NP TSS Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Datatypes

SceNpTssSlotId

TSS slot ID type definition

Definition

```
#include <np.h>
typedef SceInt32 SceNpTssSlotId;
```

Description

This ID datatype is for specifying a TSS slot.

SceNpTssIfModifiedSinceParam

Structure for specifying time condition

Definition

```
#include <np.h>
typedef enum{
        SCE_NP_TSS_IFTYPE_IF_MODIFIED_SINCE,
        SCE_NP_TSS_IFTYPE_IF_RANGE
} SceNpTssIfType;

typedef struct SceNpTssIfModifiedSinceParam{
        SceInt32 ifType;
        SceUInt8 padding[4];
        SceRtcTick lastModified;
} SceNpTssIfModifiedSinceParam;
```

Members

ifType	Specify the condition type Specify one of the following SCE_NP_TSS_IFTYPE_IF_MODIFIED_SINCE SCE_NP_TSS_IFTYPE_IF_RANGE
padding	Padding
lastModified	Specify time to make evaluation Because evaluation is carried out according to the time on the server, do not use time obtained locally, use the time of the TSS server obtainable with the *lastModified* member of SceNpTssDataStatus

Description

This structure is for specifying the time condition when obtaining TSS data with sceNpTssGetData().

When SCE_NP_TSS_IFTYPE_IF_MODIFIED_SINCE is specified to *ifType*, SCE_NP_TSS_STATUS_TYPE_NOT_MODIFIED will be stored in the *statusCodeType* member and 0 will be stored in the *contentLength* member of SceNpTssDataStatus if the time at which the TSS file was last updated on the server is equal to or older than the time specified to *lastModified*.

Use SCE_NP_TSS_IFTYPE_IF_MODIFIED_SINCE when caching TSS data to verify its validity.

When SCE_NP_TSS_IFTYPE_IF_RANGE is specified to *ifType*, the time at which the TSS file was last updated on the server is compared to the time specified to *lastModified*, and operation will be as follows.

(a) When the time at which the TSS file was last updated on the server is equal to or older than the time specified to *lastModified*:
Data of the range specified by the *offset* and *lastByte* members of SceNpTssGetDataOptParam will be obtained.

(b) When the time at which the TSS file was last updated on the server is newer than the time specified to *lastModified*:
The range specification of SceNpTssGetDataOptParam will be ignored and the entire file will be obtained.

Use SCE_NP_TSS_IFTYPE_IF_RANGE to obtain a specific range of the TSS file to avoid mixing TSS files of different versions. For example, when structuring the TSS file to store data at the beginning of the file, first obtain the beginning section where the data is placed. Then specify the time of obtainment when the file was last updated to *lastModified* and obtain the range with SCE_NP_TSS_IFTYPE_IF_RANGE. When SCE_NP_TSS_STATUS_TYPE_PARTIAL is stored to *statusCodeType* of SceNpTssDataStatus, this will mean the specified range of data was properly obtained. When SCE_NP_TSS_STATUS_TYPE_OK is stored to *statusCodeType*, assume TSS data was updated and obtain data from the beginning of the file again.

SceNpTssGetDataOptParam

Extended parameters for receiving data conditionally

Definition

```
#include <np.h>
typedef struct SceNpTssGetDataOptParam{
        SceSize size;
        SceOff *offset;
        SceOff *lastByte;
        SceNpTssIfModifiedSinceParam *ifParam;
} SceNpTssGetDataOptParam;
```

Members

size	Size of the structure Specify `sizeof(SceNpTssGetDataOptParam)`
offset	To specify the position from which to obtain TSS data Specify a pointer to the variable storing that byte count Specify NULL when not required
lastByte	If the end of the data to obtain is known Specify a pointer to the variable storing that byte count Specify NULL when not required
ifParam	Specify a pointer to the structure representing the time condition Specify NULL when not required

Description

This structure is for specifying conditions upon obtaining TSS data with `sceNpTssGetData()`.

For *offset*, specify the starting position of the data to obtain.

For *lastByte*, specify the end position of the data to obtain.

For *ifParam*, specify the condition that uses the last update time.

Specify NULL when the member specification is not required.

