NP ScoreRanking 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 Features

The NP ScoreRanking library is a library for using the "ranking server" provided by PSN℠.

The PSN℠ ranking server provides multiple scoreboards onto which a score can be registered per user (player), in applications for which score ranking services have been requested. Settings can be made to each of the scoreboards - including the number of users that can be registered, the condition for updating scores, and the order in which scores are displayed. A score can be registered with a comment. Moreover, replay data and character data can be attached as well, as long as overall game data is within the set size limit. There are 3 types of ranking: serial ranking (where same scores are ranked according to the order by which they are registered; a score registered earlier gets higher ranking), ranking (where same scores are ranked the same), and the highest rank achieved by a given score.

Communication with the ranking server is performed using synchronous and asynchronous APIs. Large data can be sent/received in increments using multiple function calls.

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_SCORE_RANKING) != SCE_OK ) {
    // Error processing
}
```

Upon building the program, libSceNpScore_stub.a.

Sample Programs

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

sample_code/network/api_np/console/

This sample shows the basic procedure for using the NP ScoreRanking library.

Reference Materials

Refer to the following document for an overview of the 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

2 Components of the NP ScoreRanking Library

This chapter explains the components of the NP ScoreRanking library.

User

The NP client libraries have an NP ID and NP ID structure defined to identify users. The NP ScoreRanking library also uses NP IDs to identify users. For details on the NP ID and NP ID structure, refer to the "NP Library Reference" document.

Player Character ID

The player character ID is a unique ID used by the NP ScoreRanking library. It can be used, for example, to handle a player named foo as multiple people, each with a different name of foo-0, foo-1, foo-2, etc. Make use of the player character ID when you want to handle each score of a single player as belonging to a different player, when the save data can be saved in multiple locations. Set the player character IDs using `sceNpScoreSetPlayerCharacterId()`. In terms of format, a value should be attached to the NP ID for each player character ID. Values from 0 to 9 can be set, and the total number of player character IDs that can be created per player is 10.

Because the player character ID is only valid in the NP ScoreRanking library, care is required when coordinating with the other NP features, if it is used. For example, the NP ID is used to register a player as a friend. It will be possible to register a user named foo as a friend, but it will not be possible to distinguish between foo-0 and foo-2 when sending a message. Moreover, because an invitation message gets sent to a player by specifying his/her NP ID in the NP matching system, foo-1 cannot be specified as the destination of an invitation message. In addition, information that can be added to the NP ID, such as, the Avatar and profile is limited to 1 each per NP ID; they cannot be set per player character ID.

When using player character IDs, make sure you consider application specifications with the above obstacles in mind.

Server

Use of the ranking server is by application only. A server will only be provided for titles that use score ranking.

Scoreboard

Multiple scoreboards are provided per server. Basically, a user is permitted to enter only one score per scoreboard. However, up to 10 scores can be entered by the use of player character IDs. Scoreboards can be customized as indicated below, but the settings are locked (cannot be changed) after the title is submitted to SCE QA.

- Change the number of scoreboards or the maximum number of user entries per board
- Change score display to ascending order/descending order
- Set mode for score registration (overwrite without exception or overwrite only when breaking record)
- Change the maximum size of game data to be attached to a ranked score
- Change the minimum rank for which game data can be registered and attached to a score
- Set the ranking log output and score reset schedule per scoreboard (log is in CSV format)

There is a time lag from when a score is registered to the scoreboard until it actually gets reflected in the ranking, due to the time it takes for the server to run processing entailed in the creation of the new ranking. The time lag is usually 5 to 10 minutes. This time lag will be shortened if smaller values are set to maximum number of user registration entries and size of game data to be attached to scores. Note, however, that the time lag may be lengthened if a problem occurs or when a scoreboard is manually operated on (for example, when server problems occur, or to delete an invalid score) - program your application so that it does not break down in such cases. Moreover, you will be able to obtain a "temporary rank" for your score when you register it. This is the rank determined for your score at that exact moment of score registration - take note that this rank is determined independent to the above batch processing.

Scoreboards are subject to the following limitations.

- The number of user registrations on all the scoreboards must not exceed 3,200,000
- The total number of scoreboards must not exceed 1000
- The number of scores on one scoreboard must not exceed approximately 1,000,000
- The total size of data attachments on all the scoreboards must not exceed 1 GiB
- The size of one data attachment must not exceed 1 MiB
- To use PSN℠ Server Management Tools, the scoreboard ID must be in the range of 0-999
- If possible, avoid setting the ranking log output and score reset schedule to the time 00:00
  If there are already many scoreboards using a particular time, a different time may need to be selected.

3 Using the Library

Initialization

To use the NP ScoreRanking library, first initialize the NP library and then initialize the NP ScoreRanking library.

(1) Load PRX

Load PRX by specifying `SCE_SYSMODULE_NP_SCORE_RANKING` for the module ID and calling `sceSysmoduleLoadModule()`.

(2) Initialize NP library

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

(3) Initialize the NP ScoreRanking library

Initialize the NP ScoreRanking library by `sceNpScoreInit()`. With this process, a thread for inter-process communication is created in the application process and shell process respectively. For security reasons, NP ScoreRanking library communication is performed in the shell process. Therefore, it is not necessary to initialize libhttp and libssl. However, some of the PSN℠-related libraries require the initialization of libhttp and libssl. Concerning this, refer to the respective documents.

