NP Signaling Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceNpSignalingConnectionInfo

Union for connection information

Definition

```
#include <np.h>
typedef union SceNpSignalingConnectionInfo {
        SceUInt32 rtt;
        SceUInt32 bandwidth;
        SceNpId npId;
        struct {
              SceNetInAddr addr;
              SceNetInPort_t port;
              SceUChar8 padding[2];
        } address;
        SceUInt32 packet_loss;
} SceNpSignalingConnectionInfo;
```

Members

rtt	Round-trip time (microseconds)
bandwidth	Estimated bandwidth (bits/second)
npId	NP ID
address	IP address and port number
packet_loss	Packet loss percentage

Description

This datatype is for connection information.

A variable of this type is used when receiving information with `sceNpSignalingGetConnectionInfo()`.

See Also

`sceNpSignalingGetConnectionInfo()`

SceNpSignalingNetInfo

Structure for network information

Definition

```
#include <np.h>
typedef struct SceNpSignalingNetInfo {
        size_t size;
        SceNetInAddr local_addr;
        SceNetInAddr mapped_addr;
        int nat_status;
} SceNpSignalingNetInfo;
```

Members

size	Size of structure
local_addr	Local IP address
mapped_addr	External IP address
nat_status	NAT status type

Description

This datatype is for network information.

A variable of this type is used when receiving information with `sceNpSignalingGetLocalNetInfo()` or `sceNpSignalingGetPeerNetInfoResult()`.

See Also

`sceNpSignalingGetLocalNetInfo()`, `sceNpSignalingGetPeerNetInfoResult()`

SceNpSignalingMemoryInfo

Memory information

Definition

```
#include <np.h>
typedef struct SceNpSignalingMemoryInfo {
        SceSize totalMemSize;
        SceSize curMemUsage;
        SceSize maxMemUsage;
        SceUChar8 reserved[12];
} SceNpSignalingMemoryInfo;
```

Members

totalMemSize	Memory area size (bytes)
curMemUsage	Current memory usage (bytes)
maxMemUsage	Historic maximum memory usage (bytes)
reserved	Reserved area

Description

This structure indicates the NP Signaling library heap area memory information.

Specify it as an argument when executing `sceNpSignalingGetMemoryInfo()`.

See Also

`sceNpSignalingGetMemoryInfo()`

Initialization/Termination Functions

sceNpSignalingInit

Initialize library

Definition

```
#include <np.h>
int sceNpSignalingInit(
        SceSize poolSize,
        SceInt32 threadPriority,
        SceInt32 cpuAffinityMask,
        SceSize threadStackSize
);
```

Calling Conditions

Not multithread safe.

Arguments

poolSize	Memory pool size for the library in bytes
threadPriority	Priority of internal thread
cpuAffinityMask	CPU affinity of internal thread
threadStackSize	Stack size of internal thread in bytes

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_ALREADY_INITIALIZED	0x80552702	Already initialized. sceNpSignalingInit() may have already been executed. Check the calling order.

For a list of the NP error codes, refer to each reference document.

Description

This function initializes the NP Signaling Library. Heap memory will be allocated for the library at initialization and NP Signaling library internal threads will be created. Specify the memory pool size for the NP Signaling Library for *poolsize*. Specify the stack size and priority of the internal thread to *threadStackSize* and *threadPriority*, respectively. Specify the CPU affinity of the internal thread to *cpuAffinityMask*.

When 0 is specified for *threadStackSize* and *threadPriority*, the default values will be set.

See Also

`sceNpSignalingTerm()`

sceNpSignalingTerm

Terminate library

Definition

```
#include <np.h>
int sceNpSignalingTerm(
        void
);
```

Calling Conditions

Not multithread safe.

Arguments

None

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED`	0x80552701	Not initialized. `sceNpSignalingInit()` may not have been called yet. Check the calling order

Description

This function terminates the NP Signaling library.

See Also

`sceNpSignalingInit()`

Context Functions

sceNpSignalingCreateCtx

Create context

Definition

```
#include <np.h>
int sceNpSignalingCreateCtx(
        const SceNpId *npId,
        SceNpSignalingHandler handler,
        void *arg,
        SceUInt32 *ctxId
)
```

Calling Conditions

Multithread safe.

