NP Basic Library Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Purpose and Characteristics

The NP Basic library provides a feature for exchanging messages on PSN℠ and a feature related to shared play history.

Main Features

Main features provided by the NP Basic library are as follows.

Message Exchanges

- Sending InGame data messages (arbitrary format) to a user who has the same NP Communication ID registered
- Receiving InGame data messages sent from a user who has the same NP Communication ID registered

Shared Play History

- Recording shared play history
- Obtaining shared play history/number of shared plays

Note
Features related to the friends list and presence information that were provided by the NP Basic library have been removed. To obtain friends information or set presence, use the User Profile Web APIs. For details, refer to the following Technical Notes on the PlayStation®Vita Developer Network.
https://psvita.scedev.net/technotes/view/423

Embedding into a Program

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

Load the PRX module in the program, as follows.

```
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_NP_BASIC) != SCE_OK ) {
    // Error handling
}
```

Upon building the program, link libSceNpBasic_stub.a.

Sample Program

A sample program using the NP Basic library is as follows.

sample_code/network/api_np/np_basic/

This program exemplifies the basic usage of the NP Basic library.

Reference Materials

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

- PSN℠ Overview

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

- NP Library Overview
- NP Library Reference

Refer to the following document regarding the network check dialog for switching service states of the NP library.

- Network Overview

Refer to the following document regarding the Friends application for viewing shared play history.

- System Software Overview

Refer to the following document regarding how to set in param.sfo the NP Communication ID to be used in the message exchange feature.

- Param File Editor User's Guide (contained in Publishing Tools)

Refer to the following document for cross-platform features realized with PlayStation®3 and with PlayStation®4.

- Cross-Platform Application Creation Guide

2 Feature Overview

Features Provided by the Library and Their Usage Conditions

Features provided by the NP Basic library may or may not be used depending on the PlayStation®Vita state.

The NP library service states (`SceNpServiceState`) are signed-out, signed-in, and online. The use of each NP Basic library feature depends on these 3 service states and whether a network is available for use or not, as follows.

NP Basic Library Feature	PlayStation®Vita **State (Service State/Network Availability)**				
	Signed-out		Signed-in		Online
	Network unavailable	Network available	Network unavailable	Network available	Network available
Sending/Receiving InGame data messages	N/A		N/A		**OK**
Shared play history	**OK(*)**		**OK**		**OK**

To use all features provided by the NP Basic library, use the network check dialog and set the service state to "Online".

(*)It is still possible to use the shared play history feature on its own in the signed-out state. However, due to this signed-out state, it may not be possible to obtain the IDs of players (`SceNpId` structure) used to register the shared play history. Even if shared play history information can be obtained at this time, it will not be possible to use PSN℠ communication features by setting player IDs taken from the shared play history as addressees.

Types of Communication Data

The NP Basic library supports the following types of communication data.

Type of Communication Data	**Description**
InGame data message	Communication data with content and format freely-defined by the application as long as it is within the defined size range

Limited Communication Parties

The sending and receiving of InGame data messages is possible as long as the other user has the same NP Communication ID set as the user - and typically when both parties are playing the same game - regardless of whether the other user is a friend or not.

Characteristics of InGame Data Messages

InGame data messages can be sent/received between parties who have the same NP Communication ID set (typically, users playing the same game).

The format of the message can be freely-defined by the application as long as it is within the defined size range. Game-specific information - such as, detailed status of a certain character - can be efficiently communicated in this way.

Note that InGame data messages are not kept on the server of PSN℠. Unlike email-type messages that are stored, InGame data messages are exchanged in real-time (like chats); thus, the service state of both the sender and the receiver must be online.

Shared Play History

It is possible to add text information (describing a play status) to an ID of a player with whom an online game was played and to store it as shared play history. In addition to being able to obtain shared play history from within the game, it can also be viewed from system software applications (ex. Friends application).

3 Using the Library

Pre-processing

(1) Loading the PRX

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP_BASIC` specified as the module ID to load the PRX.

(2) Initialize the library

Call `sceNpBasicInit()` to initialize the NP Basic library. `sceNpBasicInit()` returns 0 for successful initialization.

Receiving Events

The NP Basic library notifies the application of an event when a message is received from another user. The procedure to receive this event is as follows.

(1) Register an event handler for InGame data messages

Use `sceNpBasicRegisterInGameDataMessageHandler()` to register the event handler.

Set the NP Communication ID to be used for communication in the second argument of this function, *`commId`*. If NP Communication ID is not set in this function, the NP Communication ID specified in the initialization function of the NP library, `sceNpInit()`, will be used in its place.

