NP TSS 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 Title Small Storage library (NP TSS library) is a library for using the title small storage service provided by PSN℠. The title small storage service supports the distribution of special events, game balance adjustments, and data to be modified after the title's release (such as announcements). Using this service is much easier than creating a server just for the title when server maintenance is taken into account (server monitoring, domain name registration, SSL certificate updates). The following types of data can be distributed.

- 16 files of slot ID 0 to slot ID 15
  Maximum 64 KiB for slot 0, maximum 4 MiB for each file from slot 1 to slot 15
- Non-executable file
- Data deemed by SCE not to require QA after the title's release

Files (TSS files) stored on the TSS server with the title small storage service are identified by the NP Communication ID and slot ID. Because the title small storage service can freely change data with scp without permission by QA, it cannot be used - as a general rule - for data content requiring QA after the title launch. Notify SCE of the data content during the product planning stage. Data that requires QA approval must use the patching system instead. For the patch system, refer to the "Patch Overview" document.

Embedding into a Program

The NP TSS library is inclusive to the NP TUS library. Thus, the procedure for embedding it is the same as the NP TUS library.

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_TUS) != SCE_OK ) {
    // Error handling
}
```

Upon building the program, link libSceNpTus_stub.a.

Sample Program

A sample program that uses the NP TSS library is provided as follows.

sample_code/network/api_np/console/

This sample exemplifies the basic usage of the NP ScoreRanking, TUS, and TSS libraries.

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 Using the Library

Initialization

Implementation of the NP TSS library is inclusive to the NP TUS library. To use the NP TSS library, first initialize the NP library and then initialize the NP TUS library.

(1) Load the PRX

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP_TUS` specified as the module ID to load the PRX of the NP TUS library.

(2) Initialize the NP Library

Call `sceNpInit()` to initialize the NP library. 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 TUS Library

Call `sceNpTusInit()` to initialize the NP TUS library. This processing will also initialize the NP TSS library. With this process, a thread for inter-process communication is created in the application process and shell process respectively. For security reasons, NP TSS 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 = sceNpTusInit(SCE_KERNEL_DEFAULT_PRIORITY_USER,
SCE_KERNEL_THREAD_CPU_AFFINITY_MASK_DEFAULT, NULL);
if (ret < 0) {
    // Error handling
}
```

(4) Create a Title Context

Call `sceNpTusCreateTitleCtx()` to create a title context. At this time, if NULL is specified as an 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.

```
int ret, titleCtxId;

ret = sceNpTusCreateTitleCtx(NULL, NULL, NULL);
if (ret < 0) {
    // Error handling
}
titleCtxId = ret;
```

Communication Processing

(1) Create a Request ID

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

```
int ret, reqId, titleCtxId;

// Assuming that an appropriate value is stored in titleCtxId

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

(2) Obtain the TSS Data

Call `sceNpTssGetData()` to extract data from the TSS server. Specify the following arguments.

- Request ID
- Slot ID of file to obtain
- Pointer to structure for storing data status
- Size of structure to support future extension
- Ample buffer prepared by the application and the buffer size
- NULL (option to specify scope of obtainment, for example)

```
// Process communication
void *data;
SceSize dataSize;
SceNpTssSlotId slotId=SLOTID;
SceNpTssDataStatus dataStatus;
const char *ptr =NULL;
SceSize recvdSize=0;
SceSize totalSize=0;
SceSize recvSize=0;

do {
        ret = sceNpTssGetData(
              reqId,
              slotId,
              &dataStatus,
              sizeof(SceNpTssDataStatus),
              ptr,
              recvSize,
              NULL);
        if (ret < 0) {
                    // Error handling
                    goto error;
        }
        if (dataStatus.contentLength == 0){
              // Processing when there is no data
              goto finish;
        }
        if (ptr == NULL){
              ptr = malloc(dataStatus.contentLength);
              if (ptr == NULL){
                    // Error handling
```

```
                        goto error;
                    }
                    recvSize = BLOCKSIZE;
                }
                recvedSize += ret;
                ptr += ret;
    } while (recvedSize < dataStatus.contentLength);

    // Use the obtained data. Watch for buffer overflows

    error:
    if (ptr != NULL){
            free(ptr);
    }
    // Delete the request ID
    if (reqId > 0) {
            ret = sceNpTusDeleteRequest(reqId);
            printf("sceNpTusDeleteRequest () done. ret = 0x%x¥n", ret);
    }
```

If TSS file is not placed on the TSS server, `sceNpTssGetData()` will terminate normally and return 0 as the obtained data size. Specifically, `SCE_NP_TSS_STATUS_TYPE_OK` is stored in the *`statusCodeType`* member and 0 is stored in the *`contentLength`* member of `SceNpTssDataStatus`, and `sceNpTssGetData()` returns 0 for normal termination. Make sure to keep the above case in mind and program your application so that it does not malfunction in such cases.

`sceNpTssGetData()` performs synchronization and is blocking until data of the size specified in *`recvSize`* is obtained from the server. An asynchronous version of this function, `sceNpTssGetDataAsync()`, is also provided. Use whichever is appropriate.

Termination

(1) Destroy the Title Context

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

```
int ret;
uint32_t titleCtxId;

// Assuming that an appropriate value is stored in titleCtxId

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

(2) Terminate the NP TUS Library

Call `sceNpTusTerm()` to terminate the NP TUS library. This will also terminate the NP TSS library.

```
// The thread created for communication between processes will stop
sceNpTusTerm();
```

> **Note**
> When `sceNpTusTerm()` is called, the created title contexts and request IDs will be automatically deleted, however, it is recommended that this function be called after these are explicitly deleted from the application side.

(3) Terminate the NP Library

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

```
// Always succeeds
sceNpTerm();
```

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

3 Server Settings

Obtain an NP Communication ID

The NP TSS library requires an NP Communication ID to manage the title.

To obtain an NP Communication ID, register and submit an application for it on the PlayStation®Vita Developer Network website.

Configure the Title Small Storage Service

The title small storage service is available for the development environment, the QA environment, and the production environment. The only method available for transferring files to any of the servers is scp.

The scp accounts and required authentication files can be created with the Server Management Tools (SMT). For more information, refer to the "Server Management Tools Title Small Storage Service User's Guide" document.

In the development stage, first upload the files to the development environment for testing. When development of the application is complete, upload the files to the QA environment and submit the master files to SCE QA. Notify QA of the file specifications and file modification schedule by contacting SCE. If information of the QA environment (such as of the ranking server in the QA environment) is required for generating the files for title small storage, contact SCE as necessary regarding QA in the development environment.

After passing QA, upload the files to the production environment for game users. When generating data from environment-specific values (such as ranking values), remember to set the data based on the ranking server in the production environment (as opposed to the server in the development environment).

In the interest of preventing mishaps, the scp account for the development environment and QA environment is separate from the scp account for the production environment.

4 Notes

Delays Due to Caching

The title small storage service uses CDN (Contents Delivery Network), which makes the update timing of contents dependent on the terminal server accessed by the user. The delay depends on the life of the cache on the terminal server, but can be up to approximately one hour.

Using Versions for File Verification

The title small storage service does not use the concept of versions in the system. Files are checked in the authentication process using SSL server certificates, but are not checked for consistency. For these reasons, it is recommended for the application to make use of version and parity values in the file format.

Data Security

In the title small storage service, the client carries out server authentication, but the server does not carry out client authentication. For this reason, the possibility of server spoofing is low, but it is possible for data on the server to be downloaded to a PC (for example). Therefore do not put secret keys and other data that should not be available to users on the server.