Moreover, since HTTP communication entails a certain amount of overhead, such as, the HTTP header, it is often more efficient to specify NULL to *lastByte* if *lastByte* is not determined before obtaining data and to disconnect once the required information is obtained.

SceNpTssDataStatus

Structure representing status of TSS data

Definition

```
#include <np.h>
typedef enum {
        SCE_NP_TSS_STATUS_TYPE_OK,
        SCE_NP_TSS_STATUS_TYPE_PARTIAL,
        SCE_NP_TSS_STATUS_TYPE_NOT_MODIFIED
} SceNpTssStatusCodeType;

typedef struct SceNpTssDataStatus{
        SceRtcTick lastModified;
        SceInt32 statusCodeType;
        SceSize contentLength;
} SceNpTssDataStatus;
```

Members

lastModified	Last update time
statusCodeType	When specifying condition with SceNpTssGetDataOptParam, the result of that condition evaluation will be stored
contentLength	The byte size of the data to receive from the server will be stored

Description

This structure represents status of the TSS data.

For *lastModified*, the time at which the TSS file uploaded to the server was last updated will be stored. For *statusCodeType*, the result of the condition evaluation upon using SceNpTssGetDataOptParam will be stored.

For *contentLength*, the value to be stored will vary according to *statusCodeType* as follows.

Value of *statusCodeType*	**Description and value stored in *contentLength***
SCE_NP_TSS_STATUS_TYPE_OK	Condition was not specified, or specified condition did not match. Size of the entire file will be stored in *contentLength*.
SCE_NP_TSS_STATUS_TYPE_PARTIAL	Received data of the range specified by *offset* and *lastbyte* members of SceNpTssGetDataOptParam. Byte size of the specified range will be stored in *contentLength*.
SCE_NP_TSS_STATUS_TYPE_NOT_MODIFIED	The last update time of the TSS file placed on the server is equal to or older than the time specified to the SCE_NP_TSS_IFTYPE_IF_MODIFIED_SINCE type of the *ifParam* member of SceNpTssGetDataOptParam. 0 will be stored in *contentLength*.

Communication Processing Functions

sceNpTssGetSmallStorage

Obtain TSS data (synchronous)

Definition

```
#include <np.h>
int sceNpTssGetSmallStorage(
        SceInt32 reqId,
        void *data,
        SceSize maxSize,
        SceSize *contentLength,
        void *option
);
```

Arguments

reqId	Request ID (IN)
data	Pointer to the memory to store the TSS data (OUT)
maxSize	Size of the memory specified to *data*. Maximum 64KiB (IN)
contentLength	Data size stored in *data* (OUT)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 upon normal termination.

Returns a negative value for an error. The main error codes are shown below. (The application must not malfunction even if other error codes are returned.)

Value	(Number)	Description
SCE_NP_COMMUNITY_ERROR_NOT_INITIALIZED	0x80550702	Used before sceNpTusInit() was called
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	Value other than NULL was specified to *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_BODY_TOO_LARGE	0x80550709	Data of a size larger than *maxSize* is on the server
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified to *data*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This API gets the data assigned to an NP Communication ID (one file of maximum 64KiB) from the TSS server. If the size of the data on the server is *maxSize* or smaller, the data will be stored to the memory specified in *data* and the total size of the data obtained will be stored to *contentLength*. For the memory specified in *data*, a size of at least *maxSize* must always be allocated.

This function performs synchronous processing. In other words, it is blocking until all the data is obtained. Upon returning from this function, call sceNpTusDeleteRequest() to destroy the request used (Functions of the NP TUS library are used to handle the context of the NP TSS library).

