirect                 = 307,
    sceHTTPC_BadRequest                        = 400,
    sceHTTPC_Unauthorized                      = 401,
    sceHTTPC_PaymentRequired                   = 402,
    sceHTTPC_Forbidden                         = 403,
    sceHTTPC_NotFound                          = 404,
    sceHTTPC_MethodNotAllowed                  = 405,
    sceHTTPC_NotAcceptable                     = 406,
    sceHTTPC_ProxyAuthenticationRequired       = 407,
    sceHTTPC_RequestTimeout                    = 408,
    sceHTTPC_Conflict                          = 409,
    sceHTTPC_Gone                              = 410,
    sceHTTPC_LengthRequired                    = 411,
    sceHTTPC_PreconditionFailed                = 412,
    sceHTTPC_RequestEntityTooLarge             = 413,
    sceHTTPC_RequestURITooLarge                = 414,
    sceHTTPC_UnsupportedMediaType              = 415,
    sceHTTPC_RequestedRangeNotSatisfiable      = 416,
    sceHTTPC_ExceptionFailed                   = 417,
    sceHTTPC_InternalServerError               = 500,
    sceHTTPC_NotImplemented                    = 501,
    sceHTTPC_BadGateway                    = 502,
    sceHTTPC_ServiceUnavailable            = 503,
    sceHTTPC_GatewayTimeout                = 504,
    sceHTTPC_HTTPVersionNotSupported       = 505
} sceHTTPStatusCode_t;
```

Description

These constants represent HTTP 1.1 response status. The values are specified in RFC2616.

See also

sceHTTPResponse_t, sceHTTPErrorString()

Chapter 3: Network Socket Library
Table of Contents

Structures

sceInsockHostent_t

Internet host structure

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

Structure

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

Description

This structure represents a host on the Internet.

See also

sceInsockGethostbyaddr(), sceInsockGethostbyname()

sceInsockInAddr_t

IPv4 address structure

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

Structure

typedef struct sceInsockInAddr {

u_int *s_addr;*	IPv4 address (4 bytes)

} sceInsockInAddr_t;

#define in_addr sceInsockInAddr

Description

This structure is used for saving an IPv4 address.

See also

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

sceInsockSockaddr_t

Socket address structure

Library	Introduced	Documentation last modified
libinsck	2.3	July 2, 2001

Structure

typedef u_char sceInsockSaFamily_t;

typedef struct sceInsockSockaddr {

u_char *sa_len;*	Address structure size
sceInsockSaFamily_t *sa_family;*	Address family
char *sa_data***[14];**	Protocol-dependent address

} sceInsockSockaddr_t;

#define sa_family_t sceInsockSaFamily_t

#define sockaddr sceInsockSockaddr

Description

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

See also

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

sceInsockSockaddrIn_t

Internet socket address structure

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

Structure

typedef struct sceInsockSockaddrIn {	
u_char *sin_len;*	Address structure size (16 bytes)
u_char *sin_family;*	Address family (AF_INET only)
u_short *sin_port;*	TCP or UDP port number (network byte order)
sceInsockInAddr_t *sin_addr;*	IPv4 address
char *sin_zero***[8];**	Unused
} sceInsockSockaddrIn_t;	
#define sockaddr_in sceInsockSockaddrIn	

Description

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

See also

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

BSD Socket API-compatible Functions

sceInsockAccept

Get socket for which TCP connection was established

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

Syntax

#include < libinsck.h >	
typedef u_int sceInsockSocklen_t;	
int sceInsockAccept(	
int *s,*	Listening socket
	(sceInsockBind() and sceInsockListen() completed)
**sceInsockSockaddr_t ****addr,*	Pointer to area for storing connection destination address structure
sceInsockSocklen_t **paddrlen***)**	Pointer to area for storing size of addr
#define accept sceInsockAccept	
#define socklen_t sceInsockSocklen_t	

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

New client socket descriptor	Normal termination
-1	Error

See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t, sceInsockErrno

sceInsockBind

Bind address to socket

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

int sceInsockBind(

int *s,*	Descriptor of socket to which local address is to be bound
const sceInsockSockaddr_t **addr,*	Pointer to local address structure
sceInsockSocklen_t *addrlen***);**	Local address structure size (always 16 bytes)

#define bind sceInsockBind

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t, sceInsockErrno

sceInsockConnect

Connect to server

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

int sceInsockConnect(

int *s,*	Descriptor of socket to be used for connection
const sceInsockSockaddr_t **addr,*	Pointer to local address structure
sceInsockSocklen_t *addrlen***);**	Local address structure size (always 16 bytes)

#define connect sceInsockConnect

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

0	Normal termination
-1	Error

See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t, sceInsockErrno

sceInsockErrno

Get socket function error value

Library	*Introduced*	*Documentation last modified*
libinsck	2.3	December 3, 2001

Syntax

#include < libinsck.h >

int sceInsockErrno;

Calling conditions

Can be called from a thread

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

Description

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

Error values that can be referenced are defined in /usr/local/sce/ee/gcc/ee/include/sys/errno.h.

Return value

Error code

List of libinsck error codes.

