NP Signaling Library Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

1 Library Overview

Purpose and Characteristics

The NP Signaling library is a library for performing the communication processing required for P2P communication such as for online games and chat on PSN℠. With the NP Signaling library and UDPP2P protocol provided by libnet, P2P communication environments compatible with the various user network environments are provided.

By further combining with librudp, it will be possible to divide usage based on the purposes of the various communication types required by P2P communication, such as absence/presence of reliability and packet delivery order guarantees.

Main Features

The main functions provided by the NP Signaling library are as follows.

- Function for getting the IP address and UDP port number from the NP ID of the communication target
- NAT Traversal function for communicating across a NAT router
- Function for getting one's own network information and that of the communication target

Embedding into a Program

Include np.h in the source program. Various header files will be automatically included as well.

Load also the PRX module in the program as follows.

```
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_NP_SIGNALING) != SCE_OK ) {
    // Error processing
}
```

Upon building the program, link libSceNpSignaling_stub.a.

Sample Program

The following program is provided as a NP Signaling library sample program for reference purposes.

sample_code/network/api_np/np_signaling/signaling_rudp_sample/

This sample replaces the signaling feature of the NP Matching 2 library sample signaling_rudp_sample with the NP Signaling feature. This sample uses the NP Signaling library and librudp.

Reference Materials

Refer to the following document for an overview of the PSN℠ functionalities.

- PSN℠ Overview

Refer to the following documents regarding the NP library, which is commonly required when using the PSN℠ functionalities.

- NP Library Overview
- NP Library Reference

Refer to the following document regarding Network Check Dialog for switching service states of the NP library.

- Network Overview

Refer to the following documents regarding the P2P exclusive protocols.

- Network Overview
- libnet Overview
- libnet Reference

Refer to the following documents regarding librudp, which supports reliable data transfer (RUDP).

- librudp Overview
- librudp Reference

2 Usage Procedure

Basic Processing Procedure

(1) Load PRX

Load PRX by calling `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP_SIGNALING` specified for the module ID.

(2) Initialize NP Signaling library

Call `sceNpSignalingInit()` to initialize the NP Signaling library. At this time, specify the internal thread stack size and priority. When 0 is specified for these values, the default values will be set.

The following processing is performed in the initialization of the NP Signaling library.

(1) Creation of internal threads

(2) Creation of the heap area used by the NP Signaling library

If initialization with `sceNpSignalingInit()` is successful, 0 is returned.

(3) Create a signaling context

Call `sceNpSignalingCreateCtx()` to create a signaling context. At the same time, set a callback function for receiving an event notification via an argument.

If `sceNpSignalingCreateCtx()` has successfully created a context, the context ID is stored in an argument, and 0 is returned.

(4) Activate a connection

Call `sceNpSignalingActivateConnection()` and specify the context ID and the NP ID of the communication target to activate a connection. If `sceNpSignalingActivateConnection()` has successfully started processing for establishing a connection, the connection ID is stored in an argument, and 0 is returned.

(5) Wait for an ESTABLISHED event

After activation, wait for an ESTABLISHED event using a callback function. When a connection is established, an ESTABLISHED event will be notified for the associated connection ID. If a DEAD event is notified, error handling will be performed for the failed connection attempt.

(6) Get IP address and port number

After the connection is established, call `sceNpSignalingGetConnectionStatus()` and specify the context ID and connection ID to get the IP address and port number of the communication target. If `sceNpSignalingGetConnectionStatus()` successfully gets the status, the current status, IP address, and port number are saved in arguments, and 0 is returned.

(7) Create a UDPP2P protocol socket and bind the address

Call `sceNetSocket()` and specify `SCE_NET_AF_INET` for the domain of an argument and `SCE_NET_SOCK_DGRAM_P2P` for the type to create a UDPP2P protocol socket. Then, call `sceNetBind()` to bind the address to the socket. Specify the `SceNetSockaddrIn` structure for the local address of an argument, the local IP address for the IP address of the `SceNetSockaddrIn` structure and 3658 (`SCE_NP_PORT`) for the port number. For the virtual port number, specify an arbitrary port for use by the application.

