libhttp Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceHttpUriElement

Structure storing URI elements

Definition

```
#include <libhttp.h>
typedef struct SceHttpUriElement{
        SceBool opaque;
        char *scheme;
        char *username;
        char *password;
        char *hostname;
        char *path;
        char *query;
        char *fragment;
        SceUShort16 port;
        SceUChar8 reserved[10];
} SceHttpUriElement;
```

Members

opaque	Whether "//" exists after a *scheme* string or not (SCE_TRUE if it does not exist, SCE_FALSE if it does)
scheme	URI scheme name
username	URI username
password	URI password
hostname	URI hostname
path	URI pathname
query	URI query
fragment	URI fragment
port	URI port number
reserved	Reserved area

Description

In sceHttpUriParse(), this structure is used to store URI elements after parsing is completed, and in sceHttpUriBuild(), URIs are created using the values specified in this structure.

See Also

sceHttpUriParse(), sceHttpUriBuild()

SceHttpMemoryPoolStats

Structure to which memory pool status is stored

Definition

```
#include <libhttp.h>
typedef struct SceHttpMemoryPoolStats{
        SceSize poolSize;
        SceSize maxInuseSize;
        SceSize currentInuseSize;
        SceInt32 reserved;
} SceHttpMemoryPoolStats;
```

Members

poolSize	Size of memory pool specified in `sceHttpInit()`
maxInuseSize	Maximum memory size used by libhttp from `sceHttpInit()` onward
currentInuseSize	Current memory size used by libhttp
reserved	Reserved area

Description

This structure is used to store the current memory pool status with `sceHttpGetMemoryPoolStats()`.

See Also

`sceHttpGetMemoryPoolStats()`

SceHttpCookieStats

Structure to store the cookie storage status

Definition

```
#include <libhttp.h>
typedef struct SceHttpCookieStats{
        SceSize currentInuseSize;
        SceUInt32 currentInuseNum;
        SceSize maxInuseSize;
        SceUInt32 maxInuseNum;
        SceUInt32 removedNum;
        SceInt32 reserved;
} SceHttpCookieStats;
```

Members

currentInuseSize	Size of memory used to store cookies in the current memory pool
currentInuseNum	Number of cookies stored in the current memory pool
maxInuseSize	Maximum memory volume used to store cookies after `sceHttpInit()`
maxInuseNum	Maximum number of cookies stored after `sceHttpInit()`
removedNum	Number of cookies removed after `sceHttpInit()`
reserved	Reserved area

Description

This structure is used to store the current cookie storage status with `sceHttpGetCookieStats()`.

See Also

`sceHttpGetCookieStats()`

SceHttpNBEvent

Structure to store the non-blocking request status

Definition

```
#include <libhttp.h>
typedef struct SceHttpNBEvent {
        SceUInt32 events;
        SceUInt32 eventDetail;
        SceInt32 id;
        void* userArg;
} SceHttpNBEvent;
```

Members

events	Request status
eventDetail	Reserved area
id	ID of request object whose status is stored in this structure
userArg	Pointer specified by user with `sceHttpSetEpoll()`

Description

This structure is used to store the statuses of non-blocking requests with `sceHttpWaitRequest()`.

See Also

`sceHttpWaitRequest()`

SceHttpEpollHandle

epoll handle used to obtain the state of a non-blocking request

Definition

```
#include <libhttp.h>
typedef void* SceHttpEpollHandle;
```

Description

The epoll handle is used to obtain the state of a non-blocking request with `sceHttpWaitRequest()`. The application must not make a direct access into the handle.

See Also

`sceHttpCreateEpoll()`, `sceHttpDestroyEpoll()`, `sceHttpSetEpoll()`, `sceHttpUnsetEpoll()`, `sceHttpWaitRequest()`

Constant Definitions

SceHttpHttpVersion

HTTP version constants

Definition

```
#include <libhttp.h>
typedef enum{
        SCE_HTTP_VERSION_1_0=1,
        SCE_HTTP_VERSION_1_1
} SceHttpHttpVersion;
```

Description

These constants represent the HTTP versions. They are used by `sceHttpCreateTemplate()`.

See Also

`sceHttpCreateTemplate()`

SceHttpMethods

HTTP method constants

Definition

```
#include <libhttp/http_methods.h>
typedef enum{
        SCE_HTTP_METHOD_GET,
        SCE_HTTP_METHOD_POST,
        SCE_HTTP_METHOD_HEAD,
        SCE_HTTP_METHOD_OPTIONS,
        SCE_HTTP_METHOD_PUT,
        SCE_HTTP_METHOD_DELETE,
        SCE_HTTP_METHOD_TRACE,
        SCE_HTTP_METHOD_CONNECT
} SceHttpMethods;
```

Description

These constants represent the options used in `sceHttpCreateRequest()` and `sceHttpCreateRequestWithURL()`.

Notes

Although `SCE_HTTP_METHOD_OPTIONS` and `SCE_HTTP_METHOD_CONNECT` are defined as constants, the associated HTTP methods for them are not supported.

See Also

`sceHttpCreateRequest(),sceHttpCreateRequestWithURL()`

SceHttpAddHeaderMode

HTTP header addition mode constants

Definition

```
#include <libhttp.h>
typedef enum{
        SCE_HTTP_HEADER_OVERWRITE,
        SCE_HTTP_HEADER_ADD
} SceHttpAddHeaderMode;
```

Description

These constants specify whether to append or overwrite when adding a header using `sceHttpAddRequestHeader()`.

See Also

`sceHttpAddRequestHeader()`

Initialization/Termination Functions

sceHttpInit

Initialize libhttp

Definition

```
#include <libhttp.h>
int sceHttpInit(
        SceSize poolSize
);
```

Arguments

poolSize	Size of memory pool used by library

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space
`SCE_HTTP_ERROR_ALREADY_INITED`	0x80431020	`sceHttpInit()` was called a second time without calling `sceHttpTerm()`

Description

This function initializes the libhttp. This function must be called before using any other functions in libhttp. This function allocates from the system a memory pool of *poolSize* bytes which is used as a memory pool for this library.

The size of the memory pool to specify must be a multiple of 4 KiB.

Examples

```
#define LIBHTTP_POOLSIZE     (30 * 1024)

ret = sceHttpInit(LIBHTTP_POOLSIZE);
if(ret < 0){
        // error handling
}
```

Notes

This function is not multithread safe.

See Also

`sceHttpTerm()`

sceHttpTerm

Terminate libhttp

Definition

```
#include <libhttp.h>
int sceHttpTerm(
        void
);
#define sceHttpEnd() sceHttpTerm()
```

Arguments

None

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	Before library initialization

Description

This function terminates the libhttp, and frees memory which it had allocated.

Notes

Call this function after using `sceHttpDeleteTemplate()`, `sceHttpDeleteConnection()`, and `sceHttpDeleteRequest()` to delete all objects created with `sceHttpCreateTemplate()`, `sceHttpCreateConnection()`, and `sceHttpCreateRequest()`.

This function is not multithread safe.

See Also

`sceHttpInit()`, `sceHttpCreateTemplate()`, `sceHttpCreateConnection()`, `sceHttpCreateRequest()`, `sceHttpDeleteTemplate()`, `sceHttpDeleteConnection()`, `sceHttpDeleteRequest()`

Memory Management Functions

sceHttpGetMemoryPoolStats

Get memory pool status

Definition

```
#include <libhttp.h>
int sceHttpGetMemoryPoolStats (
        SceHttpMemoryPoolStats* currentStat
);
```

Arguments

`currentStat`	Memory address storing memory pool status

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_BROKEN`	0x80431085	The memory pool status is invalid

Description

This function obtains the status of the memory pool currently being used by libhttp. This function can be used to obtain the maximum size of the memory pool, in other words the memory pool size specified with `sceHttpInit()`, the currently used size, and the maximum memory usage size from `sceHttpInit()` until now.

See Also

`sceHttpInit()`

HTTP Object Creation/Deletion Functions

sceHttpCreateTemplate

Create a template for HTTP settings

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateTemplate(
        const char*userAgent,
        SceInt32 httpVer,
        SceBool autoProxyConf
);
```

Arguments

userAgent	Contains a pointer to the user agent name, which is stored as an ASCIZ string
httpVer	HTTP version (details below)
autoProxyConf	Whether or not to use the HTTP Proxy settings stored in the system SCE_TRUE: Uses system settings SCE_FALSE: Does not use system settings

The HTTP versions to specify for *httpVer* are defined as follows.

Value	Description
SCE_HTTP_VERSION_1_0	HTTP 1.0
SCE_HTTP_VERSION_1_1	HTTP 1.1

Return Values

If this function completes normally, the ID (>0) of the created template settings is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	Insufficient free memory space
SCE_HTTP_ERROR_INVALID_VERSION	0x8043106a	The HTTP version is invalid
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized

Description

The *userAgent* argument specifies a pointer to the string to be used as the user agent. When the string is actually sent out as a request header, a token that represents both the libhttp and system software version is added to the end of the string specified here.

The *httpVer* argument specifies the HTTP version. Note that when SCE_HTTP_VERSION_1_0 is specified, the functionality added starting with Version 1.1 cannot as a rule be used.

If SCE_TRUE is set to *autoProxyConf*, the HTTP(S) proxy server settings will be automatically read in from the network settings. Since, due to the nature of wireless LANs, there is a good chance that the need for a proxy will change based on the connected base station, the SCE_TRUE setting, which automatically determines whether or not a proxy is necessary from the currently utilized network settings, is recommended.

Notes

This function creates basic HTTP settings (referred to below as template settings). In the libhttp library, the settings are divided into three stages - template settings, connection settings, and request settings - and are classified respectively as settings for items which do not depend strongly upon the accessed server, settings for each accessed server, and settings for each request. For details, please refer to the "libhttp Overview" document.

See Also

`sceHttpDeleteTemplate()`

sceHttpDeleteTemplate

Delete template settings

Definition

```
#include <libhttp.h>
SceInt32 sceHttpDeleteTemplate(
        SceInt32 templateId
);
```

Arguments

templateId	ID of template to delete

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID is invalid
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized

Description

This function deletes the template settings having the ID specified by *templateId*, and frees the memory which had been used for them.

See Also

`sceHttpCreateTemplate()`

sceHttpCreateConnection

Create settings for an individual connected server

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateConnection(
        SceInt32 tmplId,
        const char *server,
        const char *scheme,
        SceUShort16 port,
        SceBool enableKeepalive
);
```

Arguments

tmplId	ID of template settings to use
server	Pointer to the host name of the server to access, which is stored as an ASCIZ string
scheme	Specifies HTTP or HTTPS Pointer to the scheme (e.g., "http"), which is stored as an ASCIZ string
port	Port number to access
enableKeepalive	Whether or not to use HTTP Keep-Alive `SCE_TRUE`: Enables Keep-Alive `SCE_FALSE`: Disables Keep-Alive

Return Values

If this function completes normally, the ID (>0) of the created connection settings is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified template ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_UNKNOWN_SCHEME`	0x80431061	A scheme other than http or https was specified for the URI

Description

This function creates settings for each connected server. Template settings must first be created using `sceHttpCreateTemplate()`.

The value to specify for *`port`* must not be converted to network byte order since it is not necessary.

Notes

The TCP connection with the server is actually established when a request object is created using `sceHttpCreateRequest()` and then `sceHttpSendRequest()` is called.