Table 3-1

Error code	Value	Meaning
ENOMEM	12	Memory allocation for each thread or socket failed
EBADF	9	Invalid socket number was specified
EPFNOSUPPORT	96	family argument of socket() function is not AF_INET
EPROTOTYPE	107	type argument of socket() function is unsupported value
EINVAL	22	Argument is invalid (for example, value of addrlen for bind is invalid)
EADDRINUSE	112	bind() was called for local port that is in use
EAFNOSUPPORT	106	((struct sockaddr_in*)addr)>sin_family of bind() function is invalid
EOPNOTSUPP	95	Invalid call to socket (for example, sendto() to SOCK_STREAM)

List of conversions between libinsck error codes and INET error codes which libinsck obtains via libnet

Table 3-2

Error code	Value	INET error code
0	0	sceINETE_OK
ETIMEDOUT	116	sceINETE_TIMEOUT
ECONNABORTED	113	sceINETE_ABORT
EBUSY	16	sceINETE_BUSY
ENETDOWN	115	sceINETE_LINK_DOWN
ENOMEM	12	sceINETE_INSUFFICIENT_RESOURCES
EADDRNOTAVAIL	125	sceINETE_LOCAL_SOCKET_UNSPECIFIED sceINETE_FOREIGN_SOCKET_UNSPECIFIED
EISCONN	127	sceINETE_CONNECTION_ALREADY_EXISTS
ENOTCONN	128	sceINETE_CONNECTION_DOES_NOT_EXIST
ESHUTDOWN	110	sceINETE_CONNECTION_CLOSING
ECONNRESET	104	sceINETE_CONNECTION_RESET
ECONNREFUSED	111	sceINETE_CONNECTION_REFUSED
EINVAL	22	sceINETE_INVALID_ARGUMENT sceINETE_INVALID_CALL
EHOSTUNREACH	118	sceINETE_NO_ROUTE

See also

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

sceInsockGethostbyaddr

Get host structure from 32-bit IPv4 address

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

Syntax

#include < libinsck.h >

sceInsockHostent_t *sceInsockGethostbyaddr(

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

#define gethostbyaddr sceInsockGethostbyaddr

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

Pointer to Internet host structure	Normal termination
0	Error

See also

sceInsockHErrno

sceInsockGethostbyname

Get host structure from hostname

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

Syntax

#include < libinsck.h >	
sceInsockHostent_t *sceInsockGethostbyname(	
const char **name***);**	Internet host name
#define gethostbyname sceInsockGethostbyname	

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

Pointer to Internet host structure	Normal termination
0	Error

See also

sceInsockHErrno

sceInsockGetpeername

Get socket connection destination information

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

int sceInsockGetpeername(

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

#define getpeername sceInsockGetpeername

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t

sceInsockGetSockName

Get local information of socket

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

Syntax

#include < libinsck.h >

int sceInsockGetsockname(

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

#define getsockname sceInsockGetsockname

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t

sceInsockGetsockopt

Get socket option

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

int sceInsockGetsockopt(

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

#define getsockopt sceInsockGetsockopt

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

Table 3-3

Socket Option Level	Meaning
IPPROTO_TCP	TCP related

Table 3-4

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

Return value

0 Normal termination

-1 Error

See also

sceInsockSockaddr_t, sceInsockSockaddrIn_t

sceInsockHErrno

Get error value of host structure function

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

Syntax

#include < libinsck.h >

int sceInsockHErrno;

Calling conditions

Can be called from a thread

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

Description

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

Return value

Table 3-5

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

See also

sceInsockGethostbyaddr(), sceInsockGethostbyname()

sceInsockInetAddr

Get 32-bit address from dot-format IPv4 address

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

Syntax

#include < libinsck.h >

u_int sceInsockInetAddr(

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

#define inet_addr sceInsockInetAddr

Calling conditions

Can be called from a thread

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

Description

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

Return value

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

sceInsockInetAton

Get 32-bit address from dot-format IPv4 address

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

Syntax

#include < libinsck.h >	
int sceInsockInetAton(	
const char **cp,*	Pointer to dot-decimal IPv4 address string
sceInsockInAddr_t **addr***);**	Pointer to area for storing converted 32-bit IPv4 address value
#define inet_aton sceInsockInetAton	

Calling conditions

Can be called from a thread

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

Description

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

Return value

1	Normal termination
0	String is illegal

See also

sceInsockInAddr_t

sceInsockInetLnaof

Get local network address from IPv4 address

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

Syntax

#include < libinsck.h >	
u_int sceInsockInetLnaof(	
sceInsockInAddr_t ***in*****);**	32-bit IPv4 address value
#define inet_lnaof sceInsockInetLnaof	

Calling conditions

Can be called from a thread

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

Description

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

Return value

Local network address value

See also

sceInsockInAddr_t

sceInsockInetMakeaddr

Construct IPv4 address from network address

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

Syntax

#include < libinsck.h >	
sceInsockInAddr_t sceInsockInetMakeaddr(	
u_int *net,*	Network address part
u_int *host***);**	Local network address part
#define inet_makeaddr sceInsockInetMakeaddr	

Calling conditions

Can be called from a thread

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

Description

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

Return value

Combined IPv4 address value

See also

sceInsockInAddr_t

sceInsockInetNetof

Get network address from IPv4 address

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

Syntax

#include < libinsck.h >	
u_int sceInsockInetNetof(	
sceInsockInAddr_t *in***);**	32-bit IPv4 address value
#define inet_netof sceInsockInetNetof	

Calling conditions

Can be called from a thread

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

Description

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

Return value

Network address value

See also

sceInsockInAddr_t

sceInsockInetNetwork

Get 32-bit address from dot-format IPv4 address

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

Syntax

#include < libinsck.h >	
u_int sceInsockInetNetwork(	
const char ****cp*);**	Pointer to dot-decimal IPv4 address string
#define inet_network sceInsockInetNetwork	

Calling conditions