Arguments

npId	NP ID to associate with context (application's own NP ID)
handler	Callback function
arg	Address to be passed to callback function as an argument
ctxId	Area to store obtained context ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_OUT_OF_MEMORY	0x80552703	Could not allocate memory. This code indicates that memory could not be allocated from the NP Signaling library heap memory. It is possible that the heap memory size specified with `sceNpSignalingInit()` is too small.
SCE_NP_SIGNALING_ERROR_CTXID_NOT_AVAILABLE	0x80552704	Exceeded the maximum number of contexts that can be created. The application can have up to 8 contexts at one time. Check if there are unnecessary contexts that can be deleted.
SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *npId* or *ctxId*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function creates a context. To *`npId`*, pass the NP ID of the application obtained with `sceNpManagerGetNpId()`. If the function terminates normally, the context ID will be returned to *`ctxId`*.

Examples

```
extern SceNpSignalingHandler handler;      // User-defined callback function
extern void *arg;                          // User-defined pointer to pass to
callback function

int ret;
SceNpId npId;

ret = sceNpManagerGetNpId(&npId);
if ( ret < 0 ) {
        // Error handling
}

ret = sceNpSignalingCreateCtx(&npId, handler, arg, &ctxId);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

The maximum number of contexts that can exist at one time is 8 (`SCE_NP_SIGNALING_CTX_MAX`).

See Also

`sceNpSignalingDestroyCtx()`,`sceNpManagerGetNpId()`

sceNpSignalingDestroyCtx

Delete context

Definition

```
#include <np.h>
int sceNpSignalingDestroyCtx(
        SceUInt32 ctxId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. sceNpSignalingInit() has not been called, or sceNpSignalingTerm() has already been called. Check the calling order.

For a list of the NP error codes, refer to each reference document.

Description

This function deletes a context. Simultaneously, all ACTIVE and PENDING connections of the context will become INACTIVE.

Examples

```
extern SceUInt32 ctxId;     // Context ID currently in use

int ret;

ret = sceNpSignalingDestroyCtx(ctxId);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

This function also returns `SCE_OK` for calls for context IDs that do not exist.

See Also

`sceNpSignalingCreateCtx()`

sceNpSignalingSetCtxOpt

Set context options

Definition

```
#include <np.h>
int sceNpSignalingSetCtxOpt(
        SceUInt32 ctxId,
        int optname,
        int optval
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
optname	Option
optval	Value

Specify the following value to *optname*.

Value	(Number)	Description
SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE	1	Measure estimated bandwidth

When *optname* is SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE, specify one of the following values to *optval*.

Value	(Number)	Description
SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_ENABLE	1	Enable (default)
SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_DISABLE	0	Disable

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. sceNpSignalingInit() has not been called, or sceNpSignalingTerm() has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that an invalid value was specified for *optname* or *optval*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function sets the options of the specified context.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use

int ret;

ret = sceNpSignalingSetCtxOpt(ctxId,
        SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE,
        SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_DISABLE);
if ( ret < 0 ) {
        // Error handling
}
```

See Also

`sceNpSignalingGetCtxOpt()`

sceNpSignalingGetCtxOpt

Get context options

Definition

```
#include <np.h>
int sceNpSignalingGetCtxOpt(
        SceUInt32 ctxId,
        int optname,
        int *optval
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
optname	Option
optval	Area to store the value obtained

Specify the following value to *optname*.

Value	(Number)	Description
SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE	1	Measure estimated bandwidth

When *optname* is SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE, one of the following values will be set to *optval*.

Value	(Number)	Description
SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_ENABLE	1	Enabled (default)
SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_DISABLE	0	Disabled

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_ NOT_INITIALIZED	0x80552701	Not initialized. sceNpSignalingInit() has not been called, or sceNpSignalingTerm() has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_ CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_ INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that an invalid value was specified for *optname* or *optval*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains the value set to an option of the specified context. Specify the macro of the option to *`optname`*. The result will return to `*`*`optval`*.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use

int ret, optval;

ret = sceNpSignalingGetCtxOpt(ctxId,
         SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE,
         &optval);
if ( ret < 0 ) {
         // Error handling
}
```

See Also

`sceNpSignalingSetCtxOpt()`

Connection Functions

sceNpSignalingActivateConnection

Activate connection

Definition

```
#include <np.h>
int sceNpSignalingActivateConnection(
        SceUInt32 ctxId,
        const SceNpId *npId,
        SceUInt32 *connId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
npId	NP ID of communication target
connId	Area to store obtained connection ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_ NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_ OUT_OF_MEMORY	0x80552703	Could not allocate memory. This code indicates that memory could not be allocated from the NP Signaling library heap memory. It is possible that the heap memory size specified with `sceNpSignalingInit()` is too small.
SCE_NP_SIGNALING_ERROR_ CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_ CONNID_NOT_AVAILABLE	0x8055270d	Could not obtain the connection ID. This code indicates that the unique ID allocated to a connection could not be obtained. This error does not occur during normal usage.
SCE_NP_SIGNALING_ERROR_ INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *npId* or *connId*. Check the argument values.
SCE_NP_SIGNALING_ERROR_ OWN_NP_ID	0x80552716	Specified one's own NP ID. This code indicates that one's own NP ID was specified for *npId*. One's own NP ID cannot be specified for connection.