```
ret = sceNpScoreInit(SCE_KERNEL_DEFAULT_PRIORITY_USER,
SCE_KERNEL_THREAD_CPU_AFFINITY_MASK_DEFAULT, NULL);
if (ret < 0) {
        // Error handling
}
```

(4) Create a NP ScoreRanking title context

Call `sceNpScoreCreateTitleCtx()` to create an NP ScoreRanking title context. At this time, if NULL is specified as argument for NP Communication ID and NP communication passphrase, the value specified in `sceNpInit()` will be used. Specify target NP Communication IDs and NP communication passphrases other than NULL when using multiple NP Communication IDs.

Communication

(1) NP ScoreRanking request ID

Create a request ID to use for aborting or deleting a communication process. Create this per communication process and delete after communication process completion.

```
int ret, reqId, titleCtxId;

// titleCtxId assumed to store an appropriate value

ret = sceNpScoreCreateRequest (titleCtxId);
if (ret < 0) {
        // Error handling
}
reqId = ret;
// Communication process
```

```
ret = sceNpScoreRecordScore (
        reqId,       // Request ID
        1,           // Scoreboard number
        100000,      // Score
        comment,     // comment assumed to store an appropriate value
        info,        // info assumed to store an appropriate value
        &tmpRank,    // Store temporary rank upon score registration
        NULL);
if (ret < 0) {
        // Error handling
}
....

// Destroy request ID
sceNpScoreDeleteRequest (reqId);
```

(2) Special transaction

Functions `sceNpScoreGetGameData()` or `sceNpScoreGetGameDataAsync()` and `sceNpScoreRecordGameData()` or `sceNpScoreRecordGameDataAsync()` can be called multiple times within 1 communication process to adjust the size of data to be sent/received in 1 function call.

Note, however, that you will be unable to use another communication process until sends/receives of all data are completed for this communication process.

```
int ret, reqId, titleCtxId;
char data[BLOCKSIZE];
size_t remainSize, sendSize;
// titleCtxId and data are assumed to store appropriate values


ret = sceNpScoreCreateRequest (titleCtxId);
if (ret < 0) {
        // Error handling
}
reqId = ret;