Can be called from a thread

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

Description

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

Return value

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

sceInsockInetNtoa

Get dot-format address from 32-bit IPv4 address

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

Pointer to dot-decimal IPv4 address string

See also

sceInsockInAddr_t

sceInsockListen

Accept TCP connection

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

Syntax

#include < libinsck.h >

int sceInsockListen(

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

#define listen sceInsockListen

Calling conditions

Can be called from a thread

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

Description

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

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

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

Return value

0	Normal termination
-1	Error

See also

sceInsockErrno

sceInsockRecv

Receive data

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

Syntax

#include < libinsck.h >

size_t sceInsockRecv(

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

#define recv sceInsockRecv

Calling conditions

Can be called from a thread

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

Description

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

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

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

Return value

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

See also

sceInsockErrno

sceInsockRecvfrom

Receive data (also get address structure of sending host)

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

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

#define recvfrom sceInsockRecvfrom

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

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

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

Return value

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

See also

sceInsockSockaddr_t, sceInsockErrno

sceInsockSend

Send data

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

Syntax

#include < libinsck.h >

size_t sceInsockSend(

int *s,*	Descriptor of socket that is to send data
const void **buf,*	Pointer to send data
size_t *len,*	Size of data to be sent (in bytes)
int *flags***);**	Not supported (must be set to 0)

#define send sceInsockSend

Calling conditions

Can be called from a thread

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

Description

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

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

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

Return value

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

See also

sceInsockErrno

sceInsockSendto

Send data (specify address structure of receiving host)

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

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

#define sendto sceInsockSendto

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

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

See also

sceInsockSockaddr_t, sceInsockErrno

sceInsockSetsockopt

Set socket option

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

Syntax

#include < libinsck.h >

typedef u_int sceInsockSocklen_t;

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

#define setsockopt sceInsockSetsockopt

#define socklen_t sceInsockSocklen_t

Calling conditions

Can be called from a thread

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

Description

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

Table 3-6

Socket Option Level	Meaning
IPPROTO_TCP	TCP related

Table 3-7

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

Return value

0	Normal termination
-1	Error

sceInsockShutdown

Close socket

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceInsockErrno

sceInsockSocket

Create socket

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

Syntax

#include < libinsck.h >

size_t sceInsockSocket(

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

#define socket sceInsockSocket

Calling conditions

Can be called from a thread

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

Description

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

Return value

Positive value	Descriptor of generated socket
-1	Error

See also

sceInsockErrno

Other Functions

sceInsockAbort

Abort processing

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

Syntax

#include < libinsck.h >

int sceInsockAbort(

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

sceInsockSetRecvTimeout

Set receive timeout

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

Syntax

#include < libinsck.h >

int sceInsockSetRecvTimeout(

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

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

0	Normal termination
-1	Error

sceInsockSetSendTimeout

Set send timeout

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

Syntax

#include < libinsck.h >

int sceInsockSetSendTimeout(

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

sceInsockSetSifMBindRpcValue

Set buffer size, stack size and priority

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

Syntax

#include < libinsck.h>

int sceInsockSetSifMBindRpcValue(

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

Calling conditions

Can be called from a thread

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

Description

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

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

Return value

0	Normal termination
-1	Error

sceInsockTerminate

Free memory area

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

Syntax

#include < libinsck.h >

int sceInsockTerminate(

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

Calling conditions

Can be called from a thread

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

Description

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

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

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

Return value

0	Normal termination
-1	Error

Chapter 4: General-Purpose Network Wrapper API (netglue) Table of Contents

Structures

sceNetGlueHostent_t

Internet host structure

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

Structure

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

Description

This structure represents a host on the Internet.

See also

sceNetGlueGethostbyaddr(), sceNetGlueGethostbyname()

sceNetGlueInAddr_t

IPv4 address structure

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

Structure

typedef struct sceNetGlueInAddr {	
u_int *s_addr*;	IPv4 address (4 bytes)
} sceNetGlueInAddr_t;	
#define in_addr sceNetGlueInAddr	

Description

This structure is used to keep an IPv4 address.