(8) Send and receive packets

To send packets, either call `sceNetConnect()` to bind the destination address and then call `sceNetSend()`, or specify the destination address and call `sceNetSendto()`. Specify the destination address with the `SceNetSockaddrIn` structure. For the IP address and port number, specify the values that were obtained from `sceNpSignalingGetConnectionStatus()`. For the virtual port number, specify an arbitrary port for use by the application.

To receive packets, call `sceNetRecv()` or `sceNetRecvfrom()`. For the sender address, reference the value stored in the `SceNetSockaddrIn` structure.

(9) Terminate the connection

When communication with the relevant NP ID is finished, call `sceNpSignalingTerminateConnection()` and specify the context ID and the NP ID of the communication target to deactivate the connection. If `sceNpSignalingTerminateConnection()` successfully deactivates the connection, 0 is returned.

(10) Destroy the signaling context

When use of the NP Signaling library is finished, destroy the signaling context that was created. Call `sceNpSignalingDestroyCtx()` and specify the context ID.

If `sceNpSignalingDestroyCtx()` successfully destroys the context, 0 is returned.

(11) Terminate the NP Signaling library

Call `sceNpSignalingTerm()` to terminate the NP Signaling library. At this time, all remaining connections will be disconnected, and all contexts will be deleted.

(12) Unload PRX

Unload PRX by calling `sceSysmoduleUnloadModule()` with `SCE_SYSMODULE_NP_SIGNALING` specified for the module ID.

List of Functions

Functions for Creating, Deleting, and Setting a Context

Function Name	Description
`sceNpSignalingCreateCtx()`	Creates signaling context
`sceNpSignalingDestroyCtx()`	Destroys signaling context
`sceNpSignalingSetCtxOpt()`	Sets context options
`sceNpSignalingGetCtxOpt()`	Gets context options

Functions for Performing Connection Operations

Function Name	Description
`sceNpSignalingActivateConnection()`	Activates connection
`sceNpSignalingDeactivateConnection()`	Deactivates connection
`sceNpSignalingTerminateConnection()`	Terminates connection

Functions for Obtaining Connection Status

Function Name	Description
`sceNpSignalingGetConnectionStatus()`	Gets connection state
`sceNpSignalingGetConnectionInfo()`	Gets information of connection

Functions to Search for a Connection

Function Name	Description
`sceNpSignalingGetConnectionFromNpId()`	Gets the connection ID from the NP ID
`sceNpSignalingGetConnectionFromPeerAddress()`	Gets the connection ID from the IP address and port number of a peer

Functions for Obtaining Network Information

Function Name	Description
`sceNpSignalingGetLocalNetInfo()`	Gets own network information
`sceNpSignalingGetPeerNetInfo()`	Requests network information of communication target
`sceNpSignalingCancelPeerNetInfo()`	Cancels request of network information of communication target
`sceNpSignalingGetPeerNetInfoResult()`	Gets network information of communication target

3 Description of Operation

Signaling Processing Flow

(1) Connection

The NP Signaling library defines P2P communication between two given NP IDs as a **connection**. An application calls the NP Signaling library to establish a connection before performing P2P communication.

(2) Connection ID

The NP Signaling library assigns a **connection ID** to each connection. The lifetime of the connection ID is the same as the lifetime of the corresponding connection (from the time that connection processing begins until it ends). The connection ID, which is a locally unique value, is different at the two endpoints of the connection. Also, the values are temporarily assigned, and the same connection IDs are not always assigned to the same set of NP IDs.

(3) Activation

A request to establish a connection that is issued to the NP Signaling library is called **activation**. The application performs activation for the required connection before starting communication. A new connection ID is assigned to a connection that was activated from the INACTIVE state, and the state of that connection transitions to PENDING. The application waits for the connection to be established (ACTIVE state) for the assigned connection ID.