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_TOO_MANY_CONN`	0x80552717	Number of connections exceeded 64. Up to 64 connections (in the PENDING or ACTIVE state) can exist at one time.

For a list of the NP error codes, refer to each reference document.

Description

This function activates a connection with the specified NP ID. If the function terminates normally, an INACTIVE connection will transition to the PENDING state. Notification of transitions in connection state (ESTABLISHED events and DEAD events) will be carried out with callback functions.

Examples

```
extern SceUInt32 ctxId;       // Context ID currently in use
extern SceNpId npId;          // NP ID of communication target

int ret;
SceUInt32 connId;

ret = sceNpSignalingActivateConnection(ctxId, &npId, &connId);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

If `sceNpSignalingActivateConnection()` is called for a connection that is already ACTIVE, the call will be successful, and there will be another notification of the ESTABLISHED event.

See Also

`sceNpSignalingDeactivateConnection()`, `sceNpSignalingTerminateConnection()`

sceNpSignalingDeactivateConnection

Deactivate connection

Definition

```
#include <np.h>
int sceNpSignalingDeactivateConnection(
        SceUInt32 ctxId,
        SceUInt32 connId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
connId	Connection ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_ NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_ CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_ CONN_NOT_FOUND	0x8055270e	Specified connection does not exist. This code indicates that the connection with the ID *connId* either does not exist or is in the INACTIVE state. Make sure that the values specified to the arguments are correct.

For a list of the NP error codes, refer to each reference document.

Description

This function deactivates the connection of the specified connection ID.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceUInt32 connId;     // Connection ID currently in use

int ret;

ret = sceNpSignalingDeactivateConnection(ctxId, connId);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

After this function is called, there will be no more notification of transitions in the connection state (ESTABLISHED events and DEAD events).

See Also

`sceNpSignalingActivateConnection(),sceNpSignalingTerminateConnection()`

sceNpSignalingTerminateConnection

Terminate connection

Definition

```
#include <np.h>
int sceNpSignalingTerminateConnection(
        SceUInt32 ctxId,
        SceUInt32 connId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
connId	Connection ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. sceNpSignalingInit() has not been called, or sceNpSignalingTerm() has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_CONN_NOT_FOUND	0x8055270e	Specified connection does not exist. This code indicates that the connection with the ID *connId* either does not exist or is in the INACTIVE state. Make sure that the values specified to the arguments are correct.

For a list of the NP error codes, refer to each reference document.

Description

This function terminates the connection of the specified connection ID. If the function terminates normally, the connection will transition to the INACTIVE state.

If the connection is in the PENDING or ACTIVE state, the communication target will be notified of `SCE_NP_SIGNALING_ERROR_TERMINATED_BY_PEER` and a DEAD event. If the caller had the connection also activated in another context, `SCE_NP_SIGNALING_ERROR_TERMINATED_BY_MYSELF` and a DEAD event will be notified.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceUInt32 connId;     // Connection ID currently in use

int ret;

ret = sceNpSignalingTerminateConnection(ctxId, connId);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

After this function is called, there will be no more notification of transitions in the connection state (ESTABLISHED events and DEAD events).

See Also

`sceNpSignalingActivateConnection()`, `sceNpSignalingDeactivateConnection()`

Get Connection Status Functions

sceNpSignalingGetConnectionStatus

Get status of connection

Definition

```
#include <np.h>
int sceNpSignalingGetConnectionStatus(
        SceUInt32 ctxId,
        SceUInt32 connId,
        int *connStatus,
        SceNetInAddr *peerAddr,
        SceNetInPort_t *peerPort
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
connId	Connection ID
connStatus	Area to store obtained connection state
peerAddr	Area to store obtained IP address of peer
peerPort	Area to store obtained port number of peer (network byte order)

One of the following values will be set to *connStatus*.