See also

sceNetGlueSockaddrIn_t, sceNetGlueInetAton(), sceNetGlueInetLnaof(), sceNetGlueInetNetof(), sceNetGlueInetNtoa()

sceNetGlueSockaddr_t

Socket address structure

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

Structure

typedef u_char sceNetGlueSaFamily_t;	
typedef struct sceNetGlueSockaddr {	
u_char *sa_len*;	Address structure size
sceNetGlueSaFamily_t *sa_family*;	Address family
char *sa_data*[14];	Protocol-dependent address
} sceNetGlueSockaddr_t;	
#define sa_family_t sceNetGlueSaFamily_t	
#define sockaddr sceNetGlueSockaddr	

Description

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

See also

sceNetGlueAccept(), sceNetGlueBind(), sceNetGlueConnect(), sceNetGlueGetpeername(), sceNetGlueGetsockname(), sceNetGlueRecvfrom(), sceNetGlueSendto()

sceNetGlueSockaddrIn_t

Internet socket address structure

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

Structure

typedef struct sceNetGlueSockaddrIn {	
u_char *sin_len***;**	Address structure size (16 bytes)
u_char *sin_family***;**	Address family (AF_INET only)
u_short *sin_port***;**	TCP or UDP port number (network byte order)
sceNetGlueInAddr_t *sin_addr***;**	IPv4 address
char *sin_zero***[8];**	Unused
} sceNetGlueSockaddrIn_t;	
#define sockaddr_in sceNetGlueSockaddrIn	

Description

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

See also

sceNetGlueConnect(), sceNetGlueGetpeername(), sceNetGlueGetsockname(), sceNetGlueRecvfrom(), sceNetGlueSendto()

Functions

__sceNetGlueErrnoLoc

Get error value for socket functions

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

Syntax

#include < netglue.h >

int *__sceNetGlueErrnoLoc(void);

#define sceNetGlueErrno (*__sceNetGlueErrnoLoc())

Calling Conditions

Can be called from a thread

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

Description

This function returns the error code for socket functions (sceNetGlueAccept(), sceNetGlueBind(), sceNetGlueConnect(), sceNetGlueListen(), sceNetGlueRecv(), sceNetGlueRecvfrom(), sceNetGlueSend(), sceNetGlueSendto(), sceNetGlueShutdown(), sceNetGlueSocket()).

One sceNetGlueErrno exists for each thread, and the sceNetGlueErrno corresponding to each thread is returned by this function internally within the netglue library.

Return value

Table 4-1

Error Code	Value	Meaning
ETIMEDOUT	60	Timeout occurred
ECONNABORTED	53	Aborted by sceNetGlueAbort
EBUSY	16	Library not available yet (initialization not completed, for example)
ENETDOWN	50	Interface is down
ENOMEM	12	Insufficient memory
EADDRNOTAVAIL	49	Invalid address was specified
EISCONN	56	Specified connection is already established
ENOTCONN	57	Specified connection does not exist
ECONNRESET	54	Connection was reset
ECONNREFUSED	61	Request to establish connection was refused
EINVAL	22	Invalid argument was specified
EHOSTUNREACH	51	Network unreachable
EBADF	9	Invalid descriptor was specified

Error Code	Value	Meaning
EPFNOSUPPORT	46	Unsupported protocol family was specified
EPROTOTYPE	41	Unsupported protocol type was specified
EADDRINUSE	48	Attempt was made to bind to bound port
EAFNOSUPPORT	47	Specified address family is a value that is unsupported by socket protocol family
EOPNOTSUPP	45	Invalid call for socket

See also

sceNetGlueAccept(), sceNetGlueBind(), sceNetGlueConnect(), sceNetGlueListen(), sceNetGlueRecv(), sceNetGlueRecvfrom(), sceNetGlueSend(), sceNetGlueSendto(), sceNetGlueShutdown(), sceNetGlueSocket()

__sceNetGlueHErrnoLoc

Get error value for host structure functions

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

Syntax

#include < netglue.h >

int *__sceNetGlueHErrnoLoc(void);

#define sceNetGlueHErrno (*__sceNetGlueHErrnoLoc())

Calling Conditions

Can be called from a thread

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

Description

This function returns the error code for host structure functions (sceNetGlueGethostbyaddr(), sceNetGlueGethostbyname()).

One sceNetGlueHErrno exists for each thread, and the sceNetGlueHErrno corresponding to each thread is returned by this function internally within the netglue library.

Return value

Table 4-2

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

See also

sceNetGlueGethostbyaddr(), sceNetGlueGethostbyname()

sceNetGlueAbort

Abort processing of specified socket

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

Syntax

#include < netglue.h >	
int sceNetGlueAbort(	
int *s***,**	Descriptor of socket for which processing is to be aborted
int *flags***);**	This argument is currently unused (set to 0)

Calling Conditions

Can be called from a thread

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

Description

This function immediately cancels blocking of all threads that are blocked by a netglue function for socket s. A thread for which blocking was canceled will return with an error with errno = ECONNABORTED.

Return value

0	Normal termination
-1	Error

See also

__sceNetGlueErrnoLoc()

sceNetGlueAccept

Get socket for which TCP connection was established

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueAccept(

int *s*,	Listening socket (sceNetGlueBind() and sceNetGlueListen() were already executed)
sceNetGlueSockaddr_t **addr*,	Pointer to area for storing connection destination address structure
sceNetGlueSocklen_t **paddrlen*)	Pointer to area for storing size of addr

#define accept sceNetGlueAccept

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

When the host is operating as a TCP server, this function gets the connection that was connected from the client and returns its socket descriptor. At the same time, the client's address structure is stored in the area pointed to by the *addr* argument, and the size of the structure (always 4 bytes) is stored in the area pointed to by *paddrlen*.