The NP Signaling library uses STUN to get IP address and port number information, exchanges address information with the communication target, and performs communication via UDP packets. The state becomes ACTIVE at the point when communication is verified, and then the connection is established. When the connection is established, the NP Signaling library reports an ESTABLISHED event to the application.

Activation is performed at both endpoints of the connection. However, even when activation is only performed at endpoint A to endpoint B, the connection state becomes ACTIVE at endpoint A and the ESTABLISHED event will be notified. At this point, a PEER_ACTIVATED event is notified to endpoint B. If activation is then performed at endpoint B, the connection state will be ACTIVE at endpoint B and the ESTABLISHED event will be notified. In addition, an MUTUAL_ACTIVATED event, which indicates that activation was performed at both endpoints, will be notified (see Figure 2 and Figure 3).

If activation is not also performed at endpoint B within a certain period of time after it is performed at endpoint A, the connection times out, the connection state is returned to INACTIVE, and a DEAD event is notified to endpoint A. Currently, the timeout interval is set to 1 minute.

In addition, if the attempt to establish the connection fails, or if processing is terminated before the connection is established for some reason, the NP Signaling library reports a DEAD event to the application together with an error code.

When using multiple contexts, the same connection is used for the same NP IDs. Therefore, if a connection is in the ACTIVE state due to activation from another context, an ESTABLISHED event will be immediately notified for contexts that later call an activation.

(4) Getting connection status

When the connection state is ACTIVE, the application can get the IP address and port number of the connection target's endpoint from the NP Signaling library. This information is used by the UDPP2P protocol for communication.

(5) Connection: Keep-Alive

When the connection state is ACTIVE, a keep-alive packet is exchanged every 10 seconds in order to maintain the connection, regardless of whether or not P2P communication is attempted by the application. If a keep-alive packet does not arrive for over a minute, the connection will be judged to have been terminated, and a DEAD event will be notified.

(6) Disconnection

To disconnect a connection, in other words to place it in the INACTIVE state, call `sceNpSignalingTerminateConnection()`. If the connection at that other endpoint is PENDING or ACTIVE, the communication target will receive the error code `SCE_NP_SIGNALING_ERROR_TERMINATED_BY_PEER` and be notified of a DEAD event (see Figure 4).

When using multiple contexts, if a disconnection is performed from a context, disconnection processing will be executed even if activated from another context. For the other contexts, `SCE_NP_SIGNALING_ERROR_TERMINATED_BY_MYSELF` and a DEAD event will be notified.

(7) Deactivation

The reporting of the end of a connection to the NP Signaling library is called **deactivation**. When deactivation is executed at endpoint A to endpoint B, a PEER_DEACTIVATED event is notified to endpoint B.

A connection that was deactivated at endpoint A does not immediately transition to the INACTIVE state at endpoint B, but remains ACTIVE for one minute. If the connection is not activated again within that period at endpoint A, a timeout occurs and the state transitions to INACTIVE (see Figure 5).

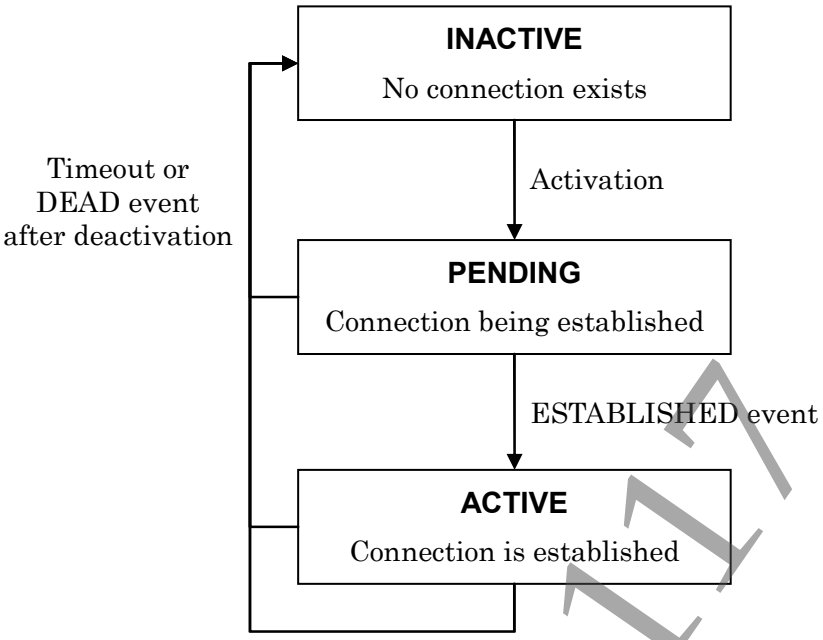
When using multiple contexts, deactivation processing will be delayed and the connection will be maintained until all activated contexts have been deactivated.