See Also

`sceHttpDeleteConnection()`, `sceHttpCreateTemplate()`, `sceHttpCreateRequest()`, `sceHttpSendRequest()`

sceHttpCreateConnectionWithURL

Create settings for an individual connected server

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateConnectionWithURL(
        SceInt32 tmplId,
        const char *url,
        SceBool enableKeepalive
);
```

Arguments

tmplId	ID of template settings
url	Contains a pointer to the URL to access, which is stored as an ASCIZ string
enableKeepalive	Whether or not to use HTTP Keep-Alive SCE_TRUE: Enables Keep-Alive SCE_FALSE: Disables Keep-Alive

Return Values

If this function completes normally, the ID (>0) of the created connection settings is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	Insufficient free memory space
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified template ID is invalid
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_UNKNOWN_SCHEME	0x80431061	A scheme other than http or https was specified in the URI

Description

This function is like `sceHttpCreateConnection()` with URL parsing capability. Template settings must first be created using `sceHttpCreateTemplate()`. This function parses the string which is specified by *url*, obtains the protocol (HTTP, HTTPS), the host name of the server, the port number, the user name to use for authentication, and the password, and creates the connection settings. The memory allocated within the library when the URL is parsed is released when the connection settings are deleted using `sceHttpDeleteConnection()`.

Notes

Note that this function does not utilize the path segment of the URL.

See Also

`sceHttpCreateConnection(), sceHttpDeleteConnection(), sceHttpCreateTemplate()`

sceHttpDeleteConnection

Delete settings for an individual connected server

Definition

```
#include <libhttp.h>
SceInt32 sceHttpDeleteConnection (
        SceInt32 connId
);
```

Arguments

connId	ID of the connection settings to delete

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function deletes the connection settings having the ID specified by *connId*, and frees the memory which had been used for them.

If HTTP Keep-Alive is enabled and a TCP connection is open, it is closed.

See Also

`sceHttpCreateConnection()`, `sceHttpCreateConnectionWithURL()`

sceHttpCreateRequest

Create a request object

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateRequest(
        SceInt32 connId,
        SceInt32 method,
        const char *path,
        SceULong64 contentLength
);
```

Arguments

connId	ID of the connection settings to use
method	HTTP method to use (details below)
path	Pointer to the path to access, which is stored as an ASCIZ string
contentLength	Number of bytes for the total size when using the POST method 0 when using a method other than POST

Specify one of the following values for *method*.

Value	Description
SCE_HTTP_METHOD_GET	GET method
SCE_HTTP_METHOD_HEAD	HEAD method
SCE_HTTP_METHOD_POST	POST method
SCE_HTTP_METHOD_PUT	PUT method
SCE_HTTP_METHOD_DELETE	DELETE method
SCE_HTTP_METHOD_TRACE	TRACE method

Return Values

If this function completes normally, the ID (>0) of the created request object is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	Insufficient free memory space
SCE_HTTP_ERROR_INVALID_VERSION	0x8043106a	Using connection settings specifying 1.0 as the HTTP version, specifies PUT or DELETE to *method*
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The ID of the connection setting is invalid
SCE_HTTP_ERROR_UNKNOWN_METHOD	0x8043106b	The value specified in *method* is invalid

Description

This function creates a request to the connection destination specified by *connId*. Connection settings must first be created using `sceHttpCreateConnection()` or `sceHttpCreateConnectionWithURL()`. Note that sending of data is not performed until `sceHttpSendRequest()` is called.

See Also

`sceHttpCreateConnection()`, `sceHttpCreateConnectionWithURL()`, `sceHttpDeleteRequest()`, `sceHttpCreateRequestWithURL()`

sceHttpCreateRequestWithURL

Create a request object

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateRequestWithURL(
        SceInt32 connId,
        SceInt32 method,
        const char *url,
        SceULong64 contentLength
);
```

Arguments

connId	ID of the connection settings to use
method	HTTP method to use (details below)
url	Pointer to the URL to access, which is stored as an ASCIZ string
contentLength	Number of bytes for the total size when using the POST method 0 when using a method other than POST

Specify one of the following values for *method*.

Value	Description
`SCE_HTTP_METHOD_GET`	GET method
`SCE_HTTP_METHOD_HEAD`	HEAD method
`SCE_HTTP_METHOD_POST`	POST method
`SCE_HTTP_METHOD_PUT`	PUT method
`SCE_HTTP_METHOD_DELETE`	DELETE method
`SCE_HTTP_METHOD_TRACE`	TRACE method

Return Values

If this function completes normally, the ID (>0) of the created request object is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space
`SCE_HTTP_ERROR_INVALID_VERSION`	0x8043106a	Using connection settings specifying 1.0 as the HTTP version, specifies PUT or DELETE to *method*
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The ID of the connection settings is invalid
`SCE_HTTP_ERROR_UNKNOWN_METHOD`	0x8043106b	The value specified in *method* is invalid

Description

This function is like `sceHttpCreateRequest()` with URL parsing capability.

Connection settings must first be created using `sceHttpCreateConnection()` or `sceHttpCreateConnectionWithURL()`. This function parses the string specified by *url*, and creates a request object. The memory allocated within the library when the URL is parsed is released when the request object is deleted using `sceHttpDeleteRequest()`.

Notes

Note that this function does not utilize any portion of the URL other than the path, username and password.

See Also

`sceHttpCreateConnection()`, `sceHttpCreateConnectionWithURL()`, `sceHttpDeleteRequest()`

sceHttpCreateRequest2

Create a request (path specification)

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateRequest2(
        SceInt32 connId,
        const char *method,
        const char *path,
        SceULong64 contentLength
);
```

Arguments

connId	ID of the connection settings to use
method	HTTP method to use (ASCIIZ string)
path	Path to access (ASCIIZ string)
contentLength	Total size (number of bytes) of the data to send as the request body 0 when not sending a request body and when transfer encoding is enabled

Return Values

If this function completes normally, the ID (>0) of the created request is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID of the connection setting is invalid

Description

This function creates a request to the connection target specified by *connId*. For *method*, an arbitrary HTTP method can be specified. Otherwise, it is the same as `sceHttpCreateRequest()`.

See Also

`sceHttpCreateConnection()`, `sceHttpCreateConnectionWithURL()`, `sceHttpDeleteRequest()`, `sceHttpCreateRequest()`

sceHttpCreateRequestWithURL2

Create a request (URL specification)

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateRequestWithURL2(
        SceInt32 connId,
        const char* method,
        const char *url,
        SceULong64 contentLength
);
```

Arguments

connId	ID of the connection settings to use
method	HTTP method to use (ASCIIZ string)
url	URL to access (ASCIIZ string)
contentLength	Total size (number of bytes) of the data to send as the request body 0 when not sending a request body and when transfer encoding is enabled

Return Values

If this function completes normally, the ID (>0) of the created request is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	Insufficient free memory space
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID of the connection settings is invalid

Description

This function creates a request to the connection target specified by *connId*. For *method*, an arbitrary HTTP method can be specified. Otherwise, it is the same as `sceHttpCreateRequestWithURL()`.

See Also

`sceHttpCreateConnection()`, `sceHttpCreateConnectionWithURL()`, `sceHttpDeleteRequest()`, `sceHttpCreateRequestWithURL()`

sceHttpDeleteRequest

Delete a request object

Definition

```
#include <libhttp.h>
SceInt32 sceHttpDeleteRequest(
        SceInt32 reqId
);
```

Arguments

reqId	ID of request object to delete

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function deletes the request object specified by *reqId*, and frees the memory which had been allocated for it.

See Also

`sceHttpCreateRequest()`, `sceHttpCreateRequestWithURL()`

HTTP Communication Processing Functions

sceHttpSendRequest

Send an HTTP request

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSendRequest(
        SceInt32 reqId,
        const void *postData,
        SceSize size
);
```

Arguments

reqId	ID of request object to send
postData	Starting address of memory containing data on which to perform POST NULL for any method other than POST
size	Size of *postData* 0 for any method other than POST

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_BUSY`	0x80431021	One of these three has occurred: - Attempted to send multiple requests simultaneously in a multithreaded environment when using a single connection object - Attempted to send the next request before completely receiving the response to the previous request with `sceHttpReadData()`, using a single connection object - Attempted to send another request during the sending of multipart POST data
`SCE_HTTP_ERROR_NETWORK`	0x80431063	An error was returned by the TCP stack
`SCE_HTTP_ERROR_TIMEOUT`	0x80431068	Either the timeout period set using timeout setting function or the TCP timeout period has elapsed
`SCE_HTTP_ERROR_EAGAIN`	0x80431082	Request is blocked
`SCE_HTTP_ERROR_PROXY`	0x80431084	Failed to establish the connection to the HTTP Proxy

Description

This function sends the request object specified by *reqId* to the server. If POST is specified as the method of the request object, specify in *postData* the starting address of the memory which contains the data to be used in the request body, and specify in *size* the size of the specified data. If the size of the Content-Length cannot be allocated all at once in the memory then call `sceHttpSendRequest()` multiple times. Content-Length is set when a request object is created using the `sceHttpCreateRequest()` function; set it to the total size to be sent using POST.

When a request is set to blocking mode and a request body will not be sent, this function does not return until the request has been sent and a response header is received from the server. When sending a request body, if there is any remaining data to send, then this function returns immediately when that data has been sent. After the data size specified with Content-Length has completed sending or when the last chunk has been sent, then this function waits until the response header is received from the server and then it returns.

On the other hand, when a request is set to non-blocking mode and network sending/receiving is blocked due to incomplete communication or buffer overflow when a request send or response header receive has not completed, `SCE_HTTP_ERROR_EAGAIN` will return as the return value. However, request send/receive processing will be performed without it returning as long as network sending/receiving can continue, therefore blocking may occur for several dozen milliseconds or more, particularly in HTTPS communication where internal library processing is frequent. `sceHttpWaitRequest()` can be used to determine if network sending/receiving can continue without blocking or not. If network sending/receiving can continue, call `sceHttpSendRequest()` (this function) multiple times until normal termination occurs.

Notes

With libhttp, chunked encoding is supported for receive only; it is not supported for send. As a result, when sending data using the POST method, Content-Length must be specified. Moreover, since libhttp does not detect the format of data to POST, add the Content-Type header using `sceHttpAddRequestHeader()` as necessary.

If the connection to the HTTP Proxy failed to be established, `sceHttpSendRequest()` returns `SCE_HTTP_ERROR_PROXY`. When the further details on the failure is required, use `sceHttpGetLastErrno()` to obtain the details.

Examples

```
/* Initialization of libssl (if https communication is performed)*/
ret = sceSslInit(SSL_POOL_SIZE);
if (ret < 0){
        goto ssl_term;
}
/* Initialize libhttp */
ret = sceHttpInit(HTTP_POOLSIZE);
if (ret < 0){
     goto http_term;
}

/* Create template configuration */
tmplId = sceHttpCreateTemplate(USER_AGENT, SCE_HTTP_VERSION_1_1,
                     SCE_TRUE);
if (tmplId < 0){
     goto http_term;
}

/* Create connection configuration */
```

```
connId = sceHttpCreateConnectionWithURL(tmplId, url,
              SCE_TRUE);
if(connId < 0){
     goto http_term;
}

/* Create request object */
reqId = sceHttpCreateRequestWithURL(connId, SCE_HTTP_METHOD_POST,
     url, sizeof(postData));
if (reqId < 0){
     goto http_term;
}

/* Send request */
ret = sceHttpSendRequest(reqId, postData, sizeof(postData));

if (ret < 0){
     printf("sceHttpSendRequest() returns %x\n", ret);
     goto http_term;
}

---
```

Example of sending POST data by dividing into multiple transmissions

```
/* Obtain size of file to be uploaded */
ret = sceIoGetstat(uploadFile, &fstat);
if (ret < 0){
     goto http_term;
}
requestLength = fstat.st_size;

/* Create request using the file size */
reqId2 = sceHttpCreateRequestWithURL(connId, SCE_HTTP_METHOD_POST,
     uri, request_length);
if (reqId2 < 0){
     goto http_term;
}

/* Open file to be uploaded */
ret = sceIoOpen(uploadFile, SCE_O_RDONLY, 0);
if (ret < 0){
     goto http_term;
}

/* Subdivide file size for transmission */
while (requestLength > 0){
     /* Read one subdivided portion */
     if (requestLength > UPLOAD_BLOCKSIZE){
        sendSize = UPLOAD_BLOCKSIZE;
     } else {
        sendSize = (SceSize) requestLength;
     }
     ret = sceIoRead(fd, uploadBuf, sendSize);
     if (ret < 0){
        goto http_term;
     }
     /* Send one subdivided portion */
     ret = sceHttpSendRequest(reqId2, uploadBuf, sendSize);
     if (ret < 0){
        goto http_term;
     }
     /* Subtract transmitted size from total POST size */
```

```
            requestLength -= sendSize;
        }
```

See Also

`sceHttpCreateRequest(),sceHttpAbortRequest(),sceHttpGetLastErrno()`

sceHttpAbortRequest

Abort transmission of an HTTP request

Definition

```
#include <libhttp.h>
SceInt32 sceHttpAbortRequest(
        SceInt32 reqId
);
```

Arguments

reqId	ID of request object to abort

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function immediately aborts communication with the server involving the request object specified by *reqId*, and the corresponding `sceHttpSendRequest()` or `sceHttpReadData()` immediately returns.