If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

New client socket descriptor	Normal termination
-1	Error

See also

sceNetGlueSockaddr_t, __sceNetGlueErrnoLoc()

sceNetGlueBind

Bind address to socket

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueBind(

int *s*,	Descriptor of socket to which local address is to be bound
sceNetGlueSockaddr_t **addr*,	Pointer to local address structure
sceNetGlueSocklen_t *addrlen*);	Local address structure size (always 16 bytes)

#define bind sceNetGlueBind

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceNetGlueSockaddr_t, __sceNetGlueErrnoLoc()

sceNetGlueConnect

Connect to server

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueConnect(

int *s***,**	Descriptor of socket to be used for connection
sceNetGlueSockaddr_t **addr***,**	Pointer to local address structure
sceNetGlueSocklen_t *addrlen***);**	Local address structure size (always 16 bytes)

#define connect sceNetGlueConnect

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

0	Normal termination
-1	Error

See also

sceNetGlueSockaddr_t, __sceNetGlueErrnoLoc()

sceNetGlueGethostbyaddr

Get host structure from 32-bit IPv4 address

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

Syntax

#include < netglue.h >	
sceNetGlueHostent_t *sceNetGlueGethostbyaddr(	
const char * ***addr*****,**	Pointer to 32-bit IPv4 address value
int *len***,**	Size of address structure (4 bytes)
int *type***);**	Address family (AF_INET only)
#define gethostbyaddr sceNetGlueGethostbyaddr	

Calling Conditions

Can be called from a thread

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

Description

This function gets the Internet host structure corresponding to the 32-bit IPv4 address that was specified by the argument and returns a pointer to it. len is always 4 bytes, and type is always AF_INET. If an error occurs, details of the error can be obtained with sceNetGlueHErrno.

Return value

Pointer to Internet host structure	Normal termination
NULL	Error

See also

__sceNetGlueHErrnoLOC(), sceNetGlueHostent_t

sceNetGlueGethostbyname

Get host structure from host name

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

Syntax

#include < netglue.h >	
sceNetGlueHostent_t *sceNetGlueGethostbyname(	
const char ****name*);**	Internet host name
#define gethostbyname sceNetGlueGethostbyname	

Calling Conditions

Can be called from a thread

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

Description

This function gets the Internet host structure corresponding to the host name that was specified by the *name* argument and returns a pointer to it.

If an error occurs, details of the error can be obtained with sceNetGlueHErrno.

Return value

Pointer to Internet host structure	Normal termination
NULL	Error

See also

__sceNetGlueHErrnoLOC(), sceNetGlueHostent_t

sceNetGlueGetpeername

Get connection destination information for socket

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueGetpeername(

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

#define getpeername sceNetGlueGetpeername

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceNetGlueSockaddr_t

sceNetGlueGetsockname

Get local information for socket

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueGetsockname(

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

#define getsockname sceNetGlueGetsockname

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

Return value

0	Normal termination
-1	Error

See also

sceNetGlueSockaddr_t

sceNetGlueGetsockopt

Get socket option

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueGetsockopt(

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

#define getsockopt sceNetGlueGetsockopt

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

Table 4-3

Socket Option Level	Meaning
IPPROTO_TCP	TCP-related

Table 4-4

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

Return value

0 Normal termination

-1 Error

sceNetGlueHtonl

Convert 4-byte numeric value from local byte order to network byte order

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

Syntax

#include < netglue.h >	
u_int sceNetGlueHtonl(	
u_int *hostlong***);**	Numeric value for which byte order is to be converted
#define htonl sceNetGlueHtonl	

Calling Conditions

Can be called from a thread

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

Description

This function converts a 4-byte numeric value from local byte order to network byte order.

Return value

Numeric value after converting byte order

sceNetGlueHtons

Convert 2-byte numeric value from local byte order to network byte order

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

Syntax

#include < netglue.h >

u_int sceNetGlueHtons(

u_int *hostshort***);**	Numeric value for which byte order is to be converted

#define htons sceNetGlueHtons

Calling Conditions

Can be called from a thread

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

Description

This function converts a 2-byte numeric value from local byte order to network byte order.