Connection Status Transition Diagram

The following are the possible states of a connection.

- INACTIVE
  State when no connection exists. No connection ID is assigned either.
- PENDING
  State when processing is in progress to establish a connection.
- ACTIVE
  State when a connection is established.

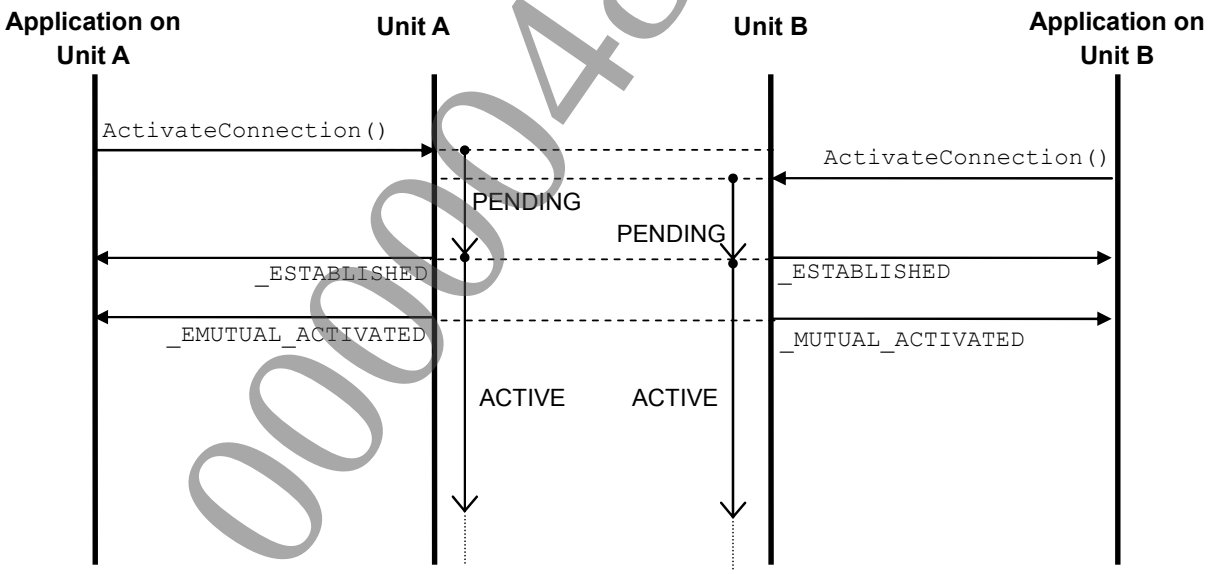
Figure 1 Connection Status Transition Diagram