Notes

All currently existing functions of libhttp are blocking functions. With methods other than POST, `sceHttpSendRequest()` does not return until the request is sent and a response header is received from the server. With the POST method, if the amount of data specified by Content-Length has not been sent, then this function returns when the specified data has been sent, and if the data size specified by Content-Length has already been sent, then this function returns when the response header is received from the server.

See Also

`sceHttpSendRequest()`, `sceHttpReadData()`

sceHttpReadData

Read the response body

Definition

```
#include <libhttp.h>
SceInt32 sceHttpReadData(
        SceInt32 reqId,
        void *data,
        SceSize size
);
```

Arguments

reqId	ID of the request object
data	Start address of memory to which to store the data obtained
size	Size of the memory specified by *data*

Return Values

Upon normal completion, the size of the data which was written into the memory specified by *data* is returned. When the response body has been received and there is no data to read in, 0 is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_SEND`	0x80431065	The specified request object has not been sent yet
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_READ_BY_HEAD_METHOD`	0x8043106f	This function was called using the HEAD method

Description

This function receives a response body. The received response body will be written to the address specified with *data*.

Blocking is released for `sceHttpReadData()` under the following conditions.

(a) Receiving the number of bytes of data specified by *size* has been completed

(b) (If Content-Length exists) receiving a response body having the size specified by Content-Length has been completed

(c) When 0 or a negative value is returned by `sceNetRecv()`, which is called internally within libhttp

(d) Receiving the final chunk has been completed when chunk encoded data is being received

Notes

When using the HEAD method, the response body is not sent from the server, so this function should not be called.

See Also

`sceHttpSendRequest()`, `sceHttpGetResponseContentLength()`

Response Status Acquisition Functions

sceHttpGetResponseContentLength

Get Content-Length of a response

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetResponseContentLength(
        SceInt32 reqId,
        SceULong64 *contentLength
);
```

Arguments

reqId	ID of the request object
contentLength	Starting address of the memory which contains the byte count representing the size of the response body

Return Values

Upon normal completion, the size of the body of the response from the server corresponding to the specified request object is stored in the memory specified by *contentLength*, and `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_CHUNK_ENC`	0x80431072	The response is in chunked encoding, so Content-Length cannot be obtained
`SCE_HTTP_ERROR_NO_CONTENT_LENGTH`	0x80431071	The response does not include Content-Length, nor is it in chunked encoding
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_SEND`	0x80431065	The specified request object has not been sent yet
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function uses the request object ID which has already been sent using `sceHttpSendRequest()`. Reception of the response headers from the server is complete when `sceHttpSendRequest()` returns.

Notes

If the response from the server is in chunked encoding format, Content-Length cannot be obtained. In such cases, receive the data by successively calling `sceHttpReadData()` until the return value is 0. There are also servers which do not send Content-Length even when the data is not in chunked encoding format. Handle these cases as well by successively calling `sceHttpReadData()` until the return value is 0.

See Also

`sceHttpSendRequest(), sceHttpReadData()`

sceHttpGetStatusCode

Get the response status code

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetStatusCode (
        SceInt32 reqId,
        SceInt32 *statusCode
);
```

Arguments

reqId	ID of the request object
statusCode	Address of memory to store the status code

Return Values

Upon normal completion, the status code of the response from the server corresponding to the specified request object is stored as an integer value in the memory specified by *statusCode*, and SCE_OK (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID is invalid
SCE_HTTP_ERROR_BEFORE_SEND	0x80431065	The specified request object has not been sent yet
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized

Description

This function is used with the ID of a request object which has already been sent using sceHttpSendRequest(). Reception of the response header from the server is complete when sceHttpSendRequest() returns.

Notes

With request objects for which the automatic redirection setting has been enabled, when a response of 300, 301, 303, or 307 is returned from the server, a retry to the location of the redirect is automatically generated within libhttp. In such cases, the status code which can be obtained using sceHttpGetStatusCode() will represent the status code of the response returned by the server at the location of the redirect.

See Also

sceHttpSendRequest()

sceHttpGetAllResponseHeaders

Get response headers

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetAllResponseHeaders(
        SceInt32 reqId,
        char **header,
        SceSize *headerSize
);
#define  sceHttpGetAllHeader(reqId, header, headerSize) \
        sceHttpGetAllResponseHeaders(reqId, header, headerSize)
```

Arguments

reqId	ID of the request object to obtain headers
header	Address to store the start address of the response headers, which are stored as ASCIZ strings
headerSize	Size of the headers specified in *header*

Return Values

Upon normal completion, the start address of the response headers, which are stored as ASCIZ strings, is contained in *header*, the size of the headers is stored in *headerSize*, and `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_SEND`	0x80431065	The specified request object has not been sent yet
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function is used with the ID of a request object which has already been sent using `sceHttpSendRequest()`. Reception of the response headers from the server is complete when `sceHttpSendRequest()` returns. Further, the data obtained is released when `sceHttpDeleteRequest()` is called for the corresponding request object. If there is data which needs to be saved, the application must allocate memory and copy the data prior to calling `sceHttpDeleteRequest()`.

Notes

To parse the header that was obtained, use the `sceHttpParseXxx()` API.

To obtain Content-Length and status code, the dedicated functions `sceHttpGetResponseContentLength()` and `sceHttpGetStatusCode()` are provided.

See Also

`sceHttpSendRequest()`, `sceHttpDeleteRequest()`, `sceHttpParseStatusLine()`, `sceHttpParseResponseHeader()`

sceHttpSetResponseHeaderMaxSize

Set maximum size of memory to prepare for receiving the response header

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetResponseHeaderMaxSize(
        SceInt32 id,
        SceSize headerSize
);
```

Arguments

id	ID of target template settings or connection settings
headerSize	Maximum size of memory in bytes to prepare for storing the response header

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function specifies the maximum value for the memory size to prepare for storing the response header against the template settings or connections settings specified in *id*. The default value is 1500 bytes.

See Also

`sceHttpSendRequest()`

sceHttpSetInflateGZIPEnabled

Set response body GZIP unzipping

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetInflateGZIPEnabled(
        SceInt32 id,
        SceBool isEnable
);
```

Arguments

id	ID of target template setting, connection setting or request
isEnable	Whether to perform GZIP unzipping or not (`SCE_FALSE` to receive it as-is, `SCE_TRUE` to perform unzipping)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The ID specified for the argument is invalid

Description

This function sets whether to receive data GZIP unzipped with `sceHttpReadData()` or to receive it as-is for GZIP encoded responses. GZIP unzipping is enabled by default.

See Also

`sceHttpReadData()`

Timeout Setting Functions

sceHttpSetResolveTimeOut

Set the name resolution timeout

Definition

```
#include <libhttp.h>
int sceHttpSetResolveTimeOut (
        SceInt32 id,
        SceUInt32 usec
);
```

Arguments

id	ID of the relevant template settings or connection settings
usec	timeout time to be set (in microseconds)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

Store the ID of the template setting or connection setting in *id*, and the timeout for name resolution, in microseconds, is specified by *usec*.

Notes

The ID of a request object cannot be specified to *id*. Furthermore, even if the template settings are modified, these template settings will have no effect on connection settings which have already been created.

libhttp uses 1 second as the default value, with 5 retries as the default number of retries, so the effective default timeout for name resolution is 31 seconds. For details about the timeout time and changing the maximum wait time through the number of retries setting, refer to the "`sceNetResolverStartNtoa`" section of the "libnet Reference" document.

See Also

`sceHttpCreateTemplate()`, `sceHttpCreateConnection()`, `sceHttpSetResolveRetry()`

sceHttpSetResolveRetry

Set number of send retries for name resolution

Definition

```
#include <libhttp.h>
int sceHttpSetResolveRetry (
        SceInt32 id,
        SceInt32 retry
);
```

Arguments

id	ID of the relevant template settings or connection settings
retry	Number of send retries to be set

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

Store the ID of the template setting or connection setting in *id*, and specify the retry count setting in *retry*.

Notes

The ID of a request object cannot be specified in *id*. Furthermore, even if the template settings are modified, these template settings will have no effect on connection settings which have already been created.

The default value of the number of send retries under libhttp is 5. The timeout time set by default being 1 second, name resolution times out after approximately 31 seconds by default. For details about the timeout time and changing the maximum wait time through the number of retries setting, refer to the "`sceNetResolverStartNtoa`" section of the "libnet Reference" document.

See Also

`sceHttpCreateTemplate(),sceHttpCreateConnection()`

sceHttpSetConnectTimeOut

Set the connection timeout

Definition

```
#include <libhttp.h>
int sceHttpSetConnectTimeOut (
        SceInt32 id,
        SceUInt32 usec
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object
usec	timeout time to be set (in microseconds)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

The ID of the relevant template settings, connection settings, or request object is stored in *id*, and the timeout for establishing a TCP connection, in microseconds, is specified by *usec*.

Notes

The connection timeout time is set to 30 seconds by default.

The order of precedence for the settings is template settings < connection settings < request object. Furthermore, even if the template settings are modified, these template settings will not modify the connection settings which have already been created.

See Also

`sceHttpCreateTemplate()`, `sceHttpCreateConnection()`, `sceHttpCreateRequest()`

sceHttpSetSendTimeOut

Set the send timeout

Definition

```
#include <libhttp.h>
int sceHttpSetSendTimeOut (
        SceInt32 id,
        SceUInt32 usec
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object
usec	timeout time to be set (in microseconds)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

The ID of the relevant template settings, connection settings, or request object is stored in *id*, and the send timeout, in microseconds, is specified by *usec*.

Notes

The send timeout time is set to 120 seconds by default.

The order of precedence for the settings is template settings < connection settings < request object. Furthermore, even if the template settings are modified, these template settings will not modify the connection settings which have already been created.

See Also

`sceHttpCreateTemplate(),sceHttpCreateConnection(),sceHttpCreateRequest()`

sceHttpSetRecvTimeOut

Set the receive timeout

Definition

```
#include <libhttp.h>
int sceHttpSetRecvTimeOut (
        SceInt32 id,
        SceUInt32 usec
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object
usec	timeout time to be set (in microseconds)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

The ID of the relevant template settings, connection settings, or request object is stored in *id*, and the receive timeout, in microseconds, is specified in *usec*.

Notes

The receive timeout time is set to 120 seconds by default.

The order of precedence for the settings is template settings < connection settings < request object. Furthermore, even if the template settings are modified, these template settings will not modify the connection settings which have already been created.

See Also

`sceHttpCreateTemplate(),sceHttpCreateConnection(),sceHttpCreateRequest()`

Redirect Setting Functions

sceHttpSetAutoRedirect

Enable and disable automatic redirection

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetAutoRedirect (
        SceInt32 id,
        SceBool enable
);

#define sceHttpEnableRedirect(id) \
        sceHttpSetAutoRedirect(id, SCE_TRUE)
#define sceHttpDisableRedirect(id) \
        sceHttpSetAutoRedirect(id, SCE_FALSE)
```

Arguments

id	ID of the template settings, connection settings or request object for which to enable automatic redirection
enable	Automatic redirection setting (SCE_FALSE: disabled, SCE_TRUE: enabled)

Return Values

If this function completes normally, SCE_OK (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID is invalid
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument

Description

This function enables automatic redirection for the template settings, connection settings or request object specified by *id*. Automatic redirection is enabled by default.

See Also

sceHttpGetAutoRedirect()

sceHttpGetAutoRedirect

Get current automatic redirection setting

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetAutoRedirect (
        SceInt32 id,
        SceBool *enable
);
```

Arguments

id	ID of the template settings, connection settings or request object for which to enable automatic redirection
enable	Pointer to variable to be received automatic redirection setting (`SCE_FALSE`: disabled, `SCE_TRUE`: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function obtains the current automatic redirection settings from the template settings, connection settings or request object specified by *id*.