Return value

Numeric value after converting byte order

sceNetGlueInetAddr

Get 32-bit address from dot-format IPv4 address

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

Syntax

#include < netglue.h >	
u_int sceNetGlueInetAddr(	
const char ****cp*);**	Pointer to dot decimal IPv4 address string
#define inet_addr sceNetGlueInetAddr	

Calling Conditions

Can be called from a thread

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

Description

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

Return value

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

sceNetGlueInetAton

Get 32-bit address from dot-format IPv4 address

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

Syntax

#include < netglue.h >

int sceNetGlueInetAton(

const char **cp*,	Pointer to dot decimal IPv4 address string
sceNetGlueInAddr_t **addr*);	Pointer to area for storing 32-bit IPv4 address value after conversion

#define inet_aton sceNetGlueInetAton

Calling Conditions

Can be called from a thread

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

Description

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

Return value

1	Normal termination
0	String is invalid

See also

sceNetGlueInAddr_t

sceNetGlueInetLnaof

Get local network address from IPv4 address

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

Syntax

#include < netglue.h >	
u_int sceNetGlueInetLnaof(	
sceNetGlueInAddr_t *in***);**	32-bit IPv4 address value
#define inet_lnaof sceNetGlueInetLnaof	

Calling Conditions

Can be called from a thread

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

Description

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

Return value

Local network address value

See also

sceNetGlueInAddr_t

sceNetGlueInetMakeaddr

Construct IPv4 address from network address

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

Syntax

#include < netglue.h >	
sceNetGlueInAddr_t sceNetGlueInetMakeaddr(	
u_int *net***,**	Network address portion
u_int *host***);**	Local network address portion
#define inet_makeaddr sceNetGlueInetMakeaddr	

Calling Conditions

Can be called from a thread

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

Description

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

Return value

Combined IPv4 address value

See also

sceNetGlueInAddr_t

sceNetGlueInetNetof

Get network address from IPv4 address

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

Syntax

#include < netglue.h >	
u_int sceNetGlueInetNetof(	
sceNetGlueInAddr_t *in***);**	32-bit IPv4 address value
#define inet_netof sceNetGlueInetNetof	

Calling Conditions

Can be called from a thread

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

Description

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

Return value

Network address value

See also

sceNetGlueInAddr_t

sceNetGlueInetNetwork

Get 32-bit address from dot-format IPv4 address

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

Syntax

#include < netglue.h >	
u_int sceNetGlueInetNetwork(	
const char ****cp*);**	Pointer to dot decimal IPv4 address string
#define inet_network sceNetGlueInetNetwork	

Calling Conditions

Can be called from a thread

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

Description

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

Return value

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

sceNetGlueInetNtoa

Get dot-format address from 32-bit IPv4 address

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

Syntax

#include < netglue.h >	
char *sceNetGlueInetNtoa(	
sceNetGlueInAddr_t *in***);**	32-bit IPv4 address value
#define inet_ntoa sceNetGlueInetNtoa	

Calling Conditions

Can be called from a thread

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

Description

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

Return value

Pointer to dot decimal IPv4 address string

See also

sceNetGlueInAddr_t

sceNetGlueListen

Accept TCP connection

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

Syntax

#include < netglue.h >	
int sceNetGlueListen(	
int *s*,	Descriptor of socket that will wait for the TCP connection
int *backlog***);**	Connection acceptance queue size (number of pending connections)
#define listen sceNetGlueListen	

Calling Conditions

Can be called from a thread

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

Description

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

backlog indicates the maximum size of the connection acceptance queue. If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

0	Normal termination
-1	Error

See also

__sceNetGlueErrnoLoc()

sceNetGlueNtohl

Convert 4-byte numeric value from network byte order to local byte order

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

Syntax

#include < netglue.h >

u_int sceNetGlueNtohl(

u_int *netlong***);**	Numeric value for which byte order is to be converted

#define ntohl sceNetGlueNtohl

Calling Conditions

Can be called from a thread

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

Description

This function converts a 4-byte numeric value from network byte order to local byte order.

Return value

Numeric value after converting byte order

sceNetGlueNtohs

Convert 2-byte numeric value from network byte order to local byte order

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

Syntax

#include < netglue.h >	
u_int sceNetGlueNtohs(	
u_int *netshort***);**	Numeric value for which byte order is to be converted
#define ntohs sceNetGlueNtohs	

Calling Conditions

Can be called from a thread

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

Description

This function converts a 2-byte numeric value from network byte order to local byte order.

Return value

Numeric value after converting byte order

sceNetGlueRecv

Receive data

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

Syntax

#include < netglue.h >	
size_t sceNetGlueRecv(	
int *s*,	Descriptor of socket that is to receive data
**void ****buf*,	Pointer to area for storing receive data
size_t *len*,	Data size to be received (in bytes)
int *flags***);**	Not supported (always set to 0)
#define recv sceNetGlueRecv	

Calling Conditions

Can be called from a thread

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

Description

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

Since the *flags* argument is not supported, it must always be set to 0. If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

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

See also

__sceNetGlueErrnoLoc()

sceNetGlueRecvfrom

Receive data (also get address structure of sending host)

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

size_t sceNetGlueRecvfrom(

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

#define recvfrom sceNetGlueRecvfrom

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

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

If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

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

See also

sceNetGlueSockaddr_t, __sceNetGlueErrnoLoc()

sceNetGlueSend

Send data

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

Syntax

#include < netglue.h >

size_t sceNetGlueSend(

int *s*,	Descriptor of socket that is to send data
void **buf*,	Pointer to send data
size_t *len*,	Size of data to be sent (in bytes)
int *flags***);**	Not supported (always set to 0)

#define send sceNetGlueSend

Calling Conditions

Can be called from a thread

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

Description

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

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

If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

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

See also

__sceNetGlueErrnoLoc()

sceNetGlueSendto

Send data (specify address structure of destination host)

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

size_t sceNetGlueSendto(

int *s***,**	Descriptor of socket that is to send data
void **buf***,**	Pointer to send data
size_t *len***,**	Size of data to be sent (in bytes)
int *flags***,**	Not supported (always set to 0)
sceNetGlueSockaddr_t **addr***,**	Pointer to address structure of destination host
sceNetGlueSocklen_t *addrlen***);**	Size of address structure of destination host (always 16 bytes)

#define sendto sceNetGlueSendto

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

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

See also

sceNetGlueSockaddr_t, __sceNetGlueErrnoLoc()

sceNetGlueSetSifMBindRpcValue

Set buffer size, stack size, and priority

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

Syntax

#include < netglue.h>

int sceNetGlueSetSifMBindRpcValue(

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

Calling Conditions

Can be called from a thread

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

Description

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

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

Return value

0	Normal termination
-1	Error

sceNetGlueSetsockopt

Set socket option

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

Syntax

#include < netglue.h >

typedef u_int sceNetGlueSocklen_t;

int sceNetGlueSetsockopt(

int *s*,	Descriptor of socket for which socket option is to be set
int *level*,	Socket option level
int *optname*,	Socket option name
void **optval*,	Pointer to area for storing socket option value
sceNetGluesocklen_t *optlen***);**	Size of socket option value

#define setsockopt sceNetGlueSetsockopt

#define socklen_t sceNetGlueSocklen_t

Calling Conditions

Can be called from a thread

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

Description

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

Table 4-5

Socket Option Level	Meaning
IPPROTO_TCP	TCP-related

Table 4-6

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

Return value

0 Normal termination

-1 Error

sceNetGlueShutdown

Close socket

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

Syntax

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

Calling Conditions

Can be called from a thread

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

Description

This function closes socket s. Specifying a shutdown method with the *how* argument is not supported (half close cannot be performed), so this argument must always be set to 0. If an error occurs, details of the error can be obtained with sceNetGlueErrno.

