NP Party Library Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Features

The NP Party library allows your application to get information about players who are using the Party application for system voice chat on PlayStation®Vita. It allows your application to gather information about the players in a party and then use this information to create application-specific online games for members in the party.

Also, the NP Party library allows your application to track information about the party itself and the state of each party member. It allows the game to keep all party members together as one group while playing their game online.

More specifically, the NP Party library provides functions that allow your application to:

- Determine whether a client is currently in a party
- Receive notifications when a player enters or leaves a party
- Determine the state of a party
- Receive information about other members who are in the party but may not be playing your game

System Resources

`libSceNpParty` uses the following system resources (Table 1).

Table 1 System Resources

Resource	Description
Footprint	Minimal.
Work memory	Heap space 4 KB
Thread	None.
Processor time	Minimal.
Network bandwidth usage	The maximum size of a party is currently 8 players. The party application uses network bandwidth to facilitate party organization and system voice chat. The maximum upstream bandwidth used by the party application for voice chat will be (1500*7) bytes/sec for an 8-player party. The maximum downstream bandwidth for voice chat will be –(1500*7) bytes/sec for an 8-player party.

Embedding into Your Program

Include `np/np_party.h` in your source files; this will automatically include additional header files as well.

Load the PRX module in the program, as follows:

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

To build the program, link `libSceNpParty_stub.a`.

Party System Application on PlayStation®Vita

The NP Party API should be used in conjunction with the Party application on PlayStation®Vita. The Party application can be launched from the HOME screen. The party can be created before or after your game is launched. Your game should not make assumptions about when the party will be created.

There are two types of parties that a user can create:

(1) Public: parties in which friends and "friends of friends" can join.

(2) Private: parties in which only invited users are allowed to join.

You can use the `privateParty` Boolean member of `SceNpPartyMemberList` to determine if the user has joined a public or private party. Alternatively, you can use the `sceNpPartyGetState()` function.

Table 2 summarizes how to use the party application to affect state or information change in the NP Party API.

Table 2 Party System Application on PlayStation®Vita

User Action*	NP Party API
Create a party.	`sceNpPartyGetState()` and `sceNpPartyGetMembers()` will be updated.
Join a party.	`sceNpPartyGetState()` and `sceNpPartyGetMembers()` will be updated.
Party member joined.	`sceNpPartyGetState()` and `sceNpPartyGetMembers()` will be updated.
Party member left.	`sceNpPartyGetState()` and `sceNpPartyGetMembers()` will be updated.
Leave party.	`sceNpPartyGetState()` will return a status of not in party (`SCE_NP_PARTY_STATE_NOT_IN_PARTY`). `sceNpPartyGetMembers()` will return an error of not in party (`SCE_NP_PARTY_ERROR_NOT_IN_PARTY`).
Kicked from party.	`sceNpPartyGetState()` will return a status of not in party (`SCE_NP_PARTY_STATE_NOT_IN_PARTY`). `sceNpPartyGetMembers()` will return an error of not in party (`SCE_NP_PARTY_ERROR_NOT_IN_PARTY`).
Not in party.	`sceNpPartyGetState()` will return a status of not in party (`SCE_NP_PARTY_STATE_NOT_IN_PARTY`). `sceNpPartyGetMembers()` will return an error of not in party (`SCE_NP_PARTY_ERROR_NOT_IN_PARTY`).

* In this context, "user action" refers to actions initiated by the user within the party application. A user action causes the state of the party to be updated, and this information is then reflected in the NP Party API.

Table 3 shows the party room events that the application will receive when certain actions occur in the party application.