See Also

`sceHttpSetAutoRedirect()`

sceHttpRedirectCacheFlush

Delete redirection cache

Definition

```
#include <libhttp.h>
SceInt32 sceHttpRedirectCacheFlush (
        void
);
```

Arguments

None

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function deletes all redirection information cached by libhttp.

See Also

`sceHttpSetAutoRedirect()`

Basic/Digest Authentication Setting Functions

sceHttpSetAuthEnabled

Enable and disable Basic/Digest authentication

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetAuthEnabled (
        SceInt32 id,
        SceBool enable
);

#define sceHttpEnableAuth(id) \
        sceHttpSetAuthEnabled(id, SCE_TRUE)
#define sceHttpDisableAuth(id) \
        sceHttpSetAuthEnabled(id, SCE_FALSE)
```

Arguments

id	ID of the template settings, connection settings or request object for which to enable Basic/Digest authentication
enable	Basic/Digest authentication setting (`SCE_FALSE`: disabled, `SCE_TRUE`: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID is invalid
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument

Description

This function enables Basic/Digest authentication for the template settings, connection settings, or request object specified by *id*. Basic/Digest authentication is enabled by default.

See Also

`sceHttpGetAuthEnabled()`

sceHttpGetAuthEnabled

Get current Basic/Digest authentication setting

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetAuthEnabled (
        SceInt32 id,
        SceBool *enable
);
```

Arguments

id	ID of the template settings, connection settings or request object to be obtained the Basic/Digest authentication setting
enable	Pointer to variable to be received Basic/Digest authentication setting (`SCE_FALSE`: disabled, `SCE_TRUE`: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function obtains the current Basic/Digest authentication settings from the template settings, connection settings or request object specified by *id*.

See Also

`sceHttpSetAuthEnabled()`

sceHttpAuthCacheFlush

Delete authentication cache

Definition

```
#include <libhttp.h>
SceInt32 sceHttpAuthCacheFlush (
        void
);
```

Arguments

None

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function deletes all Basic/Digest authentication information cached by libhttp.

See Also

`sceHttpSetAuthEnabled()`

Cookie Setting Functions

sceHttpSetCookieEnabled

Enable and disable cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieEnabled (
        SceInt32 id,
        SceBool enable
);

#define sceHttpEnableCookie(id) \
        sceHttpSetCookieEnabled(id, SCE_TRUE)
#define sceHttpDisableCookie(id) \
        sceHttpSetCookieEnabled(id, SCE_FALSE)
```

Arguments

id	ID of the template settings, connection settings or request object for which to enable cookies
enable	Cookie setting (SCE_FALSE: disabled, SCE_TRUE: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function enables cookies for the template settings, connection settings, or request object specified by *id*. Cookies are enabled by default. Also, even if the template settings are changed, the connection settings already created through the template settings will not be changed.

See Also

`sceHttpGetCookieEnabled()`

sceHttpGetCookieEnabled

Get current Cookie setting

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetCookieEnabled (
        SceInt32 id,
        SceBool *enable
);
```

Arguments

id	ID of the template settings, connection settings or request object to be obtained the Cookie setting
enable	Pointer to variable to be received cookie setting (`SCE_FALSE`: disabled, `SCE_TRUE`: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function obtains the current cookie settings from the template settings, connection settings or request object specified by *`id`*.

See Also

`sceHttpSetCookieEnabled()`

sceHttpCookieExport

Write cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCookieExport(
        void *buffer,
        SceSize size,
        SceSize *exportSize
)
```

Arguments

buffer	Pointer to the buffer to which the cookie images are written
size	Size of the buffer to which the cookie images are written
exportSize	Pointer to a variable storing the required buffer size for writing cookie images

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument
`SCE_HTTP_ERROR_OUT_OF_SIZE`	0x80431104	The buffer size is insufficient

Description

This function writes the cookie images into the buffer specified by *buffer* and *size*. By specifying *exportSize*, the required buffer size for writing cookie images will be returned. If NULL is specified to *buffer*, only the required buffer size can be obtained, however, note that the required buffer size will change if communication that affects the cookies is performed.

The written cookie images can be reloaded with `sceHttpCookieImport()`.

Notes

This function is not multithread safe.

See Also

`sceHttpCookieImport()`

sceHttpCookieImport

Read cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCookieImport(
        const void *buffer,
        SceSize size,
)
```

Arguments

buffer	Pointer to the cookie images to be read
size	Size of the cookie image buffer to be read

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space

Description

This function reads the cookie image buffer specified by *buffer* and *size*. The buffer to be specified must be the buffer in which the cookie images obtained with `sceHttpCookieExport()` is stored.

Notes

This function is not multithread safe.

See Also

`sceHttpCookieExport()`

sceHttpGetCookie

Get cookies

Definition

```
#include <libhttp.h>
int sceHttpGetCookie(
        const char *url,
        char *cookie,
        unsigned int *required,
        unsigned int prepared,
        int secure
);
```

Arguments

url	Pointer to URL to obtain *cookie* stored as ASCIIZ character string
cookie	Address storing ASCIIZ character string of cookie data associated with *url* The memory must be allocated at the application level By specifying NULL to this argument, the required memory size can be obtained beforehand
required	Required memory size for storing cookies
prepared	Memory size prepared for *cookie*
secure	Flag for indicating whether the connection using cookie data obtained through the relevant API is secure or not. Specify `SCE_TRUE` or `SCE_FALSE`

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	NULL is specified in *url*
`SCE_HTTP_ERROR_OUT_OF_SIZE`	0x80431104	Cookie cannot be stored because the size specified in *prepared* is insufficient
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space

Description

This function obtains the cookie specified in *cookie* from the cookie list managed by libhttp. Since libhttp can automatically handle the usual cookies specified with the response header, use this function to manually obtain cookies through javascript, etc.

Note that the cookies must be handled in a way to prevent users from any disadvantages.

Examples

```
SceSize malloc_size = 0;
SceChar8 *cookie;
/* Obtain memory size for storing cookie */
ret = sceHttpGetCookie(http://example.com/foobar/, NULL, &malloc_size, 0,
SCE_FALSE);
if (ret < 0){
        ERROR;
}
/* Allocate memory */
cookie = malloc(malloc_size);
if (cookie == NULL){
        ERROR;
}
/* Obtain cookie */
ret = sceHttpGetCookie(http://example.com/foobar/, cookie, NULL, malloc_size,
SCE_FALSE);
if (ret < 0){
        ERROR;
}
printf("cookie: %s\n", cookie);
```

See Also

`sceHttpAddCookie(),sceHttpSetCookieEnabled()`

sceHttpAddCookie

Add cookies

Definition

```
#include <libhttp.h>
int sceHttpAddCookie(
        const char *url,
        const char *cookie,
        SceSize cookieLength,
);
```

Arguments

url	Pointer to URL of the sender of *cookie* stored as ASCIIZ character string
cookie	Start address of cookie data sent from the server specified by *url* It is not necessary to be converted into ASCIIZ
cookieLength	Length of character string of cookie data stored in *cookie*

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	NULL is specified in *url* or *cookie*
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	Insufficient free memory space

Description

This function adds the cookie specified in *cookie* to the cookie list managed by libhttp. Since libhttp can handle the usual cookies specified with the response header, use this function to manually add cookies specified through javascript, etc.

The formats of character string specified for *cookie* are as follows. Double-quotation is not necessary.

```
NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure
```

Only `NAME` and `VALUE` are compulsory among the above formats. If specifying multiple cookies, specify ";" as a delimiter. For details on the standard specification of cookie, refer to the following website.

- HTTP State Management Mechanism
  http://tools.ietf.org/html/rfc6265

(The above reference destination has been confirmed as of March 11, 2015. Note that pages may have been subsequently moved or its contents modified.)

Notes

Note that if the target cookie is just referenced by `sceHttpSendRequest()` or `sceHttpGetCookie()` called from another thread when this function is called, the attempt to overwrite cookie will fail.

Note that the cookies must be handled in a way to prevent users from any disadvantages.

This function is not multithread safe.

See Also

`sceHttpGetCookie()`,`sceHttpSetCookieEnabled()`

sceHttpCookieFlush

Delete cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCookieFlush (
        void
);
```

Arguments

None

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function deletes all the cookies that libhttp stores.

See Also

`sceHttpSetCookieEnabled()`, `sceHttpCookieExport()`, `sceHttpCookieImport()`

sceHttpSetCookieTotalMaxSize

Set the maximum size for storing cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieTotalMaxSize (
        SceUInt32 size
);
```

Arguments

size	Maximum size of cookies to be stored by libhttp

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument

Description

This function sets the maximum size of cookie storage to be managed by libhttp. When the size of all the cookies managed by libhttp exceeds this maximum, cookies will be deleted automatically (starting with the oldest cookie).

Notes

81920 is set as the default.

When this function is called, all cookies stored by libhttp prior to the call will be deleted.

See Also

`sceHttpCookieFlush()`, `sceHttpSetCookieMaxSize()`, `sceHttpSetCookieMaxNum()`, `sceHttpSetCookieMaxNumPerDomain()`

sceHttpSetCookieMaxSize

Set the maximum size per cookie to store

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieMaxSize (
        SceUInt32 size
);
```

Arguments

size	Maximum size of 1 cookie to be stored by libhttp

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument

Description

This function sets the maximum size of 1 cookie for libhttp to store. An error will return when libhttp attempts to store a cookie larger than this maximum size.

Notes

4096 is set as the default.

When this function is called, all cookies stored by libhttp prior to the call will be deleted.

See Also

`sceHttpCookieFlush()`, `sceHttpSetCookieTotalMaxSize()`, `sceHttpSetCookieMaxNum()`, `sceHttpSetCookieMaxNumPerDomain()`

sceHttpSetCookieMaxNum

Set the maximum number of cookies to store

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieMaxNum (
        SceUInt32 size
);
```

Arguments

size	Maximum number of cookies to be stored by libhttp

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function sets the maximum number of cookies to be stored by libhttp. When the total number of cookies managed by libhttp exceeds this maximum, cookies will be automatically deleted (starting with the oldest cookie).

Notes

3000 is set as the default.

When this function is called, all cookies stored by libhttp prior to the call will be deleted.