Connection Status Transition Sequence

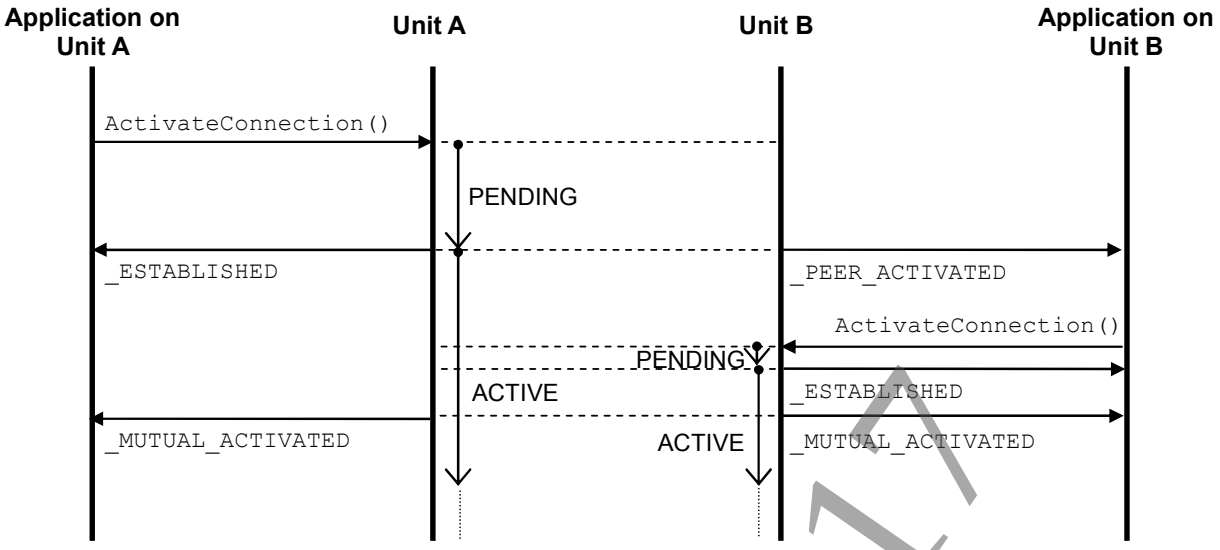
Regarding the connection between endpoint A and endpoint B, API executions, event notifications, and connection statuses can be summarized as follows.

Figure 2 When Activated Roughly at the Same Time



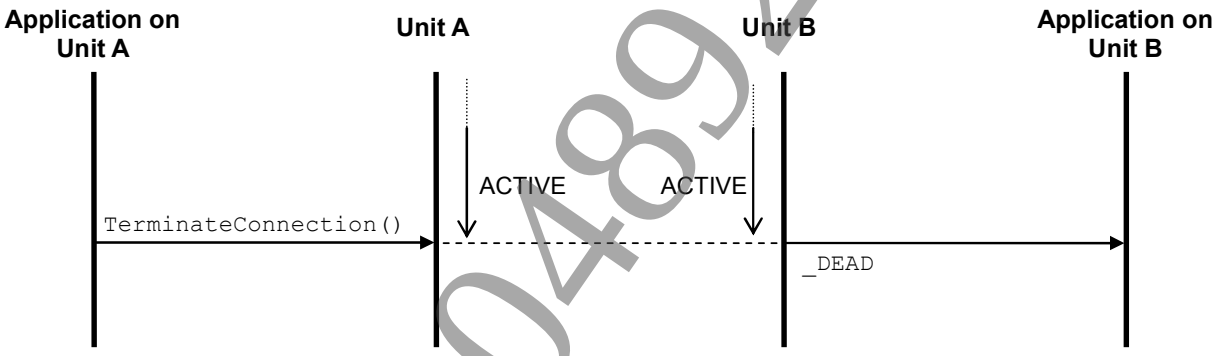
*The `sceNpSignaling` function prefix and `SCE_NP_SIGNALING_EVENT` macro prefix are omitted here.