Table 3 Party Room Events Received by the Application

User Action*	Party Room Event Type	Party State
Create a party.	`SCE_NP_PARTY_ROOM_EVENT_JOINED`	`SCE_NP_PARTY_IN_PARTY`
Join a party.	`SCE_NP_PARTY_ROOM_EVENT_JOINED`	`SCE_NP_PARTY_IN_PARTY`
Party member joined.	`SCE_NP_PARTY_ROOM_EVENT_MEMBER_JOINED`	`SCE_NP_PARTY_IN_PARTY`
Party member left.	`SCE_NP_PARTY_ROOM_EVENT_MEMBER_LEFT`	`SCE_NP_PARTY_IN_PARTY`
Create a private party.	`SCE_NP_PARTY_ROOM_EVENT_JOINED`	`SCE_NP_PARTY_STATE_IN_PRIVATE_PARTY`
Join a private party.	`SCE_NP_PARTY_ROOM_EVENT_JOINED`	`SCE_NP_PARTY_STATE_IN_PRIVATE_PARTY`

User Action*	Party Room Event Type	Party State
Party member joined private party.	SCE_NP_PARTY_ROOM_EVENT_MEMBER_JOINED	SCE_NP_PARTY_STATE_IN_PRIVATE_PARTY
Party member left private party.	SCE_NP_PARTY_ROOM_EVENT_MEMBER_LEFT	SCE_NP_PARTY_STATE_IN_PRIVATE_PARTY
Leave party.	SCE_NP_PARTY_ROOM_EVENT_LEFT	SCE_NP_PARTY_NOT_IN_PARTY
Kicked from party.	SCE_NP_PARTY_ROOM_EVENT_LEFT	SCE_NP_PARTY_NOT_IN_PARTY
Not in party.	N/A	SCE_NP_PARTY_NOT_IN_PARTY

* Refer to the table note in Table 2.

Multi-platform Support

The party feature contains cross-platform support for users of both PlayStation®4 and PlayStation®Vita. Both platforms can receive and use party invitations sent by another Np account. Therefore, users of both platforms can enter the same party and voice chat with each other. Your application can obtain the platform type for each party member by calling `sceNpGetPlatformType()` on the party member's *`NpId`*.

Sample Programs

A sample program is provided to show basic usage of the NP Party library, enabling the developer to easily understand the basic flow from initialization to retrieving information about party and party members (including voice notification). This program is located at

`samples/sample_code/network/api_np/np_party/simple`.

Related Documentation

For an overview of PSN℠ functionality, refer to the following document on PlayStation®Vita DevNet:

- *PSN℠ Overview*

For information about the NP Party library and session/invitation web APIs (which are commonly required when using PSN℠ functionality), refer to the following documents on PlayStation®Vita DevNet:

- *NP Party Library Reference*
- *Session/Invitation Overview*
- *Session/Invitation Web APIs Reference*

2 Configuration and Terminology

This chapter explains the configuration and terminology of the NP Party library.

Party

The party feature is a system-level voice chat feature that runs independently of the game. The party application can be started up before a game is running, and players can group together for online play. Each player in a party can be acting independently and be playing in different games both in offline and online mode. The party is game and application agnostic. The application should not transmit game-specific data via the party feature.

The party feature consists of the party application used for system voice chat and the NP Party library described in this document. The following table provides descriptions of several key party-related terms.

Table 4 Key Party-Related Terms

Term	Description
Joinable game	An online game that other players can join. A single player game or an online game that is full are not joinable games.
Late join	An online game that allows players to join a game mode after the online game has begun. For example, a racing game may not allow late join because new players typically are not allowed to start a race after the race has begun.
Party room	The party room is the grouping of players that are joined together into a party via the party application. A player can only be in one party at a time.
Party room leader	The party room leader is a party member who is in charge of the room. The first party room leader of a particular room is typically the player who created the party. The leader will change automatically when the current room leader leaves the party room. The application should not make any assumptions that the room leader is playing the same online game as another user.
Party member	A party member is a player in the party room.
Party size	The maximum party size is currently 8 players.

Event Callback Functions

The NP Party library notifies the application of events that occur within the party application, such as a user joining or leaving a party. These events are notified as events in the callback functions that are registered by the application. As explained below, there are several types of events and corresponding callback functions. For details about each event and callback function, refer to the *NP Party Library Reference*.

Room Events and the Room Event Callback Function

A party room event is an event that is associated with the party room that the player joins. A notification about a party room event is sent to the party room event callback function.

Party room events are as follows:

- This member (“I”) joined a party room (`SCE_NP_PARTY_ROOM_EVENT_JOINED`).
- This member (“I”) left a party room (`SCE_NP_PARTY_ROOM_EVENT_LEFT`).
- A member joined the room (`SCE_NP_PARTY_ROOM_EVENT_MEMBER_JOINED`).
- A member left the room (`SCE_NP_PARTY_ROOM_EVENT_MEMBER_LEFT`).
- A member’s game presence has been updated (`SCE_NP_PARTY_ROOM_EVENT_MEMBER_PRESENCE_UPDATE`).