See Also

`sceHttpCookieFlush()`, `sceHttpSetCookieTotalMaxSize()`, `sceHttpSetCookieMaxSize()`, `sceHttpSetCookieMaxNumPerDomain()`

sceHttpSetCookieMaxNumPerDomain

Set maximum number of cookies to store per domain

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieMaxNumPerDomain (
        SceUInt32 size
);
```

Arguments

size	Maximum number of cookies for libhttp to store per domain

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument

Description

This function sets the maximum number of cookies for libhttp to store per domain. When the total number of cookies stored in a domain exceeds this maximum, cookies will be automatically deleted (starting with the oldest cookie).

Notes

50 is set as the default.

When this function is called, all cookies stored by libhttp prior to the call will be deleted.

See Also

```
sceHttpCookieFlush(),sceHttpSetCookieTotalMaxSize(),
sceHttpSetCookieMaxSize(),sceHttpSetCookieMaxNum()
```

sceHttpGetCookieStats

Get cookie storage status

Definition

```
#include <libhttp.h>
int sceHttpGetCookieStats (
        SceHttpCookieStats* currentStat
);
```

Arguments

currentStat	Memory address for storing the cookie storage status

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function obtains the status of the cookie storage managed by libhttp in the memory pool. The current volume used for storing cookies and the number of cookies, as well as the maximum volume used and the maximum number of cookies stored after `sceHttpInit()`, can be obtained.

See Also

`sceHttpInit()`, `sceHttpCookieFlush()`, `sceHttpSetCookieTotalMaxSize()`, `sceHttpSetCookieMaxSize()`, `sceHttpSetCookieMaxNum()`, `sceHttpSetCookieMaxNumPerDomain()`

HTTP Header Parsing Functions

sceHttpParseStatusLine

Parse an HTTP status line

Definition

```
#include <libhttp.h>
SceInt32 sceHttpParseStatusLine (
        const char *statusLine,
        SceSize lineLen,
        SceInt32 *httpMajorVer,
        SceInt32 *httpMinorVer,
        SceInt32 *responseCode,
        const char **reasonPhrase,
        SceSize *phraseLen
);
```

Arguments

statusLine	Pointer to the status line string to parse
lineLen	Length of the string specified with *statusLine* (up to the CRLF)
httpMajorVer	Pointer to memory storing the HTTP major version For example, in the case of HTTP/0.9, a 0 would be stored, and in the case of HTTP/1.1, a 1 would be stored
httpMinorVer	Pointer to memory storing the HTTP minor version For example, in the case of HTTP/0.9, a 9 would be stored, and in the case of HTTP/1.1, a 1 would be stored
responseCode	Pointer to memory storing the value of the HTTP response code
reasonPhrase	Pointer to memory storing the pointer to the first character of the reason phrase in the string specified by the *statusLine*
phraseLen	Pointer to memory storing the length (not including the CRLF) of the response phrase string

Return Values

If the function completes normally, a positive value (the length of the status line, including the linefeed code) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_PARSE_HTTP_INVALID_RESPONSE	0x80432060	The format of the specified status line is invalid
SCE_HTTP_ERROR_PARSE_HTTP_INVALID_VALUE	0x804321fe	*httpMajorVer*, *httpMinorVer*, *responseCode*, *reasonPhrase* and *phraseLen* were set to NULL

Description

This function parses the status line string specified by *statusLine* and *lineLen*, stores the HTTP major version in *httpMajorVer*, the HTTP minor version in *httpMinortVer*, and the value of the response code in *responseCode*, and stores a pointer to the first character of the reason phrase in the string specified by *statusLine* in *reasonPhrase*, and the length of the reason phrase string in *phreaseLen*.

The *statusLine* string does not need to be NULL terminated, but it must include the CRLF. Note that, since a malloc is not performed in the API, the address specified by *reasonPhrase* is an address within the string specified by *statusLine*, and this string is not NULL-terminated.

if *lineLen* is larger than the length at which the CRLF is reached, the characters beyond the CRLF will be ignored.

Examples

```
SceInt32 ret, httpMajorVer, httpMinorVer, responseCode;
SceSize phraseLen;
SceChar8 *header = "HTTP/1.0 200 OK\r\n", *tmpBuf;
ret = sceHttpParseStatusLine(header, strlen(header), &httpMajorVer,
        &httpMinorVer,&responseCode, &reasonPhrase, &phraseLen);
if (ret < 0){
        goto error;
}
tmpBuf = malloc(phraseLen + 1);
if (tmpBuf == NULL){
        goto error;
}
memcpy(tmpBuf, reasonPhrase, phraseLen);
tmpBuf[phraseLen] = '\0';
printf("HTTP version  = %d.%d\n", httpMajorVer, httpMinorVer);
printf("response code = %d\n", responseCode);
printf("reason_phrase = %s\n", tmpBuf);
free(tmpBuf);
```

See Also

`sceHttpGetAllResponseHeaders()`

sceHttpParseResponseHeader

Parse an HTTP header

Definition

```
#include <libhttp.h>
SceInt32 sceHttpParseResponseHeader (
        const char *header,
        SceSize headerLen,
        const char *fieldName,
        const char **fieldValue,
        SceSize *valueLen
);
```

Arguments

header	Start address of HTTP header string
headerLen	Length of HTTP header string
fieldName	Start address of string representing the name of the header field to be obtained
fieldValue	Pointer to memory storing a pointer to the first character of the field value corresponding to the header field specified by *fieldName*
valueLen	Pointer to memory storing the length of the field value string

Return Values

If this function completes normally, a positive value (the length from header to the first linefeed code after the *fieldValue* string) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_PARSE_HTTP_INVALID_RESPONSE	0x80432060	The format of the specified HTTP header is invalid
SCE_HTTP_ERROR_PARSE_HTTP_INVALID_VALUE	0x804321fe	*fieldName*, *fieldValue* and *valueLen* were set to NULL
SCE_HTTP_ERROR_PARSE_HTTP_NOT_FOUND	0x80432025	The specified header field does not exist

Description

This function searches for the header field (for example "Date") specified by *fieldName* in the HTTP header specified by *header* and *headerLen* and, if this header field exists, stores the starting address of the string which represents the field's value in *fieldValue*, and stores the length of this string in *valueLen*. If the header field does not exist, SCE_HTTP_ERROR_PARSE_HTTP_NOT_FOUND is returned. If there are multiple corresponding header fields, only the first value is stored. Note that since a malloc is not performed in the API, the address specified by *fieldValue* is an address within the string specified by *header*, and this string is not NULL-terminated.

Examples

```
SceInt32 ret;
char *header ="HTTP Response Header", fieldName = "Date", fieldValue, *tmpBuf;
SceSize counter = 0, headerSize = strlen(header), valueLen;
while (counter < headerSize){
        ret = sceHttpParseResponseHeader(
              header + counter, headerSize - counter,
              fieldName, &fieldValue, &valueLen);
        if (ret < 0){
              goto error;
        }
        tmpBuf = malloc(valueLen + 1);
        if (tmpBuf == NULL){
              goto error;
        }
        memcpy(tmpBuf, fieldValue, valueLen);
        tmpBuf[value_len] = '\0';
        printf("[%s:] %s\n",argv[i], tmpBuf);
        free(tmpBuf);
        counter += ret;
}
```

See Also

```
sceHttpGetAllResponseHeaders()
```

URI Escape/Unescape Functions

sceHttpUriEscape

URI escape processing

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUriEscape (
        char *out,
        SceSize *require,
        SceSize prepare,
        const char*in
);
```

Arguments

out	Pointer to output byte stream
require	Pointer to memory to store the size of the output byte stream
prepare	Size of the memory provided for the output byte stream
in	Pointer to the input string

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	The number of bytes necessary for output exceeded the value specified by *prepare*
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	Both *out* and *require* were set to NULL

Description

This function performs URI escape processing on the string specified by *in*, outputs to the memory area specified by *out*, and stores the number of output bytes in *require*. If the number of output bytes exceeds the value specified by *prepare*, processing is terminated and an error is returned. If *out* is set to NULL, the size of the memory area required for output will be stored in *require* and can be obtained.

Examples

```
int ret;
SceSize mallocSize, outSize;
SceUChar8 *data = "target string";
char *out=NULL;
ret = sceHttpUriEscape(NULL, &mallocSize, 0, data);
if (ret < 0){
        printf("sceHttpUriEscape() returns %x.\n", ret);
        goto error;
}
out = (SceUChar8*)malloc(mallocSize);
if (out == NULL){
        printf("can't allocate memory\n");
        goto error;
}
ret = sceHttpUriEscape(out, &outSize, mallocSize, data);
```

sceHttpUriUnescape

URI unescape processing

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUriUnescape (
        char *out,
        SceSize *require,
        SceSize prepare,
        const char *in
);
```

Arguments

out	Pointer to output byte stream
require	Pointer to memory to store the size of the output byte stream
prepare	Size of the memory provided for the output byte stream
in	Pointer to the input string

Return Values

If this function completes normally, SCE_OK (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	The number of bytes necessary for output exceeded the value specified by *prepare*
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	Both *out* and *require* were set to NULL

Description

This function performs URI unescape processing on the string specified by *in*, outputs to the memory area specified by *out*, and stores the number of output bytes in *require*. If the number of output bytes exceeds the value specified by *prepare*, processing is terminated and an error is returned. If *out* is set to NULL, the size of the memory area required for output will be stored in *require* and can be obtained.

Examples

```
int  ret;
SceSize mallocSize, outSize;
char *data = "escaped%20string";
SceUChar8 *out=NULL;
ret = sceHttpUriUnescape(NULL, data, &mallocSize, 0);
if(ret < 0){
        printf("sceHttpUriUnescape() returns %x.\n", ret);
        goto error;
}
out = (SceUChar8*)malloc(mallocSize);
if (out == NULL){
        printf("can't allocate memory\n");
        goto error;
}
ret = sceHttpUriUnescape(out, data, &outSize, malloc_size);
```

URI Parsing and Building Functions

sceHttpUriParse

Parse a URI string

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUriParse (
        SceHttpUriElement *out,
        const char *srcUri,
        void *pool,
        SceSize *require,
        SceSize prepare
);
```

Arguments

out	Pointer to structure to store the URI elements after parsing
srcUri	Pointer to the URI to parse, which is stored as an ASCIZ string
pool	Pointer to memory buffer used to store the strings which result from parsing The starting addresses of the stored strings are specified by the respective members of *out*
require	Pointer to memory to store the size of the memory buffer required for parsing
prepare	Size of the memory provided in *pool*

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	The number of bytes necessary for output exceeded the value specified by *prepare*
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	All of *out*, *pool* and *require* have been set to NULL
`SCE_HTTP_ERROR_INVALID_URL`	0x80433060	The format of the URI specified by *srcUri* is invalid

Description

This function decomposes the string specified by *srcUri* into scheme, host name, port number, filepath, query, etc., stores these elements as an ASCIZ string using the memory specified by *pool*, and stores pointers to these respective element strings in the structure specified by *out*. The number of bytes of memory which were used is stored in *require*. If the number of bytes used exceeds the value specified by *prepare*, processing is terminated and an error is returned.

By calling this function with *out* or *pool* set to NULL, just the number of bytes of memory necessary for parsing can be obtained into *require* without the strings actually being copied into memory.

Examples

```
int ret;
void *pool;
SceSize mallocSize, useSize;
SceHttpUriElement    element;

ret = sceHttpUriParse(NULL, uri, NULL, &mallocSize, 0);
if (ret < 0){
        printf("sceHttpUriParse() returns %x.\n", ret);
        goto error;
}
pool = malloc(mallocSize);
if (pool == NULL){
        printf("can't allocate memory\n");
        ERR_STOP;
}
ret = sceHttpUriParse(&element, uri, pool, &useSize, mallocSize);
if (ret < 0){
        printf("sceHttpUriParse() returns %x.\n", ret);
        goto error;
}
```

sceHttpUriBuild

Create a URI string

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUriBuild (
        char *out,
        SceSize *require,
        SceSize prepare,
        const SceHttpUriElement *srcElement,
        SceUInt32 option
);
```

Arguments

out	Pointer to memory in which to store the URI which is created as an ASCIZ string
require	Pointer to memory to store the size of the memory necessary to store the URI strings
prepare	Size of the memory provided in *out*
srcElement	Pointer to the structure which stores the strings representing the respective elements of the created URI
option	Elements of the URI to be used (details below)

For *option*, specify the elements to include in the URI to be created with the bitwise OR of the following values.

Value	Description
SCE_HTTP_URI_BUILD_WITH_ALL	All elements
SCE_HTTP_URI_BUILD_WITH_SCHEME	Scheme
SCE_HTTP_URI_BUILD_WITH_HOSTNAME	Host name
SCE_HTTP_URI_BUILD_WITH_PORT	Port number
SCE_HTTP_URI_BUILD_WITH_PATH	Path
SCE_HTTP_URI_BUILD_WITH_USERNAME	Username
SCE_HTTP_URI_BUILD_WITH_PASSWORD	Password
SCE_HTTP_URI_BUILD_WITH_QUERY	Query
SCE_HTTP_URI_BUILD_WITH_FRAGMENT	Fragment

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_OUT_OF_MEMORY	0x80431022	The number of bytes necessary for output exceeded the value specified by *prepare*
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	Both *out* and *require* were set to NULL

Description

This function creates a URI string by assembling a scheme, host name, port number, filepath, query, etc. using the URI elements specified by *srcElement*, and stores it as an ASCIZ string in the memory specified by *out*. The number of bytes of memory which were used is stored in *require*. If the number of bytes used exceeds the value specified by *prepare*, processing is terminated and an error is returned.

By calling this function with *out* set to NULL, just the number of bytes of memory necessary for outputting the URI can be obtained into *require* without the strings actually being copied into memory.

Examples