Figure 3 When Connection is First Activated at Endpoint A



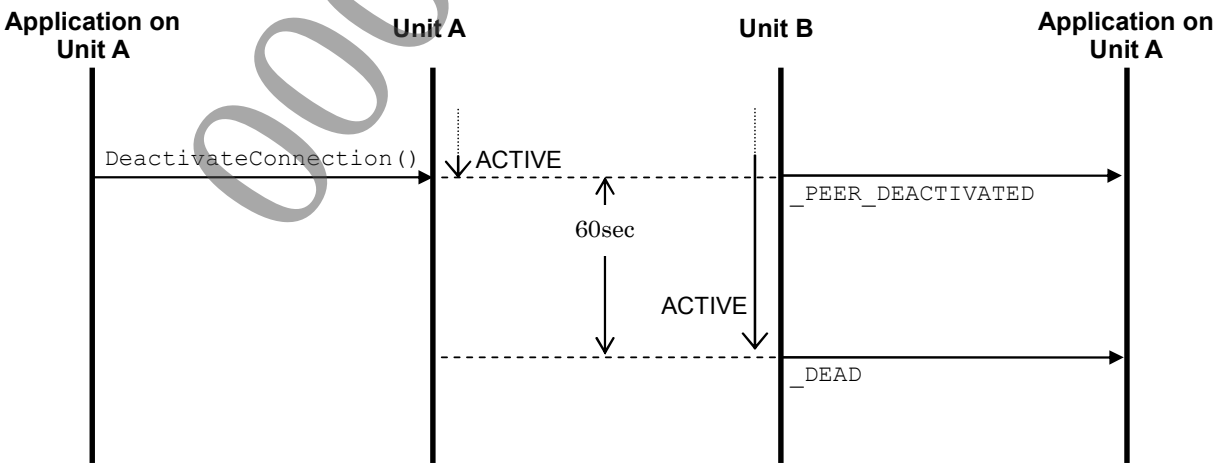
*The `sceNpSignaling` function prefix and `SCE_NP_SIGNALING_EVENT` macro prefix are omitted here.

Figure 4 When Connection is Terminated at Endpoint A



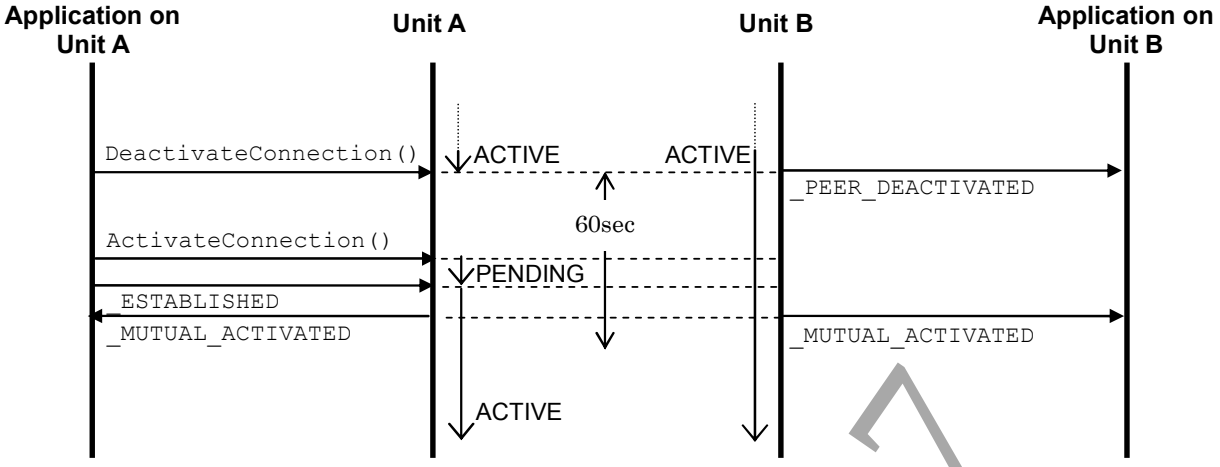
*The `sceNpSignaling` function prefix and `SCE_NP_SIGNALING_EVENT` macro prefix are omitted here.

Figure 5 When Connection is Deactivated at Endpoint A



*The `sceNpSignaling` function prefix and `SCE_NP_SIGNALING_EVENT` macro prefix are omitted here

Figure 6 When Connection is Deactivated and then Reactivated at Endpoint A



*The `sceNpSignaling` function prefix and `SCE_NP_SIGNALING_EVENT` macro prefix are omitted here.

Connection Information

Information regarding a connection in the ACTIVE state can be obtained with `sceNpSignalingGetConnectionInfo()`.