The prototype for the room event callback function is as follows:

```
typedef void (*SceNpPartyRoomEventHandler)(
    SceNpPartyRoomEventType eventType,
    const void*data,
    void* userdata);
```

The event data for a room event is obtained by casting the `const void data` pointer to the appropriate data type.

Table 5 Party Room Events

Event	Event Data Type
SCE_NP_PARTY_ROOM_EVENT_JOINED	SceNpPartyMemberList*
SCE_NP_PARTY_ROOM_EVENT_MEMBER_JOINED	SceNpPartyMemberInfo*
SCE_NP_PARTY_ROOM_EVENT_MEMBER_LEFT	SceNpPartyMemberInfo*
SCE_NP_PARTY_ROOM_EVENT_LEFT	SceNpPartyRoomLeftReason*
SCE_NP_PARTY_ROOM_EVENT_MEMBER_PRESENCE_UPDATE	SceNpPartyMemberInfo*

Voice Notification States and the Voice-Event Callback Function

You can use the NP Party API to determine which members in a party are currently speaking. This can be useful if you want a game screen to display who is speaking in a party. To receive notification of voice events, the `voiceEventHandler` must be supplied when calling `sceNpPartyRegisterHandler()`.

Voice notification states are as follows:

- A voice peer-to-peer connection has been established with this member (`SCE_NP_PARTY_MEMBER_VOICE_STATE_CONNECTED`).
- This member is currently speaking (`SCE_NP_PARTY_MEMBER_VOICE_STATE_TALKING`).
- A voice peer-to-peer connection has not been established with this member (`SCE_NP_PARTY_MEMBER_VOICE_STATE_DISCONNECTED`).
- This member has been muted in the party application (`SCE_NP_PARTY_MEMBER_VOICE_STATE_MUTED`).

The NP Party API will only send an `SCE_NP_PARTY_MEMBER_VOICE_STATE_TALKING` when a party member is speaking (or a voice packet is received). It will not specify when the party member has finished talking. Your game should use a timer to expire a party member from a talking state. This limitation was imposed to reduce the number of voice notifications sent via the NP Party API. It is recommended that games indicate that a party member is in a talking state for up to one second after an `SCE_NP_PARTY_MEMBER_VOICE_STATE_TALKING` voice event occurs.

The prototype for the voice notification callback function is as follows:

```
typedef void (*SceNpPartyVoiceEventHandler)(
    const SceNpPartyMemberVoiceInfo* memberVoiceInfo,
    const void* userdata);
```

Game-Session Notification States and Game-Session Event Callback Function

To receive notification of game-session events, you must supply the `SceNpPartyGameSessionEventHandler` when calling `sceNpPartyRegisterHandler()`. Your application must then wait for the `SCE_NP_PARTY_GAME_SESSION_UPDATE_EVENT_READY` event and check that `SceNpPartyGameSessionUpdateReady.result` returns `SCE_OK`. If `SceNpPartyGameSessionUpdateReady.result` is not equal to `SCE_OK`, your application must terminate, re-initialize the NP Party API, and then re-register the `SceNpPartyGameSessionEventHandler` to request the `SCE_NP_PARTY_GAME_SESSION_UPDATE_EVENT_READY` event again. Without an

`SceNpPartyGameSessionUpdateReady.result` value of `SCE_OK`, the NP Party API will not send any party member game-session updates to your application.

Game-session notification states are as follows:

- The NP Party API is ready to send remote party member game-session updates. (`SCE_NP_PARTY_GAME_SESSION_UPDATE_EVENT_READY`)
- This remote party member's game session has changed. (`SCE_NP_PARTY_GAME_SESSION_MEMBER_UPDATE_EVENT`)

The `SceNpPartyGameSessionEventHandler()` will only provide details about the remote party's session ID if:

- The remote party member is in a joinable game with a valid session ID (that is, this player has a joinable game, which means that `SceNpPartyMemberGameSessionUpdateInfo.type` has a value of `SCE_NP_PARTY_GAME_SESSION_UPDATE_TYPE_JOINABLE`)
- The remote party member is playing the same game as the current application

For more information, refer to the *Session/Invitation Web APIs Reference* and its associated overview.