```
int ret;
char *rebuildUri;
SceSize  mallocSize, useSize;
SceHttpUriElement    element;

memset(element, 0, sizeof(element));
element.scheme = "http";
element.hostname = "foo.com";

ret = sceHttpUriBuild(NULL, &element, &mallocSize, 0,
SCE_HTTP_URI_BUILD_WITH_ALL);
if (ret < 0){
        printf("sceHttpUriBuild() returns %x.\n", ret);
        goto error;
}
rebuildUri = (char*)malloc(malloc_size);
if (rebuildUri == NULL){
        printf("can't allocate memory\n");
        goto error;
}
ret = sceHttpUriBuild(rebuildUri, &element, &useSize, mallocSize,
        SCE_HTTP_URI_BUILD_WITH_ALL);
if (ret < 0){
        printf("sceHttpUriParse() returns %x.\n", ret);
        goto error;
}
printf("rebuild URI = %s\n", rebuildUri);
```

sceHttpUriSweepPath

Parse PATH string "../" and "./"

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUriSweepPath (
        char *dst,
        const char *src,
        SceSize srcSize
);
```

Arguments

dst	Pointer to memory storing parsed PATH string You must allocate memory based on the size specified by *srcSize*
src	Pointer to memory containing PATH string to parse The PATH string does not require a NULL terminator
srcSize	Size of the PATH specified in *src* Even when *src* has no NULL terminator, it is treated as if one is present, so *srcSize* should be specified as string length + 1. If this value is set to 0, the function will return without executing at all

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	Either *src* or *dst* were set to NULL

Description

If the PATH given by *src* and *srcSize* contains "./" or "../", it is parsed and stored as an ASCIZ string in *dst*. The PATH must begin with '/'. If the PATH does not meet these conditions, the *src* string is copied as is to *dst*, and is NULL terminated.

Examples

```
SceInt32 ret;
const char*src = "/foo/bar/../foo/./././../../../test/index.html";
SceChar8 *dst;
SceSize  srcSize;

srcSize = strlen(src) + 1;
dst = malloc(srcSize);
ret = sceHttpUriSweepPath(dst, src, srcSize);
if (ret < 0){
        printf("sceHttpUriSweepPath () returns %x.\n", ret);
        goto error;
}
printf("original path = %s sweeped path = %s\n",src, dst);
free(dst);
```

Notes

The processing performed by this function is automatically done for PATHs that are parsed using `sceHttpUriParse()`.

sceHttpUriMerge

Merge URL strings

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUriMerge (
        char*mergedUrl,
        const char*url,
        const char*relativeUri,
        SceSize *require,
        SceSize prepare,
        SceUInt32 option
);
```

Arguments

mergedUrl	Pointer to memory for storing URL string which merges *url* and *relativeUri*
url	Base URL string to merge. Must be NULL-terminated
relativeUri	Relative URI string to merge. Must be NULL-terminated
require	Size of memory that needs to be provided for *mergedUrl*
prepare	Size of memory that was provided for *mergedUrl*
option	Reserved. (Should be set to 0)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Number of bytes required exceeded value specified in *prepare*
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	Either *url* or *relativeUri* was set to NULL, or *option* was set to a non-zero value

Description

This function merges the URLs specified by *url* and *relativeUri* into an ASCIZ string and stores the result in *mergedUrl*. The number of bytes used for *mergedUrl*. is stored in *require*. If the number of bytes needed exceeds the value specified in *prepare*, the function terminates and an error is returned.

By calling this function with *mergedUrl* set to NULL, you can obtain the number of bytes needed in *require* without copying the actual string to memory.

If *relativeUri* specifies an absolute URL, then the string specified by *url* is ignored, and *relativeUri* is copied as is to *mergedUrl*.

Examples

```
SceInt32 ret;
const char *url = "http://foo.com/foo/index.html";
const char*relativeUri = "./default.html";
char*mergedUrl;
SceSize mallocSize;

ret = sceHttpUriMerge (NULL, url, relativeUri, &mallocSize, 0, 0);
if (ret < 0){
        printf("sceHttpUriMerge () returns %x.\n", ret);
        goto error;
}
mergedUrl = (char*)malloc(mallocSize);
ret = sceHttpUriMerge (mergedUrl, url, relativeUri, NULL, mallocSize, 0);
if (ret < 0){
        printf("sceHttpUriMerge () returns %x.\n", ret);
        free(mergedUrl);
        goto error;
}

printf("merged_url= %s \n",mergedUrl);
free(mergedUrl);
```

Notes

The parsing operation uses the memory it needs from the memory buffer specified in *`mergedUrl`*.

Since this is more memory than required by the URL string specified in *`mergedUrl`*, the application should perform a *`realloc`* or equivalent for the memory size after obtaining the appropriate length of the string.

In the current version, this function does not perform an operation equivalent to `sceHttpUriSweepPath()` internally.

HTTP Header Setting Functions

sceHttpAddRequestHeader

Add a request header

Definition

```
#include <libhttp.h>
SceInt32 sceHttpAddRequestHeader(
        SceInt32 id,
        const char *name,
        const char *value,
        SceInt32 mode
);
#define sceHttpInsertRequestHeader(id, name, value) \
        sceHttpAddRequestHeader(id, name, value, SCE_HTTP_HEADER_ADD)
#define sceHttpAddExtraHeader(id, name, value, mode) \
        sceHttpAddRequestHeader(id, name, value, mode)
```

Arguments

id	The relevant template settings, connection settings, or request object ID
name	The name portion of the header to be added
value	The value portion of the header to be added
mode	The behavior if the same header already exists (details below)

Specify one of the following values for *mode*.

Value	Description
SCE_HTTP_HEADER_OVERWRITE	Overwrites the existing header
SCE_HTTP_HEADER_ADD	Leaves the existing header, and adds the new one

Return Values

If this function completes normally, SCE_OK (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID is invalid
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	NULL is specified in *name* or the value specified in *mode* is invalid

Description

This function adds the new elements specified by *name* and *value* to the request headers of the template settings, connection settings, or request object specified by *id*. If an element with the same name already exists, then if SCE_HTTP_HEADER_OVERWRITE is specified in *mode*, the existing element will be overwritten, and if SCE_HTTP_HEADER_ADD is specified, the existing element will be left alone and the new one added. The *name* and *value* strings are copied within the library, so there is no need to store them after calling this function.

Notes

Note that when `SCE_HTTP_HEADER_ADD` is specified in *`mode`*, the library does not perform any checks as to whether it is acceptable, in terms of HTTP, to have multiple elements with that name.

Also, since the header resources that are added are released by `sceHttpDeleteRequest()`, `sceHttpDeleteConnection()`, or `sceHttpDeleteTemplate()`, this function does not need to be paired with `sceHttpRemoveRequestHeader()`.

Note that even when `SCE_HTTP_HEADER_OVERWRITE` is specified to *`mode`*, the following elements will not be overwritten.

- Content-Length
- Connection
- Proxy-Connection

Content-Length can be changed using `sceHttpSetRequestContentLength()`.

See Also

`sceHttpRemoveRequestHeader()`

sceHttpRemoveRequestHeader

Delete a request header

Definition

```
#include <libhttp.h>
int sceHttpRemoveRequestHeader(
        SceInt32 id,
        const char *name,
);
#define sceHttpDeleteHeader(id, name) sceHttpRemoveRequestHeader(id, name)
```

Arguments

id	The relevant template settings, connection settings, or request object ID
name	The name portion of the header to be deleted

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_NOT_FOUND`	0x80431025	The specified element does not exist in the headers

Description

This function deletes the element specified by *name* from the request headers of the template settings, connection settings, or request object specified by *id*. If there are multiple elements with the same name, all of these elements will be deleted.

See Also

`sceHttpAddRequestHeader()`

sceHttpSetRequestContentLength

Reset Content-Length

Definition

```
#include <libhttp.h>
int sceHttpSetRequestContentLength (
        SceInt32 id,
        SceULong64 contentLength
);
```

Arguments

id	ID of the target request object
contentLength	Content-Length value to set

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function stores the ID of the target request object to *id*, and sets the message body length to send to *contentLength* using a POST request.

See Also

`sceHttpCreateRequest()`

Non-Blocking Processing Functions

sceHttpSetNonblock

Set non-blocking mode

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetNonblock(
        SceInt32 id,
        SceBool enable
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object
enable	Setting of non-blocking mode (`SCE_FALSE`: disabled, `SCE_TRUE`: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function enables non-blocking mode for the template setting, connection setting, or request object specified by *id*. By default, it is disabled.

See Also

`sceHttpGetNonblock()`

sceHttpGetNonblock

Get non-blocking mode

Definition

```
#include <libhttp.h>
SceInt32 sceHttpGetNonblock(
        SceInt32 id,
        SceBool *enable
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object
enable	Setting of non-blocking mode (`SCE_FALSE`: disabled, `SCE_TRUE`: enabled)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function obtains the current non-blocking mode settings from the template settings, connection settings, or request object specified by *id*.

See Also

`sceHttpSetNonblock()`

sceHttpCreateEpoll

Create epoll handle

Definition

```
#include <libhttp.h>
SceInt32 sceHttpCreateEpoll (
        SceHttpEpollHandle *eh
);
```

Arguments

eh	Address that stores the created handle

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function creates an epoll handle required for obtaining the request status of non-blocking mode.

See Also

`sceHttpDestroyEpoll()`

sceHttpDestroyEpoll

Delete epoll handle

Definition

```
#include <libhttp.h>
SceInt32 sceHttpDestroyEpoll (
        SceHttpEpollHandle eh
);
```

Arguments

eh	epoll handle

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	An invalid value was specified for an argument
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized

Description

This function deletes the epoll handle specified by the argument.

See Also

`sceHttpCreateEpoll()`

sceHttpSetEpoll

Link epoll handle

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetEpoll (
        SceInt32 id,
        SceHttpEpollHandle eh,
        void *userArg
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object
eh	Linked epoll handle
userArg	Value freely set by the user The value set here is saved to the `SceHttpNBEvent` structure when an event occurs

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function links the epoll handle to the template settings, connection settings, or request object specified by *id*.

See Also

`sceHttpCreateEpoll(), sceHttpWaitRequest(), sceHttpUnsetEpoll()`

sceHttpUnsetEpoll

Release epoll handle link

Definition

```
#include <libhttp.h>
SceInt32 sceHttpUnsetEpoll (
        SceInt32 id
);
```

Arguments

id	ID of the relevant template settings, connection settings, or request object

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function releases the epoll handle link for the template settings, connection settings, or request object specified by *id*.

See Also

`sceHttpCreateEpoll(), sceHttpWaitRequest(), sceHttpSetEpoll()`

sceHttpWaitRequest, sceHttpWaitRequestCB

Get non-blocking request events

Definition

```
#include <libhttp.h>
SceInt32 sceHttpWaitRequest (
        SceHttpEpollHandle eh,
        SceHttpNBEvent* nbev,
        int maxevents,
        int timeout_us
);
SceInt32 sceHttpWaitRequestCB (
        SceHttpEpollHandle eh,
        SceHttpNBEvent* nbev,
        int maxevents,
        int timeout_us
);
```

Arguments

eh	epoll handle
nbev	Address that stores request events
maxevents	Maximum number of events that can be stored (1 or more)
timeout_us	Timeout (-1 (negative value): infinity) (microsecond)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

These functions get the events related to requests linked to the epoll handle. If there are multiple events that can be obtained, the number of events specified with *`maxevents`* is simultaneously obtained. If there is no event that can be obtained, these functions are blocked until the specified timeout time.