Value	(Number)	Description
SCE_NP_SIGNALING_CONN_STATUS_INACTIVE	0	Connection (ID=*connId*) is INACTIVE
SCE_NP_SIGNALING_CONN_STATUS_PENDING	1	Connection (ID=*connId*) is PENDING
SCE_NP_SIGNALING_CONN_STATUS_ACTIVE	2	Connection (ID=*connId*) is ACTIVE

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_ NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_ CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_ INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *connStatus*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains the status of the connection specified with *`connId`*. *`peerAddr`* and *`peerPort`* are valid only when *`connStatus`* returns `SCE_NP_SIGNALING_CONN_STATUS_ACTIVE`.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceUInt32 connId;     // Connection ID currently in use

int ret;
int connStatus;
SceNetInAddr peerAddr;
SceNetInPort_t peerPort;

ret = sceNpSignalingGetConnectionStatus(ctxId, connId,
        &connStatus, &peerAddr, &peerPort);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

If a connection with a *`connId`* ID does not exist, This function sets `SCE_NP_SIGNALING_CONN_STATUS_INACTIVE` to *`connStatus`* and returns `SCE_OK`.

sceNpSignalingGetConnectionInfo

Get connection information

Definition

```
#include <np.h>
int sceNpSignalingGetConnectionInfo(
        SceUInt32 ctxId,
        SceUInt32 connId,
        int code,
        SceNpSignalingConnectionInfo *info
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
connId	Connection ID
code	Target information code
info	Area to store obtained information

One of the following values will be set to *code*.

Value	(Number)	Description
SCE_NP_SIGNALING_CONN_INFO_RTT	1	Round-trip time (microseconds)
SCE_NP_SIGNALING_CONN_INFO_BANDWIDTH	2	Estimated bandwidth (bits/second)
SCE_NP_SIGNALING_CONN_INFO_PEER_NPID	3	NP ID of peer
SCE_NP_SIGNALING_CONN_INFO_PEER_ADDRESS	4	IP address and port number of peer
SCE_NP_SIGNALING_CONN_INFO_MAPPED_ADDRESS	5	Your own IP address and port number as seen from your peer
SCE_NP_SIGNALING_CONN_INFO_PACKET_LOSS	6	Packet loss percentage

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_ NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_ CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_CONN_IN_PROGRESS`	0x80552714	Connection is in the progress of being established. The connection specified in *`connId`* is in the progress of being established, and the information specified with *`code`* cannot be obtained yet. Make sure that the values specified to the arguments are correct.
`SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT`	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *`info`*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains information of the connection specified with *`connId`*. To *`code`*, specify the code of the information to be obtained.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceUInt32 connId;     // Connection ID currently in use

int ret;
SceNpSignalingConnectionInfo info;

ret = sceNpSignalingGetConnectionInfo(ctxId, connId,
SCE_NP_SIGNALING_CONN_INFO_RTT, &info);
if ( ret < 0 ) {
        // Error handling
}
```

Notes

When `SCE_NP_SIGNALING_CONN_INFO_PEER_NPID` is specified to *`code`*, information can be obtained even when the connection specified in *`connId`* is in the progress of being established. When a value other than `SCE_NP_SIGNALING_CONN_INFO_PEER_NPID` is specified, the `SCE_NP_SIGNALING_ERROR_CONN_IN_PROGRESS` error will return if the connection specified in *`connId`* is not established (either the ESTABLISHED event must be returned or the PEER_ACTIVATED event must be notified).

Search Connection Functions

sceNpSignalingGetConnectionFromNpId

Get the connection ID from the NP ID

Definition

```
#include <np.h>
int sceNpSignalingGetConnectionFromNpId(
        SceUInt32 ctxId,
        const SceNpId *npId,
        SceUInt32 *connId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
npId	NP ID of communication target
connId	Area to store obtained connection ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_CONN_NOT_FOUND	0x8055270e	Specified connection does not exist. This code indicates that the connection with the ID *connId* either does not exist or is in the INACTIVE state. Make sure that the values specified to the arguments are correct.
SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *npId* or *connId*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains the connection ID corresponding to the specified NP ID.

A connection ID can be obtained regardless of the connection state (for example, even for a connection that has been activated by a peer but not yet locally; a deactivated connection; etc.). Thus, you cannot use the return value of this function to evaluate the connection state. To evaluate the connection state, use `sceNpSignalingGetConnectionStatus()` after obtaining the connection ID.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceNpId npId;         // NP ID of communication target

int ret;
SceUInt32 connId;

ret = sceNpSignalingGetConnectionFromNpId(ctxId, &npId, &connId);
if ( ret < 0 ) {
        // Error handling
}
```