The prototype for the game-session event callback function is as follows:

```
typedef void (*SceNpPartyGameSessionEventHandler)(
    SceNpPartyGameSessionEvent* event,
    const void* data,
    const void* userdata);
```

You can obtain the event data for a game-session event by casting the `const void data` pointer to the appropriate event data type (Table 6).

Table 6 Party Game-Session Events

Event	Event Data Type
`SCE_NP_PARTY_GAME_SESSION_UPDATE_EVENT_READY`	`SceNpPartyGameSession UpdateReady*`
`SCE_NP_PARTY_GAME_SESSION_MEMBER_UPDATE_EVENT`	`SceNpPartyMemberGameSession UpdateInfo*`

Heap Area

The NP Party library has its own heap area. The library allocates a 4-KB memory block.

3 Processing Flow and Examples

This chapter explains the typical processing flow for applications that use the NP Party library. Examples are also provided. For the code examples in this chapter, the following variable is supposed to be declared for storing the return value of functions:

```
int ret;
```

(1) Load the PRX.

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP_PARTY` specified as the module ID.

(2) Initialize the NP library.

Initialize the NP library by calling `sceNpInit()`. Set the NP Communication ID, NP pass phrase, and NP signature to the `SceNpCommunicationConfig` structure. These values are issued per application; apply on the PlayStation®Vita Developer Network (https://psvita.scedev.net/). Be sure to set the issued values correctly.

(3) Initialize the NP Party library.

Initialize the NP Party library by calling `sceNpPartyInit()`:

```
// Initialize NP Party system
SceNpPartyInitParam params;
sceNpPartyInitParamInit(&params);

ret = sceNpPartyInit(&params);
if(ret < SCE_OK){
    // Error handling
}
```

(4) Register the event callback function.

After your application initializes the API, it should register the event callback function to receive events about the party and party members.

To register the callback functions, execute `sceNpPartyRegisterHandler()`:

```
// Room event callback function
void PartyRoomEventCallbackHandler (
    SceNpPartyRoomEventType   eventType,
    const void                *data,
    void                      *userdata
    )

{    . . . .

    if (event == SCE_NP_PARTY_ROOM_EVENT_JOINED) {
        // Processing when the player joins a party
    }
    else if (event == SCE SCE_NP_PARTY_ROOM_EVENT_MEMBER_JOINED) {
        // Processing when another party room member joins a party
    }
    else if (event == SCE_NP_PARTY_ROOM_EVENT_MEMBER_LEFT) {
        // Processing when another party room member leaves the party
    }

    . . . .
```

```
}

// Voice-event callback function
void voiceEventCallbackHandler(
    const SceNpPartyMemberVoiceInfo* memberVoiceInfo,
    void* userdata
    )
{
    // Processing voice state of party member (talking/mute)
}

// Game-session event callback function

void gameSessionEventCallbackHandler(
    SceNpPartyGameSessionEvent gameSessionEvent,
    const void *data,
    void *arg)
{
    // Processing party game session events (joinable/not joinable)
}

SceNpPartyEventHandlers handlers;
sceNpPartyEventHandlersInit(&handlers);
handlers.roomEventHandler  = PartyRoomEventCallbackHandler;
handlers.voiceEventHandler = voiceEventCallbackHandler;
handlers.gameSessionEventHandler = gameSessionEventCallbackHandler;

ret = sceNpPartyRegisterHandler( &handlers, NULL);
if (ret < 0) {
    // Error handling
}
```

(5) Receive the event callback function.

Your application should call `sceNpPartyCheckCallback()` in order to receive data via the callbacks registered by `sceNpPartyRegisterHandler()`.

(6) Retrieve state information.

To retrieve information about the party and the party room members, you can use the functions below to retrieve information as needed. If your application would like to receive real-time updates, it is recommended that you register the Room Event handler described previously.