// Specify total size of data to be sent
remainSize = TOTALSIZE;

while (remainSize != 0){
        // Determine send size
        if (remainSize < BLOCKSIZE){
              sendSize = remainSize;
        } else {
              sendSize = BLOCKSIZE;
        }
        ret = sceNpScoreRecordGameData (
        reqId,       // Request ID
        1,           // Scoreboard number
        100000,      // Score for which data is to be attached
        TOTALSIZE,   // Total size of data to be sent
        sendSize,    // Size of data to be sent in current send
        data,        // Data to be sent in current send
        NULL);
        if (ret < 0) {
              // Error handling
        }

        remainSize -= sendSize;
}
....
```

```
// Destroy request ID
sceNpScoreDeleteRequest (reqId);
```

Error Notification

Error codes used in NP ScoreRanking library support the Message Dialog library. When notifying the user of an error related to NP ScoreRanking library, set error codes - that are returned by the NP ScoreRanking library APIs - to the Message Dialog library, and make the notification. Especially in cases where the error greatly affects the user such as when failing to register a record-breaking score, we recommend that the error be notified to the user.

Errors that are not fatal, such as when `SCE_NP_COMMUNITY_SERVER_ERROR_NOT_BEST_SCORE` occurs upon call of `sceNpScoreRecordScore()` or `sceNpScoreRecordScoreAsync()` for a scoreboard (set to overwrite score only when breaking record), do not have to be notified to the user. In this case, however, lighten server burden by locally checking whether the record score has been exceeded or not before attempting to register the score.

Moreover, when title specifications do not distinguish between mode for performing score ranking and normal mode, design title operation so that the `SCE_NP_COMMUNITY_SERVER_ERROR_RANKING_END_OF_SERVICE` notification (returned after service of the NP ScoreRanking library ends) does not compromise normal play.

Termination

(1) Destroy NP ScoreRanking title context

When you no longer need the title context, call `sceNpScoreDeleteTitleCtx()` to destroy it.

```
int ret;
uint32_t titleCtxId;

// titleCtxId assumed to store an appropriate value

ret = sceNpScoreDeleteTitleCtx(titleCtxId);
if (ret < 0) {
        // Error handling
}
```

(2) Terminate NP ScoreRanking library

Call `sceNpScoreTerm()` to terminate the NP ScoreRanking library. Make sure to call this function after destroying all created score ranking request ID using `sceNpScoreDeleteRequest()`.

```
// Thread on the system side will stop
sceNpScoreTerm();
```

(3) Terminate NP library

Call `sceNpTerm()` to terminate use of the NP library.