sceNpSignalingGetConnectionFromPeerAddress

Get the connection ID from the address of a peer

Definition

```
#include <np.h>
int sceNpSignalingGetConnectionFromPeerAddress(
        SceUInt32 ctxId,
        SceNetInAddr peerAddr,
        SceNetInPort_t peerPort,
        SceUInt32 *connId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
peerAddr	IP address of peer
peerPort	Port number of peer (network byte order)
connId	Area to store obtained connection ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_CONN_NOT_FOUND	0x8055270e	Specified connection does not exist. This code indicates that a connection corresponding to the specified *peerAddr* and *peerPort* either does not exist or is in the INACTIVE state. Make sure that the values specified to the arguments are correct.
SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *connId*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains the connection ID for the connection corresponding to the specified IP address and port number of a peer. The connection ID of a connection in the ACTIVE state can be obtained.

Examples

```
extern SceUInt32 ctxId;                    // Context ID currently in use
extern SceNetInAddr peerAddr;              // IP address of peer
extern struct SceNetInPort_t peerPort;     // Port number of peer (network byte
order)

int ret;
SceUInt32 connId;

ret = sceNpSignalingGetConnectionFromPeerAddress(ctxId, peerAddr, peerPort,
&connId);
if ( ret < 0 ) {
        // Error handling
}
```

Get Network Information Functions

sceNpSignalingGetLocalNetInfo

Get own network information

Definition

```
#include <np.h>
int sceNpSignalingGetLocalNetInfo(
        SceUInt32 ctxId,
        SceNpSignalingNetInfo *info
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
info	Area to store obtained network information

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED`	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
`SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND`	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
`SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT`	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified to *info*, or an incorrect value was specified to the *size* member of the specified `SceNpSignalingNetInfo` structure. Check the values specified to the arguments and the *size* member of the `SceNpSignalingNetInfo` structure.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains one's own network information.

Call this function after storing the size of the structure to the *size* member of the network information structure `SceNpSignalingNetInfo`. If the external IP address is invalid, `SCE_NET_INADDR_ANY` will be stored to the *mapped_addr* member.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use

int ret;
SceNpSignalingNetInfo info;

memset(&info, 0, sizeof(info));
info.size = sizeof(info);
ret = sceNpSignalingGetLocalNetInfo(ctxId, &info);
if ( ret < 0 ) {
        // Error handling
}
```

sceNpSignalingGetPeerNetInfo

Request network information of communication target

Definition

```
#include <np.h>
int sceNpSignalingGetPeerNetInfo(
        SceUInt32 ctxId,
        const SceNpId *npId,
        SceUInt32 *reqId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
npId	NP ID of communication target
reqId	Area to store the request ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_OUT_OF_MEMORY	0x80552703	Could not allocate memory. This code indicates that memory could not be allocated from the NP Signaling library heap memory. It is possible that the heap memory size specified with `sceNpSignalingInit()` is too small.
SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_REQID_NOT_AVAILABLE	0x80552706	Could not obtain the request ID. This code indicates that the unique ID allocated to a request could not be obtained. This error does not occur during normal usage.
SCE_NP_SIGNALING_ERROR_PARSER_CREATE_FAILED	0x80552708	Error in creating a protocol message. This code indicates that an error occurred while creating a network information request message. This error does not occur during normal usage.
SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *npId* or *reqId*. Check the argument values.

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_OWN_NP_ID`	0x80552716	Specified one's own NP ID. This code indicates that one's own NP ID was specified for *`npId`*. To obtain one's own network information, use `sceNpSignalingGetLocalNetInfo()`.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains the network information of the communication target specified with NP ID.

This function only issues a request and upon normal termination, the request ID is returned to *`reqId`*. Success or failure in actually obtaining the network information is notified as an event by a callback function. When a success is notified, the result can be obtained by calling `sceNpSignalingGetPeerNetInfoResult()` with the request ID specified.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceNpId npId;         // NP ID of communication target

int ret;
SceUInt32 reqId;

ret = sceNpSignalingGetPeerNetInfo(ctxId, &npId, &reqId);
if ( ret < 0 ) {
        // Error handling
}
```

sceNpSignalingCancelPeerNetInfo

Cancel request of network information of communication target

Definition

```
#include <np.h>
int sceNpSignalingCancelPeerNetInfo(
        SceUInt32 ctxId,
        SceUInt32 reqId
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
reqId	Request ID

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. sceNpSignalingInit() has not been called, or sceNpSignalingTerm() has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
SCE_NP_SIGNALING_ERROR_REQ_NOT_FOUND	0x80552707	Specified request ID does not exist. This code indicates that the request with the ID *reqId* does not exist, or was cancelled, or its results were obtained. Make sure that the values specified to the arguments are correct.