Table 7 Functions for Retrieving State Information

Information Item	Use this function to retrieve the indicated information . . .
Party State	To retrieve information about whether the room member is in the party, your application can call `sceNpPartyGetState()`.
Party Member List	To retrieve information about party members, your application can call `sceNpPartyGetMembers()`.
Party Member Info	To retrieve information about a party member, your application can call `sceNpPartyGetMemberInfo()`.
Party Member Voice Info	To retrieve information about a party member's voice connection, your application can call `sceNpPartyGetMemberVoiceInfo()`.
Party Member Session Info	To retrieve information about a remote party member's game session, your application can call `sceNpPartyGetMemberSessionInfo()`

(7) Suspend/resume.

When your application is resumed after being suspended, it must call `sceNpPartyCheckCallback()` to retrieve any queued up events. These events should then be ignored by your application. Your application must then call `sceNpPartyGetState()` to check if the user is in a party. If the user is in a party, your application must call `sceNpPartyGetMembers()` to get the latest party member list. If your application is using voice events, it must call `sceNpPartyGetMemberVoiceInfo()` to retrieve the latest voice state for each party member. If your application is using game-session events, it must call `sceNpPartyGetMemberSessionInfo()` to retrieve the game-session state of each remote party member.

(8) Terminate the NP Party library.

When you no longer need to use the NP Party library (for example, upon termination of the application), execute the following:

```
// Terminate the NP Party library
sceNpPartyTerm();
```

(9) Terminate the NP library.

```
ret = sceNpTerm();
if (ret < 0) {
    // ERROR HANDLING
}
```

(10) Unload the Party PRX.

```
// Unload the NP Party PRX
ret = sceSysmoduleUnloadModule(SCE_SYSMODULE_NP_PARTY);
if (ret < 0) {
    // ERROR HANDLING
}
```

4 Use Cases and Implementation

This chapter describes use cases for applications that use the NP Party library.

Creating a Staging Room for Party Members (no late join)

When a player enters a game, the game can detect that the user is in a party and create a staging room for the party room members. This will allow the game to determine how many players from the party intend to play together at the beginning of an online game play session. The staging room can be used to congregate players together. After the staging room owner decides to stop waiting for additional party members, the game can allow the staging room owner to launch the current party members from the current staging room into game for online play.

Creating a Game Room for Party Members

When a player enters a game, the game can detect that the user is in the party. The game can then prompt the user to see if it should create an online game for the party members. When this player is placed into that online game, the game can create a session for this user.

The party application will then display the session details for this user in the party room. Other members in the party will then be able to join this newly created game by touching this player's room member button in the party application. The party application will then launch the game for the other party members and provide the game the details for placing each party member into the newly created game.

Joining a Game Room for Party Members

When a player enters a game, the game can detect that the user is in the party and that other party members are currently playing in a game. The game can then ask the user if the player wants to be placed into the game with their fellow party members.

5 Items That Must Be Implemented

This chapter explains items that must be implemented by all applications that use the NP Party library.

Use NP Party Library in an NP Signed-in State

The NP Party library must be used in an NP signed-in state. Make sure you start the Network Check Dialog to enter the NP signed-in state before transitioning to the state for using the NP Party library.

For details about the Network Check Dialog, refer to the *Network Overview* and *libnetctl Reference*.

Support Session/Invitation Web APIs

Session/invitation web APIs are used to create a game session that other players can join. Your game should support these APIs in order to facilitate the ability of other party members to join online games.

6 Recommended Implementation Items and Precautions

This chapter explains items that are recommended when implementing an application that uses the NP Party library.

Games for Party Members

The game should consider creating games for party members. The total number of players allowed in an online game may be less than the maximum party size. For team based games, it would be beneficial to allow party members to be on the same team.

Party Creation/Destruction

The game should not make any assumptions about when the party is created. It is possible that a game is launched first and then a party is created afterwards. Similarly, the user may decide to leave the party at any time. When a user leaves the party this should not prevent them from continuing normal online game play with their previous party members.

Party Voice Chat

There may be a concern about players using the party voice chat feature to gain an unfair advantage while playing your online multiplayer game. Make sure that your game deals with any such issues appropriately.

Party Room Leader

The party room leader may or may not be playing your game. The game should not depend on the party room leader to make any game-specific decisions.