```
// Always successful
sceNpTerm();
```

Then, unload PRX by calling `sceSysmoduleUnloadModule()` with `SCE_SYSMODULE_NP_SCORE_RANKING` specified for the module ID.

Notes Regarding Server Loads and Request Frequency

If the title is designed in a way such that the frequency of requests is extremely high, server overloads are possible when the number of users increases. Thus note the following points (in particular for large-scale titles) in the planning and implementation of title specifications.

Frequency of Score Registrations

On average, relative to the overall playing time of the game, the frequency of registering scores must be kept to under once every 5 minutes. For example, it is forbidden to register the score per game if the game ends on average every 30 seconds. However, if the title is composed of a main game and mini-games that utilize the ranking feature, and the playing time in the main game can be assumed to be relatively longer, then the frequency of registration of the mini-game scores can be calculated to be sufficiently low. Moreover, when accessing multiple boards at the same time, registration to four boards can be counted as one registration. In other words, when registering one score to four boards - normal ranking, today's ranking, this week's ranking, and this month's ranking - at the same time, this can be counted as one registration.

Using `sceNpScoreRecordGameData()` and `sceNpScoreRecordGameDataAsync()` to register scores can result in high server loads. If the data size is sufficiently small, use `sceNpScoreRecordScore()` and `sceNpScoreRecordScoreAsync()` instead, since the argument *`gameInfo`* of these functions allows the registration of data of up to 189 bytes for as many users as necessary.

Frequency of Score Accesses

If the following rules are observed, there is no limit on the frequency of accesses to the server to obtain scores.

- All obtained score information is displayed onscreen
- Score information is not discarded
- Score information is not obtained unless it is known that it is necessary
- Large amounts of data are not shown in summaries (averages of all user scores registered to the board, for example)

Even if these points are not honored, it is still acceptable to access the server for scores approximately once every 5 minutes. However, server performance is finite, so avoid unnecessary accesses as much as possible, by, for example, caching the data of attachments that are known to be used more than once.

Number of Simultaneous Communication Processes

When there are multiple communication processes to be executed simultaneously (for registering scores and accessing ranking data etc,), keep the number of simultaneous communication processes to 6 processes or fewer. The simultaneous execution of numerous communication processes causes not only the problem of server loads but extremely long wait times depending on the user's network environment. For example, a network delay will occur if a communication process is executed per friend when obtaining ranking data of a user's friends. Regarding the ranking of friends, the `sceNpScoreGetFriendsRanking()` function can be used to obtain the top 100 rankings among friends. In addition, `sceNpScoreGetRankingByNpId()` can be used to obtain the ranking of up to 101 players; obtain information in batches using these functions. As another example, if all scoreboards are obtained simultaneously when accessing ranking data, it is possible for a network delay to occur depending on the number of scoreboards. Do not obtain all the scoreboards at once, but obtain only the necessary scoreboards according to the user's scrolling operation.

Reloads

If a reload feature is implemented in ranking data accesses, design the feature so that the user cannot reload the data excessively often. For example, do not let multiple reloads occur consecutively when the reload button is held down.

Automatic Communication Processes

In principle, do not design the title so that ranking data is continually accessed without user operation as a trigger (since this will cause heavy loads on the server) unless the title can be significantly enhanced as a result. Note, however, that even when a positive effect can be hoped for, if replay data (attachments) will also be continually received, the default for this feature should be OFF and the user must choose to enter this mode. When in doubt, contact SCE.

4 FAQ

(1) Q: Can a player enter multiple scores on one scoreboard?

A: In principle, a user is allowed to register one score per scoreboard. If `sceNpScoreSetPlayerCharacterId()` is used, up to 10 scores can be registered to a scoreboard . However, note that this makes it difficult to realize friend rankings, etc.

(2) Q: Is there a filter against inappropriate text?

A: Input to the text field upon score registration is censored by the "Vulgarity Filter". The registration will result in an error if inappropriate words have been used. The functions `sceNpScoreCensorComment()`, `sceNpScoreCensorCommentAsync()`, `sceNpScoreSanitizeComment()`, and `sceNpScoreSanitizeCommentAsync()` are provided specifically for this purpose.

(3) Q: Can optional data be modified after ranking has been created?

A: The scoreboard has two settings - to overwrite a score only when its record is broken by the new score, or to always overwrite with the new score without exception. When set to the latter, registered content will always be overwritten and optional data can be modified at an arbitrary timing as well.

Note, however, that score registration processing entails heavy burden on the server's database. Please do not access unnecessarily.

(4) Q: If two users have the same score, how will score ranking be handled?

A: Two ranking systems are provided in every scoreboard - one that deems same scores to be of equal ranking, and one that favors a score according to a first-come-first-serve basis. *`rank`* and *`serialRank`* of the `SceNpScoreRankData` structure correspond to each, respectively. In serial ranking, the first score to be registered gains higher ranking over the other when 2 players are registering the same score.

(5) Q: Can size of game data that can be attached be changed?

A: Although there is a limit placed on the total size of game data to be attached, it is possible to change the limit placed on size of game data to be attached by each player.

Total size of game data for all boards should be kept under 1 GiB. Upper limits on the size of game data to be attached per player and the number of players to be registered can be changed per board. Therefore, customization of a scoreboard as in the following examples is permitted.

- Having 160 boards where players with the top 50 ranked scores can register 128 KiB of data each (= 1000 MiB)
- Having 320 boards where players with the top 25 ranked scores can register 128 KiB of data each (= 1000 MiB)
- Having 8 boards where players with the top 1000 ranked scores can register 64 KiB of data each and having 64 boards without any game data attached (= 500 MiB + 0 MiB)

Note that determination of total game data size entails a tradeoff with the time required to update ranking. The smaller the total size of game data, the shorter the time required to update ranking will be. Consider this point when setting specifications relating to data to be attached.