For a list of the NP error codes, refer to each reference document.

Description

This function cancels the request to obtain the network information of the specified request ID.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceUInt32 reqId;      // Target request ID

int ret;

ret = sceNpSignalingCancelPeerNetInfo(ctxId, reqId);
if ( ret < 0 ) {
        // Error handling
}
```

sceNpSignalingGetPeerNetInfoResult

Get network information of communication target

Definition

```
#include <np.h>
int sceNpSignalingGetPeerNetInfoResult(
        SceUInt32 ctxId,
        SceUInt32 reqId,
        SceNpSignalingNetInfo *info
)
```

Calling Conditions

Multithread safe.

Arguments

ctxId	Context ID
reqId	Target request ID
info	Area to store obtained network information

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED`	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
`SCE_NP_SIGNALING_ERROR_CTX_NOT_FOUND`	0x80552705	Specified context does not exist. This code indicates that the context with the ID *ctxId* either was not created or was deleted. Make sure that the value specified to the argument is correct.
`SCE_NP_SIGNALING_ERROR_RESULT_NOT_FOUND`	0x80552713	Specified result does not exist. This code indicates that the request with the ID *reqId* does not exist, or was cancelled, or its transaction is currently in progress, or its results were obtained. Results cannot be obtained twice for the same request ID. Make sure that the values specified to the arguments are correct.
`SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT`	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified to *info*, or an incorrect value was specified to the *size* member of the specified `SceNpSignalingNetInfo` structure. Check the values specified to the arguments and the *size* member of the `SceNpSignalingNetInfo` structure.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains the result of the network information specified with the request ID.

Call this function after storing the size of the structure to the *`size`* member of the network information structure `SceNpSignalingNetInfo`. If the external IP address is invalid, `SCE_NET_INADDR_ANY` will be stored to the *`mapped_addr`* member.

Examples

```
extern SceUInt32 ctxId;      // Context ID currently in use
extern SceUInt32 reqId;      // Target request ID

int ret;
SceNpSignalingNetInfo info;

memset(&info, 0, sizeof(info));
info.size = sizeof(info);
ret = sceNpSignalingGetPeerNetInfoResult(ctxId, reqId, &info);
if ( ret < 0 ) {
        // Error handling
}
```

Utility Functions

sceNpSignalingGetMemoryInfo

Get memory information (for development)

Definition

```
#include <np.h>
int sceNpSignalingGetMemoryInfo(
        SceNpSignalingMemoryInfo *memInfo
);
```

Calling Conditions

Multithread safe.

Arguments

memInfo	Pointer to area storing the memory information

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. Note, however, that the application must not malfunction even if other error codes are returned.

Value	(Number)	Description
SCE_NP_SIGNALING_ERROR_NOT_INITIALIZED	0x80552701	Not initialized. `sceNpSignalingInit()` has not been called, or `sceNpSignalingTerm()` has already been called. Check the calling order.
SCE_NP_SIGNALING_ERROR_INVALID_ARGUMENT	0x80552715	Specified an invalid value for an argument. This code indicates that NULL was specified for *memInfo*. Check the argument values.

For a list of the NP error codes, refer to each reference document.

Description

This function obtains memory information regarding the heap area to be used by the NP Signaling library.

When this function is executed, the size of the heap area, the current memory usage volume, and the maximum memory usage in the past, can be obtained. Use this function in application development and check the memory size required by your application.

Callback Function

SceNpSignalingHandler

NP Signaling callback function

Definition