Round-trip Time (SCE_NP_SIGNALING_CONN_INFO_RTT)

This is the time it takes a UDPP2P packet to make a round-trip to a connected peer and back (in microseconds). Measurement is made upon the exchange of the keep-alive packet, which takes place every 10 seconds. The average of the last 6 measurements is returned.

Estimated Bandwidth (SCE_NP_SIGNALING_CONN_INFO_BANDWIDTH)

This is an estimate of the minimum bandwidth (in bits/second) of the route from the host to a peer. It is calculated as the difference in arrival times between 2 consecutively sent packets using a method called the packet pair technique. 10 measurements are consecutively made immediately after a connection is established, and a measurement is made every minute thereafter. The median of the last 10 measurements is returned.

Depending on the influences from traffic of other communications, the estimate value derived using this packet pair technique may be smaller than the actual bandwidth; moreover, it may not always be possible to use this obtained bandwidth. Only take this value as a guide indicative of the line quality.

Note that when a band such as ADSL must go through an asymmetric line, there is a great difference between the values of the bands, as measured from the host to the peer and measured from the peer to the host. For example, between unit A - connected to an optical fiber, and unit B - connected to ADSL, the value that can be obtained at unit B (B -> A) is determined by the upstream ADSL broadband and will be smaller than the value that can be obtained at unit A (A -> B).

The context option `SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE` can be set with `sceNpSignalingSetCtxOpt()` to specify whether to make measurements for estimating the bandwidth. The default is `SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_ENABLE` (enabled).

To avoid affecting game traffic with the use of measurement packets, disable this context option (set it to `SCE_NP_SIGNALING_CTX_OPT_BANDWIDTH_PROBE_DISABLE`).

Note
The value for the **Network - Network Emulation** bandwidth restriction in the **★Debug Settings** function of Development Kits and Testing Kits will not be accurately reflected in the estimated bandwidth.

NP ID of the Peer (SCE_NP_SIGNALING_CONN_INFO_PEER_NPID)

This is the NP ID of the connected peer (communication target).

IP Address and Port Number of the Peer (SCE_NP_SIGNALING_CONN_INFO_PEER_ADDRESS)

This is the IP address and port number of the connected peer (communication target). The same values can be obtained as those given by *peerAddr* and *peerPort* in `sceNpSignalingGetConnectionStatus()`.

Your Own IP Address and Port Number as Seen from Your Peer (SCE_NP_SIGNALING_CONN_INFO_MAPPED_ADDRESS)

This is your own IP address and port number as seen from your connected peer (communication target). For the connection between unit A and unit B, "your own IP address and port number as seen from your peer" for unit A will be the same value as "IP address and port number of the peer" for unit B.

Packet Loss Rate (SCE_NP_SIGNALING_CONN_INFO_PACKET_LOSS)

This is the packet loss percentage when making a round-trip of a UDPP2P packet to a connected peer and back. Measurement is made upon the exchange of the keep-alive packet, which takes place every 10 seconds. The value of the last 6 measurements is returned.

Network Information

The NP Signaling library has network information that is necessary for establishing and maintaining connections. Using API, the application can obtain its own network information as well as that of the communication target.

Local IP Address

The local IP address is the IP address allocated to the interface.

External IP Address

The external IP address is the IP address used when connecting to the Internet. If the connection is via a NAT router, the address used by the NAT router to connect to the Internet is the external IP address. If connected directly to the Internet, the local IP address and the external IP address are the same.

NAT Status Type

The NAT status type indicates the level of support that the NAT Traversal function of the NP Signaling library provides for the various NAT router functions.

Type	Description
Unknown	Indicates either that the NAT status type is being identified or that the identification failed
Type 1	Indicates that the local IP address and the external IP address are the same. This would be the case when connected directly to the Internet.
Type 2	Indicates that the local IP address and the external IP address are different, and that one of the following applies: - allocation of external ports does not depend on the communication target - NP port status is Open
Type 3	Type other than Type 1 or Type 2

4 Precautions

NAT Traversal Function

The NAT Traversal function provided in the NP Signaling library is subject to restrictions that depend on the NAT configuration of the network environment (router) in combination with the communication target. The NAT Traversal function is enabled/disabled depending on the combination of the NAT status types of the two parties. Communication is enabled for two hosts when the NAT configurations are as follows.