(2) Poll sceNpBasicCheckCallback()

Regularly call `sceNpBasicCheckCallback()` to receive event notification of an event through the registered event handler. An event is not automatically notified when it is generated. The application must call `sceNpBasicCheckCallback()` to call the applicable event handler and be notified of the event.

(3) Processing of the event handler

Note the following when performing event handler processing

- An event handler is executed on the thread context on which `sceNpBasicCheckCallback()` was called
- Processing within an event handler should be simple and it is recommended that control return immediately from the event handler
- When a pointer to certain information is passed to an event handler, the memory area storing that information will be released as soon as control returns from the event handler. If that information is required after returning, the application must copy the information within the event handler onto memory.

(4) Delete the event handler

When it is no longer necessary to receive events, call `sceNpBasicUnregisterInGameDataMessageHandler()` to delete the event handler.

Sending InGame Data Messages

It is possible to send an InGame data messages using `sceNpBasicSendInGameDataMessage()`. Specify the user to whom the message is addressed in the first argument, and the data to be sent in the second argument. Sent data will not be accumulated in the PSN℠ server; it will only reach the other user's game when the addressee is running the same game, in online and has the same NP Communication ID registered with `sceNpBasicRegisterInGameDataMessageHandler()`.

In order to avoid overload of the PSN℠ server, frequency of sending/receiving message is limited to 20 times per minute.

Receiving InGame Data Messages

The user can only receive messages from users with the same NP Communication ID set.

The handler registered with `sceNpBasicRegisterInGameDataMessageHandler()` is called in the thread that called `sceNpBasicCheckCallback()`. Moreover, because the data received by the handler can only exist within the scope of that handler's call, copy the data within the handler to an appropriate memory area.

Recording Shared Play History

Shared play history can be recorded using `sceNpBasicRecordPlaySessionLog()`. Set the users played with in *`withWhom`*, the first argument, and a description of the play session in *`description`*, the second argument. When this function is called, the system will complete information such as the game title and the time of the call, storing the information in the system's internal database.

Note
The title name that is recorded upon saving shared play history is based on the value set to param.sfo of the user's PlayStation®Vita system. If param.sfo is not correctly set during development, a tentative string representing this state will be recorded.

Obtaining Shared Play History/Number of Shared Plays

The number of recorded shared plays can be obtained using `sceNpBasicGetPlaySessionLogSize()`. Shared play history can be obtained by using `sceNpBasicGetPlaySessionLog()`.

Post-processing

(1) Terminate the library

Call `sceNpTerm()` to terminate the NP Basic library.

(2) Unload the PRX

Specify `SCE_SYSMODULE_NP_BASIC` as the module ID and call `sceSysmoduleUnloadModule()` to unload the PRX.

4 Notes

Operations that Must Be Performed by the Game During System Suspension and Process Suspension

PlayStation®Vita has two types of suspension: system suspension and process suspension.

Since connection to the network is interrupted when system suspension occurs, the service state of the NP library changes to "signed-in" immediately after the system resumes, and features that require "online state" in the NP Basic library (InGame data message features) are disabled. When this happens, transition the state within the game to one before the "online state" is required (for example, to the menu screen before starting online multi-play, etc.). After this, if "online state" is again necessary in the game, specify the PSN℠ online mode by calling the network check dialog, and resume use of the NP Basic library

The game process can be suspended by pressing the PS button. This is called process suspension. The response to the occurrence of process suspension is different from the response to the occurrence of system suspension. Suspension of the game process will not interrupt network connection necessary to the NP Basic library. The network connection necessary to the NP Basic library is supported by the system software, which keeps running even while the game process is suspended. Unless the system software is suspended (unless, that is, system suspension occurs) the network will remain connected.

In the NP Basic library, events related to InGame data messages are received from the system software through inter-process communication. Events are received in a queue set up in the NP Basic library. If process suspension of the game process occurs, applicable events will be accumulated in the queue in the NP Basic library from the system process side. These events will not be consumed if the game process is suspended. Therefore, when a certain amount of events is accumulated, a queue overflow may occur. If the game process is resumed from process suspension before the queue overflows, the game process will not delete any events and will be able to receive all events received up to that point. If a queue overflow occurs during process suspension, the service state will change from "online" to "signed-in", and the events accumulated in the queue up to that point will be discarded by the system software. If the game resumes during this state, response should be taken in the same manner as in the as in the case of system suspension - transitioning to the state before the "online" state is required (for example, transitioning to the menu screen before starting online multi-play, etc.). The method for returning to the "online" state from this state is the same as in the case of system suspension.

Message Exchange with PlayStation®3 and with PlayStation®4

For notes regarding the exchange of InGame data messages with PlayStation®3 and with PlayStation®4, refer to the "Cross-Platform Application Creation Guide" document.