```
#include <np.h>
typedef void (SceNpSignalingHandler *){
        SceUInt32 ctxId,
        SceUInt32 subjectId,
        int event,
        int errorCode,
        void *arg
};
```

Arguments

ctxId	Context ID
subjectId	ID of target for which an event occurred
event	Type of event (SCE_NP_SIGNALING_EVENT_XXX)
errorCode	Error code
arg	Argument specified when context was created

One of the following values will be passed to *event* (further extension may be implemented in the future - program your application so that it does not malfunction when other values are passed).

Value	(Number)	Description
SCE_NP_SIGNALING_EVENT_DEAD	0	Connection (ID=*subjectId*) became INACTIVE
SCE_NP_SIGNALING_EVENT_ESTABLISHED	1	Connection (ID=*subjectId*) became ACTIVE
SCE_NP_SIGNALING_EVENT_NETINFO_ERROR	2	Request of network information (ID=*subjectId*) caused an error
SCE_NP_SIGNALING_EVENT_NETINFO_RESULT	3	Request of network information (ID=*subjectId*) was successful
SCE_NP_SIGNALING_EVENT_PEER_ACTIVATED	10	Connection (ID=*subjectId*) has been activated from the peer side
SCE_NP_SIGNALING_EVENT_PEER_DEACTIVATED	11	Connection (ID=*subjectId*) has been deactivated on the peer side
SCE_NP_SIGNALING_EVENT_MUTUAL_ACTIVATED	12	Connection (ID=*subjectId*) has been activated on both sides - local and peer

The main error codes passed to *`errorCode`* when *`event`* is `SCE_NP_SIGNALING_EVENT_DEAD` are shown below. (Note, however, that the application must not malfunction even if other error codes are returned.)

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_OUT_OF_MEMORY`	0x80552703	Could not allocate memory. This code indicates that memory could not be allocated from the NP Signaling library heap memory. It is possible that the heap memory size specified with `sceNpSignalingInit()` is too small.
`SCE_NP_SIGNALING_ERROR_PARSER_CREATE_FAILED`	0x80552708	Error in creating a protocol message. This code indicates that an error occurred while creating a signaling message. This error does not occur during normal usage.
`SCE_NP_SIGNALING_ERROR_PARSER_FAILED`	0x80552709	Error in parsing a protocol message. This code indicates that an error occurred while parsing a signaling message. This error does not occur during normal usage.
`SCE_NP_SIGNALING_ERROR_PEER_NOT_RESPONDING`	0x8055270c	No response from peer. This code indicates that there is no response from the peer to the signaling message. Either the peer signed out, or the connection with the PSN℠ server is disrupted (for either the peer or for oneself).
`SCE_NP_SIGNALING_ERROR_PEER_UNREACHABLE`	0x8055270f	Cannot reach peer. This code indicates that UDP connection tests with the peer failed and a connection could not be established. The NAT type of the peer (or oneself) or firewalls in the path could be the problem.
`SCE_NP_SIGNALING_ERROR_TERMINATED_BY_PEER`	0x80552710	Is in the INACTIVE state due to peer request. This code indicates that the peer did not activate the connection, or deactivated it, or terminated it.
`SCE_NP_SIGNALING_ERROR_TIMEOUT`	0x80552711	Request timed out. This code indicates that UDP keep-alive packets timed out. There may be network disruptions in the path to the peer.
`SCE_NP_SIGNALING_ERROR_TERMINATED_BY_MYSELF`	0x80552718	Is in the INACTIVE state due to own request. This code indicates that the connection was terminated by oneself.

The main error codes passed to *errorCode* when *event* is `SCE_NP_SIGNALING_EVENT_NETINFO_ERROR` are shown below. (Note, however, that the application must not malfunction even if other error codes are returned.)

Value	(Number)	Description
`SCE_NP_SIGNALING_ERROR_OUT_OF_MEMORY`	0x80552703	Could not allocate memory. This code indicates that memory could not be allocated from the NP Signaling library heap memory. It is possible that the heap memory size specified with `sceNpSignalingInit()` is too small.
`SCE_NP_SIGNALING_ERROR_PARSER_CREATE_FAILED`	0x80552708	Error in creating a protocol message. This code indicates that an error occurred while creating a network information request message. This error does not occur during normal usage.
`SCE_NP_SIGNALING_ERROR_PARSER_FAILED`	0x80552709	Error in parsing a protocol message. This code indicates that an error occurred while parsing a network information request message. This error does not occur during normal usage.