		Other Host		
		Type 1	**Type 2**	**Type 3**
Host	**Type 1**	enabled	enabled	enabled
	Type 2	enabled	enabled	enabled
	Type 3	enabled	enabled	disabled

Note that it is possible for combinations marked "enabled" to be unable to communicate depending on the behavior of the individual routers. Communication is not necessarily guaranteed for all possible combinations of routers.

NAT Traversal Information

Information regarding the NAT Traversal function can be confirmed from **Network - NAT Traversal Information** in the ★**Debug Settings** function of Development Kits and Testing Kits.

STUN Status (Failed / Succeeded)

Indicates the usage of STUN.

NAT Type (Type1 / Type2 / Type3)

Indicates the NAT type.

Mapped Address

When STUN Status is Succeeded, indicates the external IP address obtained by STUN.

Mapping Policy (Endpoint Independent / Address Dependent / Address and Port Dependent)

When STUN Status is Succeeded, indicates the allocation policy for the external port of NAT obtained by STUN.

Port Preservation (True / False)

When STUN Status is Succeeded, indicates whether or not the external port obtained by STUN is the same as the local port.

Delta

When STUN Status is Succeeded, indicates the destination IP address of STUN, as well as the difference of the external port allocation if it has been changed.

Maximum Number of Connections

The maximum number of connections that the NP Signaling library can handle at one time is 64. This is the total sum of connections in PENDING and ACTIVE states. If `sceNpSignalingActivateConnection()` is called to activate a connection exceeding this limit, the error `SCE_NP_SIGNALING_ERROR_TOO_MANY_CONN` will be returned.

UDP Local Port Numbers for UDPP2P and TCP over UDPP2P

Depending on the connection target, an ephemeral port number in the range of 49167-65535 will be used.

For UDPP2P, 3658 (`SCE_NP_PORT`) is specified for the UDP local port number, however, the UDP local port number will be overwritten in the kernel as necessary.

Port Numbers to Specify for UDPP2P and TCP over UDPP2P

When using UDPP2P and TCP over UDPP2P, the values to specify in *`sin_port`* (port number) and *`sin_vport`* (virtual port number) in the `SceNetSockaddrIn` structure are as follows.

UDPP2P

	When performing **`sceNetBind()`** on the host	When performing **`sceNetBind()`** on the client	**`sceNetSendto()`**
`sin_port`	`SCE_NP_PORT` (3658)	`SCE_NP_PORT` (3658)	Port number obtained in `sceNpSignalingGetConnectionStatus()`
`sin_vport`	Game port	Game port	Game port

TCP over UDPP2P

	When performing **`sceNetBind()`** on the host	When performing **`sceNetBind()`** on the client	**`secNetConnect()`**
`sin_port`	Game port	Game port	Game port
`sin_vport`	`SCE_NP_PORT` (3658)	`SCE_NP_PORT` (3658)	Port number obtained in `sceNpSignalingGetConnectionStatus()`

Exclusive Processing in the Callback Function of the NP Signaling Library

When performing exclusive processing using a mutex, for example, in a callback function set by `sceNpSignalingCreateCtx()`, do not call an API of the NP Signaling library from another thread while unlocking the mutex. Because the mutex is used within the NP Signaling library to perform exclusive processing for internal resources, a deadlock may occur between the callback function thread and the other thread.

Restriction on the Connection Activation Frequency

When calling `sceNpSignalingActivateConnection()` and activating a connection, IP address and NAT information is exchanged with the communication target through the PSN℠ server, and allocation of a communication channel, etc. is performed. To prevent excess burdens due to the message exchange processing and to maintain a stable PSN℠ connection environment, keep the frequency of activating connections within 100 times for every 10 minutes.

Note that there is a possibility that the frequency limit will be changed in the future depending on PSN℠ usage conditions.