Return value

0	Normal termination
-1	Error

See also

__sceNetGlueErrnoLoc()

sceNetGlueSocket

Create socket

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

Syntax

#include < netglue.h >

size_t sceNetGlueSocket(

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

#define socket sceNetGlueSocket

Calling Conditions

Can be called from a thread

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

Description

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

Return value

Positive value	Descriptor of generated socket
-1	Error

See also

__sceNetGlueErrnoLoc()

sceNetGlueThreadInit

Perform initialization processing for thread that uses netglue

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

Syntax

#include < netglue.h >

int sceNetGlueThreadInit(

int *thread_id***);**	ID of thread to be initialized

Calling Conditions

Can be called from a thread

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

Description

This function performs initialization so that a thread can use netglue. The thread ID is specified with *thread_id*.

By using this function to perform initialization processing, each thread's state can be maintained internally within the netglue library. If *thread_id* is set to 0, the calling thread will be used.

Return value

0	Normal termination
-1	Error

sceNetGlueThreadTerminate

Perform termination processing for thread that uses netglue

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

Syntax

#include < netglue.h >

int sceNetGlueThreadTerminate(

int *thread_id***);**	ID of thread for which termination processing is to be performed

Calling Conditions

Can be called from a thread

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

Description

When a thread that is using netglue terminates, this function is called to perform netglue termination processing. The thread ID is specified with *thread_id*.

When each thread's state is maintained internally within the netglue library, this function should be called to perform termination processing. If *thread_id* is set to 0, the calling thread will be used.

Return value

0	Normal termination
-1	Error

Chapter 5: Network Configuration GUI Library
Table of Contents

Structures

sceNetGuiCnf_Arg

Argument data for sceNetGuiCnf_Do()

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

Structure

typedef struct sceNetGuiCnf_Arg {

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

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

} sceNetGuiCnf_Arg_t;

Description

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

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

Table 5-1

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

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

See also

sceNetGuiCnfEnvData, sceNetGuiCnf_Color4, sceNetGuiCnf_Do

sceNetGuiCnf_Color

Color data for one vertex of sceNetGuiCnf_Color4 structure

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

Structure

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

Description

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

See also

sceNetGuiCnf_Color4

sceNetGuiCnf_Color4

Color specification structure

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

Structure

typedef struct sceNetGuiCnf_Color4 {

sceNetGuiCnf_Color_t *aColor***[4];**	aColor[0]	Upper left vertex color data
	aColor[1]	Upper right vertex color data
	aColor[2]	Lower left vertex color data
	aColor[3]	Lower right vertex color data

} sceNetGuiCnf_Color_t;

Description

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

Table 5-2

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

See also

sceNetGuiCnf_Arg

sceNetGuiCnfEnvData

Network configuration data

Library	*Introduced*	*Documentation last modified*
ntguicnf	2.4	December 3, 2001

Structure

typedef struct sceNetGuiCnfEnvData {

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

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

} sceNetGuiCnfEnvData_t;

Description

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

Table 5-3

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

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

dialing_type can be any of the following values.

Table 5-4

Constant	Value	Meaning
	-1	No setting

Constant	Value	Meaning
SCE_NETGUICNF_DIALINGTYPE_TONE	0	Tone
SCE_NETGUICNF_DIALINGTYPE_PULSE	1	Pulse

phy_config can be any of the following values.

Table 5-5

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

dhcp can be either of the following values.

Table 5-6

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

pppoe can be any of the following values.

Table 5-7

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

type can be any of the following values.

Table 5-8

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

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

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

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

Table 5-9

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

Member	Description
accm_nego	ifc->want.accm_nego
mtu	ifc->mtu

See also

sceNetGuiCnf_Arg

Function Types

sceNetGuiCnfCallback_Free

free

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

This is a free function that is ANSI-compliant.

Return value

None

sceNetGuiCnfCallback_Malloc

malloc

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

Syntax

#include <ntguicnf.h>	
typedef void * (* sceNetGuiCnfCallback_Malloc)(	
size_t *size***);**	Size of area in bytes

Calling conditions

Can be called from a thread

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

Description

This is a malloc function that is ANSI-compliant.