`SCE_NP_SIGNALING_ERROR_PEER_NOT_RESPONDING`	0x8055270c	No response from peer. This code indicates that there is no response from the peer to the network information request message. Either the peer signed out, or the connection with the PSN℠ server is disrupted (for either the peer or for oneself).

For a list of the NP error codes, refer to each reference document.

Description

This is the callback function for event notification from the NP Signaling library to an application. It is specified with `sceNpSignalingCreateCtx()`.

See Also

`sceNpSignalingCreateCtx()`

Constants

Maximum Number of Contexts

Maximum number of contexts that can be open simultaneously

Definition

Value	(Number)	Description
SCE_NP_SIGNALING_CTX_MAX	8	Maximum number of contexts that can be open at one time

Description

This constant represents the maximum number of contexts that can be open at one time.

Context Options

Context options

Definition

Value	(Number)	Description
`SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE`	1	Measure predicted bandwidth

When the option is `SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE`, the possible values are as follows. The value applies to all the connections for the specified context.

Value	(Number)	Description
`SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_ENABLE`	1	Enabled (default)
`SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_DISABLE`	0	Disabled

Description

These constants represent the types of context options.

Event Types

Types of event notified by callback function

Definition

Value	(Number)	Description
`SCE_NP_SIGNALING_EVENT_DEAD`	0	Connection became INACTIVE
`SCE_NP_SIGNALING_EVENT_ESTABLISHED`	1	Connection became ACTIVE
`SCE_NP_SIGNALING_EVENT_NETINFO_ERROR`	2	Request of network information caused an error
`SCE_NP_SIGNALING_EVENT_NETINFO_RESULT`	3	Request of network information was successful
`SCE_NP_SIGNALING_EVENT_PEER_ACTIVATED`	10	Connection has been activated from the peer side
`SCE_NP_SIGNALING_EVENT_PEER_DEACTIVATED`	11	Connection has been deactivated on the peer side
`SCE_NP_SIGNALING_EVENT_MUTUAL_ACTIVATED`	12	Connection has been activated on both sides - local and peer

Description

These constants represent the types of event notified by the callback function.

Events are expected to be added in the future.

Connection States

Current connection status

Definition

Value	(Number)	Description
SCE_NP_SIGNALING_CONN_STATUS_INACTIVE	0	INACTIVE state
SCE_NP_SIGNALING_CONN_STATUS_PENDING	1	PENDING state
SCE_NP_SIGNALING_CONN_STATUS_ACTIVE	2	ACTIVE state

Description

These constants represent the current state of the connection.

Connection Information Code

Connection information to obtain

Definition

Value	(Number)	Description
SCE_NP_SIGNALING_CONN_INFO_RTT	1	Round-trip time (microseconds)
SCE_NP_SIGNALING_CONN_INFO_BANDWIDTH	2	Estimated bandwidth (bits/second)
SCE_NP_SIGNALING_CONN_INFO_PEER_NPID	3	NP ID of peer
SCE_NP_SIGNALING_CONN_INFO_PEER_ADDRESS	4	IP address and port number of peer
SCE_NP_SIGNALING_CONN_INFO_MAPPED_ADDRESS	5	Your own IP address and port number as seen from your peer
SCE_NP_SIGNALING_CONN_INFO_PACKET_LOSS	6	Packet loss percentage

Description

This constant represents the type of connection information to obtain.

NAT Status Type

NAT status type

Definition

Value	(Number)	Description
`SCE_NP_SIGNALING_NETINFO_NAT_STATUS_UNKNOWN`	0	Unknown
`SCE_NP_SIGNALING_NETINFO_NAT_STATUS_TYPE1`	1	Type 1
`SCE_NP_SIGNALING_NETINFO_NAT_STATUS_TYPE2`	2	Type 2
`SCE_NP_SIGNALING_NETINFO_NAT_STATUS_TYPE3`	3	Type 3

Description

These constants represent the NAT status type in the network information.

Default Memory Pool Size

Default memory pool size

Definition

Value	(Number)	Description
`SCE_NP_SIGNALING_POOLSIZE_DEFAULT`	128*1024	Default memory pool size for the library

Description

This constant is the size to be used when 0 is specified for *`poolSize`* with `sceNpSignalingInit()`.

Default Internal Thread Priority

Default internal thread priority

Definition

Macro	Value	Description
`SCE_NP_SIGNALING_THREAD_PRIORITY_DEFAULT`	`SCE_KERNEL_DEFAULT_PRIORITY_USER`	Internal thread default priority

Description

This constant is the priority to be used when 0 is specified for *`threadPriority`* with `sceNpSignalingInit()`.

Default Internal Thread Stack Size

Default internal thread stack size

Definition

Value	(Number)	Description
`SCE_NP_SIGNALING_THREAD_STACK_SIZE_DEFAULT`	16 * 1024	Internal thread default stack size

Description

This constant is the priority to be used when 0 is specified for *`threadStackSize`* with `sceNpSignalingInit()`.