Notes

`sceHttpWaitRequestCB()` is a CB wait capable function. For details about using CB waiting, refer to the kernel features.

See Also

`sceHttpCreateEpoll(),sceHttpSetEpoll()`

sceHttpAbortWaitRequest

Abort waiting for getting non-blocking request event

Definition

```
#include <libhttp.h>
SceInt32 sceHttpAbortWaitRequest (
        SceHttpEpollHandle eh
);
```

Arguments

eh	epoll handle

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	An invalid value was specified for an argument

Description

This function cancels event waiting for the target epoll handle. The blocked `sceHttpWaitRequest()` is returned immediately.

See Also

`sceHttpWaitRequest()`

SSL Option Setting Functions

SCE_HTTPS_FLAG_*

Flags representing various check features for HTTPS communication

Definition

Value	(Number)	Description
`SCE_HTTPS_FLAG_SERVER_VERIFY`	0x01U	Whether or not to perform server verification upon HTTPS communication. Enabled by default. If a unique RootCA certificate is required for server verification, it can be specified with `sceHttpsLoadCert()`
`SCE_HTTPS_FLAG_CLIENT_VERIFY`	0x02U	Whether or not to perform client verification upon HTTPS communication. Disabled by default. Client certificate and secret key can be specified with `sceHttpsLoadCert()`
`SCE_HTTPS_FLAG_CN_CHECK`	0x04U	Whether or not to check if the common name field of the certificate sent from the server matches the host name of the connection target URL upon HTTPS communication. Enabled by default. This flag can only be enabled when `SCE_HTTPS_FLAG_SERVER_VERIFY` is on
`SCE_HTTPS_FLAG_NOT_AFTER_CHECK`	0x08U	Whether or not to check if the validity period of the certificate sent from the server is expired upon HTTPS communication. Enabled by default. This flag can only be enabled when `SCE_HTTPS_FLAG_SERVER_VERIFY` is on
`SCE_HTTPS_FLAG_NOT_BEFORE_CHECK`	0x10U	Whether or not to check if the validity period of the certificate sent from the server has started upon HTTPS communication. Enabled by default
`SCE_HTTPS_FLAG_KNOWN_CA_CHECK`	0x20U	Whether or not to check if the RootCA that issued the server certificate is in the locally-held RootCA upon HTTPS communication. Enabled by default. The RootCA certificate can be specified with `sceHttpsLoadCert()`

Description

These flags are for enabling/disabling various check features that are carried out within the library upon HTTPS communication. Set to enable features with `sceHttpsEnableOption()`, and set to disable features with `sceHttpsDisableOption()`.

See Also

`sceHttpsEnableOption()`, `sceHttpsDisableOption()`

sceHttpsEnableOption

Enable HTTPS communication checks

Definition

```
#include <libhttp.h>
int sceHttpsEnableOption(
        SceInt32 sslFlags
);
```

Arguments

sslFlags	Flags of check features to enable (bitwise OR of `SCE_HTTPS_FLAG_*`)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

The behavior of HTTPS communication can be set. To disable this setting, use the `sceHttpsDisableOption()` function.

See Also

`sceHttpsDisableOption()`

sceHttpsDisableOption

Disable HTTPS communication checks

Definition

```
#include <libhttp.h>
int sceHttpsDisableOption(
        SceInt32 sslFlags
);
```

Arguments

sslFlags	Flags of check features to disable (bitwise OR of `SCE_HTTPS_FLAG_*`)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

The behavior of HTTPS communication can be set. To enable this setting, use the `sceHttpsEnableOption()` function.

See Also

`sceHttpsEnableOption()`

sceHttpsEnableOption2

Enable HTTPS communication checks (ID specification)

Definition

```
#include <libhttp.h>
int sceHttpsEnableOption2(
        SceInt32 id,
        SceInt32 sslFlags
);
```

Arguments

id	ID of target template settings or connection settings
sslFlags	Flags of check features to enable (bitwise OR of `SCE_HTTPS_FLAG_*`)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

The behavior of HTTPS communication can be set for each template setting or connection setting. To disable this setting, use the `sceHttpsDisableOption2()` function.

See Also

`sceHttpsDisableOption2()`

sceHttpsDisableOption2

Disable HTTPS communication checks (ID specification)

Definition

```
#include <libhttp.h>
int sceHttpsDisableOption2(
        SceInt32 id,
        SceInt32 sslFlags
);
```

Arguments

id	ID of target template settings or connection settings
sslFlags	Flags of check features to disable (bitwise OR of `SCE_HTTPS_FLAG_*`)

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

The behavior of HTTPS communication can be set for each template setting or connection setting. To enable this setting, use the `sceHttpsEnableOption2()` function.

See Also

`sceHttpsEnableOption2()`

Error Acquisition Functions

sceHttpsGetSslError

Get detailed error code of SSL communication

Definition

```
#include <libhttp.h>
#include <libhttp_error.h>
int sceHttpsGetSslError(
        SceInt32 requestId,
        SceInt32 *errNum,
        SceUInt32 *detail
);
```

Arguments

requestId	Request ID for getting the error information of the SSL layer
errNum	Address for storing the error code
detail	Address for storing the detailed cause of the error code

Return Values

If the error code is acquired normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	Invalid *request_id* specified
`SCE_HTTP_ERROR_INVALID_VALUE`	0x804311fe	NULL is specified in *err_num* or *detail*

Description

This function obtains the detailed reason for `SCE_HTTP_ERROR_SSL` that occurred with `sceHttpSendRequest()` or `sceHttpReadData()`. When the request ID for which `SCE_HTTP_ERROR_SSL` occurred is specified to *requestId*, the error code and value indicating the reason are saved to the addresses specified with *errNum* and *detail*. The relationship between the error code stored in *errNum* and the detailed value stored in *detail* is as follows.

Insufficient Memory

Value	(Number)	Description
`SCE_HTTPS_ERROR_OUT_OF_MEMORY`	0x80435022	The memory that can be used by libssl is insufficient

0 will be stored in *detail*.

Invalid Server Certificate

Value	(Number)	Description
`SCE_HTTPS_ERROR_CERT`	0x80435060	Server certificate is invalid

The bitwise OR of the following flags will be stored in *`detail`*.

Value	(Number)	Description
`SCE_HTTPS_ERR_INTERNAL`	0x01U	Internal library error
`SCE_HTTPS_ERR_INVALID_CERT`	0x02U	Format of server certificate is invalid
`SCE_HTTPS_ERR_CN_CHECK`	0x04U	Common name check of server certificate failed
`SCE_HTTPS_ERR_NOT_AFTER_CHECK`	0x08U	Server certificate validity period expired
`SCE_HTTPS_ERR_NOT_BEFORE_CHECK`	0x10U	Before server certificate validity period
`SCE_HTTPS_ERR_UNKNOWN_CA`	0x20U	Does not have certificate of RootCA that issued server certificate

SSL Handshake Error

Value	(Number)	Description
`SCE_HTTPS_ERROR_HANDSHAKE`	0x80435061	SSL handshake error

0 will be stored in *`detail`*.

SSL Send/Receive Error

Value	(Number)	Description
`SCE_HTTPS_ERROR_IO`	0x80435062	SSL send/receive error

`detail` will be the network error code (network errno) that most recently occurred with libnet.

Internal Error

Value	(Number)	Description
`SCE_HTTPS_ERROR_INTERNAL`	0x80435063	libssl internal error

0 will be stored in *`detail`*.

HTTP PROXY Error

Value	(Number)	Description
`SCE_HTTPS_ERROR_PROXY`	0x80435064	HTTP PROXY server returned an error before SSL communication

0 will be stored in *`detail`*.

See Also

`sceHttpSendRequest()`, `sceHttpReadData()`

sceHttpGetLastErrno

Get the latest error code of request

Definition

```
#include <libhttp.h>
int sceHttpGetLastErrno (
        SceInt32 requestId,
        SceInt32 *errNum,
);
```

Arguments

requestId	Request ID to obtain the error code
errNum	Address to store the error code

Return Values

If the error code is acquired normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized
SCE_HTTP_ERROR_INVALID_ID	0x80431100	Value specified in *requestId* is invalid
SCE_HTTP_ERROR_INVALID_VALUE	0x804311fe	NULL is specified in *errNum*

Description

This function obtains the latest error code occurred through `sceHttpSendRequest()`. By specifying the request ID to *requestId*, the latest error code will be stored in the address specified in *errNum*. However, if `sceHttpSendRequest()` returns `SCE_HTTP_ERROR_PROXY`, an error code indicating the detailed reason will be stored in *errNum*.

See Also

`sceHttpSendRequest()`

RootCA Certificate Setting and Acquisition Functions

sceHttpsLoadCert

Register RootCA certificate referenced during HTTPS server authentication

Definition

```
#include <libhttp.h>
SceInt32 sceHttpsLoadCert(
        SceInt32 caCertNum,
        const SceHttpsData **caList,
        const SceHttpsData *cert,
        const SceHttpsData *privKey
);
```

Arguments

caCertNum	Number of elements of structure array indicating RootCA certificate
caList	Pointer to the structure array indicating RootCA certificate The number of elements is specified with *caCertNum*. Only PEM format certificates can be handled. If this argument is not needed, set it to NULL
cert	Pointer to the structure indicating the client certificate. Only PEM format certificates can be handled. If this argument is not needed, set it to NULL
privKey	Pointer to the structure indicating the private key. Only PEM format certificates can be handled. If this argument is not needed, set it to NULL

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized
`SCE_HTTP_ERROR_OUT_OF_MEMORY`	0x80431022	Insufficient free memory space

Description

This function registers the list of RootCA certificates referenced for server authentication. If a server certificate issued from a certificate authority that is not included among the default RootCA certificates of libhttp, add the RootCA certificate with this function.

See Also

`sceHttpsEnableOption(), sceHttpsDisableOption(), sceHttpsSetSslCallback()`

sceHttpsGetCaList

Get RootCA certificate array referenced during HTTPS server authentication

Definition

```
#include <libhttp.h>
int sceHttpsGetCaList(
        SceHttpsCaList* caList
);
```

Arguments

caList	Memory address for storing list of RootCA certificates

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function obtains the list of RootCA certificates referenced during server authentication. Information on the RootCA certificates included in the list can be obtained via the various functions of libssl.

See Also

`sceHttpsLoadCert()`, `sceSslGetSubjectName()`, `sceSslGetIssuerName()`, `sceSslGetNameEntryCount()`, `sceSslGetNameEntryInfo()`, `sceSslGetNotAfter()`, `sceSslGetNotBefore()`

sceHttpsFreeCaList

Release acquired RootCA certificate array

Definition

```
#include <libhttp.h>
int sceHttpsFreeCaList(
        SceHttpsCaList* caList
);
```

Arguments

caList	Memory address to which the list of RootCA certificates to be released is stored

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

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

Value	(Number)	Description
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function releases the list of RootCA certificates acquired with `sceHttpsGetCaList()`.