Return value

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

sceNetGuiCnfCallback_Memalign

memalign

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

Syntax

#include <ntguicnf.h>

typedef void * (*sceNetGuiCnfCallback_Memalign)(

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

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

sceNetGuiCnfCallback_PadRead

Get controller's button information

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

Syntax

#include <ntguicnf.h>

typedef void * (*sceNetGuiCnfCallback_PadRead)(

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

None

sceNetGuiCnfCallback_Realloc

realloc

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

Syntax

#include <ntguicnf.h>

typedef void * (*sceNetGuiCnfCallback_Realloc)(

void * *old_ptr*,	Area to be reallocated
size_t *new_size*);	Size of area in bytes

Calling conditions

Can be called from a thread

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

Description

This is a realloc function that is ANSI-compliant.

Return value

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

sceNetGuiCnfCallback_SJIStoUTF8

Convert string from Shift-JIS to UTF8

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

Syntax

#include <netguicnf.h>

typedef void (*sceNetGuiCnfCallback_SJIStoUTF8)(

unsigned char * *dst***,**	Output buffer pointer
size_t *dst_size***,**	Output buffer size
unsigned char const * *src***);**	Input string

Calling conditions

Can be called from a thread

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

Description

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

Return value

None

sceNetGuiCnfCallback_SKBDestroy

Software keyboard termination processing

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

This function performs software keyboard termination processing.

Notes

This function is called only once by sceNetGuiCnf_Do().

Return value

None

sceNetGuiCnfCallback_SKBEnableKey

Configure software keyboard key states

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

Syntax

#include <ntguicnf.h>

typedef void
(*sceNetGuiCnfCallback_SKBEnableKey)(

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

Calling conditions

Can be called from a thread

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

Description

This function enables/disables keys on the software keyboard.

type can have any of the following values.

Table 5-10

Constant	Value	Meaning
SCE_NETGUICNF_ENABLE_ KEY_TYPE_ENABLE_LISTED _AND_DISABLE_NOTLISTED	0	Enable listed keys and disable other keys
SCE_NETGUICNF_ENABLE_ KEY_TYPE_ENABLE_ALL	1	Enable all keys
SCE_NETGUICNF_ENABLE_ KEY_TYPE_DISABLE_LISTED	2	Disable listed keys (do nothing to other keys)

Notes

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

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

Return value

None

sceNetGuiCnfCallback_SKBEveryFrame

Software keyboard every frame processing

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

This function performs every frame processing for the software keyboard.

Return value

None

sceNetGuiCnfCallback_SKBGetStatus

Get software keyboard size

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

Syntax

#include <ntguicnf.h>

typedef void (*sceNetGuiCnfCallback_SKBGetStatus)(

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

Calling conditions

Can be called from a thread

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

Description

This function returns the size of the software keyboard.

Return value

None

sceNetGuiCnfCallback_SKBGetVif1PktTopAddr

Get software keyboard drawing packet address

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

Syntax

#include <ntguicnf.h>

typedef void * (*sceNetGuiCnfCallback_SKBGetVif1PktTopAddr)(
void);

Calling conditions

Can be called from a thread

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

Description

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

Notes

The drawing packet must satisfy the following specifications.

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

Return value

Starting address of software keyboard drawing packet.

sceNetGuiCnfCallback_SKBInit

Initialize software keyboard

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

This function initializes the software keyboard.

Notes

This function is called only once by sceNetGuiCnf_Do().

Return value

None

sceNetGuiCnfCallback_SKBSendMouseMessage

Send mouse pointer message

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

Syntax

#include <ntguicnf.h>

typedef int (*sceNetGuiCnfCallback_SKBSendMouseMessage)(

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

Calling conditions

Can be called from a thread

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

Description

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

type can be any of the following values.

Table 5-11

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

Return value

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

sceNetGuiCnfCallback_UsbKbRead

Receive USB keyboard input

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

None

sceNetGuiCnfCallback_UsbMouseRead

Receive USB mouse input

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

Syntax

#include <netguicnf.h>

typedef void
(*sceNetGuiCnfCallback_UsbMouseRead)(

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

Calling conditions

Can be called from a thread

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

Description

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

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

Table 5-12

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

Return value

None

sceNetGuiCnfCallback_UTF8toSJIS

Convert string from UTF8 to Shift-JIS

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

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

Return value

None

Functions

sceNetGuiCnf_Do

Start network configuration application

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

Syntax

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

Calling conditions

Can be called from a thread

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

Description

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

Required IOP modules

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

IOP module required to autoload USB connection device driver

- usbmload.irx

IOP modules required to use the hard disk drive

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

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

Return value

None

sceNetGuiCnf_SendKBMessage

Send key information to network configuration application

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

Syntax

#include <netguicnf.h>

void sceNetGuiCnf_SendKBMessage(

int *type*,	Keyboard type
unsigned char * *keyname***);**	Key identification characters

Calling conditions

Can be called from a thread

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

Description

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

type can be any of the following values.

Table 5-13

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

Remark

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

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

Return value

None