See Also

`sceHttpsGetCaList()`

Callback Setting Functions

sceHttpSetAuthInfoCallback

Set callback function for dynamic password input

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetAuthInfoCallback (
        SceInt32 id,
        SceHttpAuthInfoCallback cbfunc,
        void *userArg
);
```

Arguments

id	ID of target template setting, connection setting, or request object
cbfunc	Pointer to callback function
userArg	Any user-defined value. Used for argument to the callback function when it is called

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function sets the callback function that will be called when a username and password need to be entered. If no callback function is set, processing will continue as if the authentication failed.

If the URL contains a username and password when using either `sceHttpCreateConnectionWithURL()` or `sceHttpCreateRequestWithURL()`, those values will take priority.

The callback function can be returned to an unset state by specifying NULL for *cbfunc*.

Notes

The callback function that is set is executed in the context of the thread that called the HTTP communication processing function.

See Also

`sceHttpCreateConnectionWithURL()`, `sceHttpCreateRequestWithURL()`, `sceHttpSendRequest()`, `SceHttpAuthInfoCallback`

sceHttpSetRedirectCallback

Set callback function to be called when redirection occurs

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetRedirectCallback (
        SceInt32 id,
        SceHttpRedirectCallback cbfunc,
        void *userArg
);
```

Arguments

id	ID of target template setting, connection setting, or request object
cbfunc	Pointer to callback function
userArg	Any user-defined value. Used for argument to the callback function when it is called

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function is used to set a callback function to be called when a redirection occurs. If no callback function is set, whether to redirect will be decided based on the value set for `sceHttpSetAutoRedirect()`. The default is to perform redirection automatically.

The callback function can be returned to an unset state by specifying NULL for *cbfunc*.

Notes

The callback function that is set is executed in the context of the thread that called the HTTP communication processing function.

See Also

`sceHttpCreateConnectionWithURL()`, `sceHttpCreateRequestWithURL()`, `sceHttpSendRequest()`, `SceHttpRedirectCallback`

sceHttpSetCookieSendCallback

Set callback function to be called before sending cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieSendCallback (
        SceInt32 id,
        SceHttpCookieSendCallback cbfunc,
        void *userArg
);
```

Arguments

id	ID of target template setting, connection setting, or request object
cbfunc	Pointer to callback function
userArg	Any user-defined value. Used for argument to the callback function when it is called

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function is used to set a callback function to be called before a cookie is sent. If no callback function is set, cookies are processed automatically.

The callback function can be returned to an unset state by specifying NULL for *cbfunc*.

Notes

The callback function that is set is executed in the context of the thread that called the HTTP communication processing function.

See Also

`sceHttpCreateConnectionWithURL()`, `sceHttpCreateRequestWithURL()`, `sceHttpSendRequest()`, `SceHttpCookieSendCallback`

sceHttpSetCookieRecvCallback

Set callback function to be called before receiving cookies

Definition

```
#include <libhttp.h>
SceInt32 sceHttpSetCookieRecvCallback (
        SceInt32 id,
        SceHttpCookieRecvCallback cbfunc,
        void *userArg
);
```

Arguments

id	ID of target template setting, connection setting, or request object
cbfunc	Pointer to callback function
userArg	Any user-defined value. Used for argument to the callback function when it is called

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
`SCE_HTTP_ERROR_INVALID_ID`	0x80431100	The specified ID is invalid
`SCE_HTTP_ERROR_BEFORE_INIT`	0x80431001	The library is not initialized

Description

This function is used to set a callback function to be called before a cookie is received. If no callback function is set, cookies are processed automatically.

The callback function can be returned to an unset state by specifying NULL for *cbfunc*.

Notes

The callback function that is set is executed in the context of the thread that called the HTTP communication processing function.

See Also

`sceHttpCreateConnectionWithURL()`, `sceHttpCreateRequestWithURL()`, `sceHttpSendRequest()`, `SceHttpCookieRecvCallback`

sceHttpsSetSslCallback

Set callback function called during SSL communication

Definition

```
#include <libhttp.h>
int sceHttpsSetSslCallback(
        SceInt32 id,
        SceInt32 (*httpsCallback)(
              unsigned int, SceSslCert *const sslCert[], int, void* ),
        void *userArg
);
```

Arguments

id	ID of target template setting or connection setting
httpsCallback	Pointer to callback function
userArg	Any user-defined value. Used for argument to the callback function when it is called

Return Values

If this function completes normally, `SCE_OK` (=0) is returned.

If an error occurs, one of the following error codes (negative value) is returned.

Value	(Number)	Description
SCE_HTTP_ERROR_INVALID_ID	0x80431100	The specified ID is invalid
SCE_HTTP_ERROR_BEFORE_INIT	0x80431001	The library is not initialized

Description

This function sets the callback function called when SSL communication occurs. If no callback function has been set, processing is done taking the action specified with `sceHttpsEnableOption()` or `sceHttpsDisableOption()`.

The callback function can be returned to an unset state by specifying NULL for *httpsCallback*.

Notes

The set callback function is executed in the thread context in which an HTTP communication processing function was called.

See Also

`sceHttpsEnableOption()`, `sceHttpsDisableOption()`, `sceHttpSendRequest()`

Callback Function Prototypes

SceHttpAuthInfoCallback

Callback function for password obtaining during Basic/Digest authentication

Definition

```
#include <libhttp.h>
typedef enum SceHttpAuthType {
        SCE_HTTP_AUTH_BASIC,
        SCE_HTTP_AUTH_DIGEST
} SceHttpAuthType;

typedef SceInt32 (*SceHttpAuthInfoCallback)(
        SceInt32 request,
        SceHttpAuthType authType,
        const char *realm,
        char *username,
        char *password,
        SceBool needEntity,
        SceUChar8 **entityBody,
        SceSize *entitySize,
        SceBool *save,
        void *userArg
);
```

Arguments

request	ID of request object of caller
authType	Authentication type `SCE_HTTP_AUTH_BASIC`: Basic authentication `SCE_HTTP_AUTH_DIGEST`: Digest authentication
realm	String specified by the server to identify space to authenticate
username	Area for storing user name (up to `SCE_HTTP_USERNAME_MAX_SIZE` bytes)
password	Area for storing password (up to `SCE_HTTP_PASSWORD_MAX_SIZE` bytes)
needEntity	Whether *entityBody* is necessary
entityBody	Address of the memory for storing the pointer to *entityBody* It is necessary for the application to allocate this memory aside from *username* and *password* Normally need not be specified
entitySize	Pointer to the memory area to store the size of *entityBody* Normally need not be specified
save	Whether to save user name and password (set `SCE_FALSE`).
userArg	User-specified pointer specified with `sceHttpSetAuthInfoCallback()`

Return Values

Value	Description
0 or greater	Perform authentication
Negative value	Do not perform authentication

Description

This is a type definition of the callback function called during Basic/Digest authentication.

Basically, the callback function stores user name and password in *`username`* and *`password`*, and by returning a return value of 0 or greater, the header required for authentication is generated and sent. Authentication is not performed if a negative value is returned.

If saved user name and password are present, then those values are stored in *`username`* and *`password`* when the function is called.

Notes

When *`needEntity`* is set to `SCE_TRUE`, the *`entityBody`* to be sent must be specified. This occurs only when Digest authentication is performed with the POST method, and the server only accepts authentication headers of the type using *`entityBody`* information. In this case, the start address of the memory where all the data to be sent with the POST method are stored should be set in *`entityBody`*, and its size in *`entitySize`*. The specified memory must not be freed until either `sceHttpSendRequest()` has completed for the *`request`*, or until this function is called again using the same *`request`*.

See Also

`sceHttpSetAuthInfoCallback()`, `sceHttpSetAuthEnabled()`, `sceHttpGetAuthEnabled()`

SceHttpRedirectCallback

Callback function to be called when redirection occurs

Definition

```
#include <libhttp.h>
typedef SceInt32 (*SceHttpRedirectCallback)(
        SceInt32 request,
        SceInt32 statusCode,
        SceInt32 *method,
        const char *location,
        void *userArg
);
```

Arguments

request	ID of request object of caller
statusCode	Redirect status code (300 etc.)
method	Method specified by the request object
location	Location of redirect destination
userArg	User-specified pointer specified with `sceHttpSetRedirectCallback()`

Return Values

Value	Description
0 or greater	Perform redirection
Negative value	Do not perform redirection

Description

This is a type definition of the callback function called when redirection occurs.

You should return a value of 0 or greater if redirection is allowed based on *request*, *statusCode*, *method*, and *location*, and return a negative value if redirection is not allowed. *method* contains a value defined in `SceHttpMethods`. You can change the method to use after redirection by overwriting *method*. However, note that if you switch from the GET/HEAD method to the POST method, the request body will always be 0 bytes.

Notes

The memory for *location* is freed immediately after exiting the callback function. If you need to save *location*, you must copy it to a separate memory location while in the callback function. Also, if you are using libhttp with multithreading, this function must be multithread safe.

See Also

`sceHttpSetRedirectCallback(), sceHttpSetAutoRedirect(), sceHttpGetAutoRedirect()`

SceHttpCookieSendCallback

Callback function to be called before sending cookies

Definition

```
#include <libhttp.h>
typedef SceInt32 (*SceHttpCookieSendCallback)(
        SceInt32 request,
        const char *url,
        const char *cookieHeader,
        void *userArg
);
```

Arguments

request	ID of request object of caller
url	Destination URL
cookieHeader	Cookie string to send
userArg	User-defined pointer specified with `sceHttpSetCookieSendCallback()`

Return Values

Value	Description
0 or greater	Send cookies
Negative value	Do not send cookies

Description

This is a type definition of the callback function called when sending cookies.

When this function is set using `sceHttpSetCookieSendCallback()`, it will be called prior to sending cookies.

You should return a value of 0 or greater if sending cookies is allowed based on *request*, *url*, *cookieHeader*, and *userArg*, and return a negative value if sending cookies is not allowed. If a negative value is returned, libhttp will send the request to the server with the cookie header deleted.

See Also

`sceHttpSetCookieSendCallback()`, `sceHttpSetCookieEnabled()`, `sceHttpGetCookieEnabled()`

SceHttpCookieRecvCallback

Callback function to be called before receiving cookies

Definition

```
#include <libhttp.h>
typedef SceInt32 (*SceHttpCookieRecvCallback)(
        SceInt32 request,
        const char *url,
        const char *cookieHeader,
        SceSize headerLen,
        void *userArg
);
```

Arguments

request	ID of request object of caller
url	Receiving URL
cookieHeader	Cookie string to be received
headerLen	*cookieHeader* string length
userArg	User-specified pointer specified with `sceHttpSetCookieRecvCallback()`

Return Values

Value	Description
0 or greater	Receive cookies
Negative value	Do not receive cookies

Description

This is a type definition of the callback function called when receiving cookies.

When this function is set using `sceHttpSetCookieRecvCallback()`, it will be called prior to receiving cookies.

You should return a value of 0 or greater if receiving cookies is allowed based on *request*, *url*, *cookieHeader*, *headerLen* and *userArg*, and return a negative value if receiving cookies is not allowed. If a negative value is returned, libhttp will ignore the cookies without saving them.

See Also

`sceHttpSetCookieRecvCallback()`, `sceHttpSetCookieEnabled()`, `sceHttpGetCookieEnabled()`

SceHttpsCallback

Callback function called during SSL communication

Definition

```
#include <libhttp.h>
typedef int (*SceHttpsCallback)(
        unsigned int verifyErr,
        SceSslCert * const sslCert[],
        int certNum,
        void *userArg);
```

Arguments

verifyErr	Certificate verification error cause flag
sslCert	Pointer to array that indicates certificate chain
certNum	Number of certificates of certificate chain
userArg	Value specified with `sceHttpsSetSslCallback()`

Return Values

Value	Description
0 or greater	Enable SSL communication
Negative value	Disable SSL communication

Description

This is a type definition of the callback function called during SSL communication.

You should return a positive value when starting SSL communication based on the information of *verifyErr*, *sslCert*, and *certNum*, and return a negative value when stopping SSL communication. The bitwise OR of the flags that indicate errors related to server certificates will be passed to *verifyErr*, the same as the *detail* value when *errNum* is `SCE_HTTPS_ERROR_CERT` with `sceHttpsGetSslError()`. For details, refer to the `sceHttpsGetSslError()` section.

The information of the certificate chain of *sslCert* cannot be referenced directly. Obtain information using `sceSslGetSerialNumber()`, `sceSslGetSubjectName()`, `sceSslGetNameEntryCount()`, `sceSslGetNameEntryInfo()`, etc.

Since the memory of the certificate chain is released after return from the callback function, saving to a separate memory is required in the callback function if certificate information is required.

See Also

`sceHttpsSetSslCallback()`