Examples

```
int ret;
int reqId, titleCtxId;
void *data=NULL;
SceSize dataSize=0;

// The module is the same as the one for NP TUS
ret = sceSysmoduleLoadModule(SCE_SYSMODULE_NP_TUS);
if (ret < 0) {
   // Error handling
}
// Common initialization processing with NP TUS
ret = sceNpTusInit();
if (ret < 0) {
   // Error handling
}
// Context operation is the same as with NP TUS
ret = sceNpTusCreateTitleCtx(NULL, NULL, NULL);
if (ret < 0){
   // Error handling
}
titleCtxId = ret;

ret = sceNpTusCreateRequest(titleCtxId);
if (ret < 0){
   // Error handling
}
reqId = ret;
data = malloc(SCE_NP_TSS_MAX_SIZE);
if (data == NULL){
   // Error handling
}
ret = sceNpTssGetSmallStorage(reqId,
                              data,
                              SCE_NP_TSS_MAX_SIZE,
                              &dataSize,
                              NULL);
If (ret < 0){
   // Error handling
}
// Use the data obtained. Watch for buffer overflows
```

Notes

If the file is not on the server, this will be handled as though a file of 0 bytes were on the server. This means that 0 will be stored in *contentLength* and 0 will return for normal termination. This situation can occur with problems in actual operation, so the application must be designed not to hang up even in such situations.

This function is an API for obtaining slot 0 of the TSS file. As an alternate to this function, `sceNpTssGetData()` API, which obtains TSS data from slot 0 to slot 15, has been implemented. Use `sceNpTssGetData()` in the future.

See Also

`sceNpTusInit()`,`sceNpTusCreateRequest()`,`sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`,`sceNpTssGetData()`

sceNpTssGetSmallStorageAsync

Obtain TSS data (asynchronous)

Definition

```
#include <np.h>
int sceNpTssGetSmallStorageAsync(
        SceInt32 reqId,
        void *data,
        SceSize maxSize,
        SceSize *contentLength,
        void *option
);
```

Arguments

reqId	Request ID (IN)
data	Pointer to the memory to store the TSS data (OUT)
maxSize	Size of the memory specified to *data*. Maximum 64KiB (IN)
contentLength	Data size stored in *data* (OUT)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 upon normal termination.

Returns a negative value for an error. The main error codes are shown below. (The application must not malfunction even if other error codes are returned.)

Value	(Number)	Description
SCE_NP_COMMUNITY_ERROR_NOT_INITIALIZED	0x80550702	Used before sceNpTusInit() was called
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	Value other than NULL was specified to *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_BODY_TOO_LARGE	0x80550709	Data of a size larger than *maxSize* is on the server
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified to *data*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This API gets the data assigned to an NP Communication ID (one file of maximum 64KiB) from the TSS server. If the size of the data on the server is *maxSize* or smaller, the data will be stored to the memory specified in *data* and the total size of the data obtained will be stored to *contentLength*. For the memory specified in *data*, a size of at least *maxSize* must always be allocated.

This function performs asynchronous processing. Once it starts a communication processing, the function returns without waiting to obtain the result of the processing from the server. The result can be obtained with sceNpTusWaitAsync() or sceNpTusPollAsync(). After receiving the result from one of these functions, destroy the request used.

Notes

If a file is not placed on the server, this function will behave as though a file of 0 bytes is placed on the server. This means that 0 will be stored in *contentLength* and 0 will return (for normal termination). This situation can occur with problems in actual operation, so the application must be designed not to hang up even in such situations.

This function is an API for obtaining slot 0 of the TSS file. As an alternate to this function, `sceNpTssGetDataAsync()` API, which obtains TSS data from slot 0 to slot 15, has been implemented. Use `sceNpTssGetDataAsync()` in the future.

See Also

`sceNpTusCreateRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTssGetDataAsync()`

sceNpTssGetData

Obtain TSS data of the specified slot (synchronous)

Definition

```
#include <np.h>
int sceNpTssGetData (
        SceInt32 reqId,
        SceNpTssSlotId slotId,
        SceNpTssDataStatus *dataStatus,
        SceSize dataStatusSize,
        void *data,
        SceSize recvSize,
        SceNpTssGetDataOptParam *option
);
```

Arguments

reqId	Request ID (IN)
slotId	Slot ID of the TSS file to download. Specify 0 to 15 (IN)
dataStatus	Pointer to structure storing status of TSS data (OUT)
dataStatusSize	Size of structure storing status of TSS data (IN)
data	Pointer to area storing data to receive this time (OUT)
recvSize	Size of data to receive this time (IN)
option	Pointer to extended options. Specify NULL when not required (IN)

Return Values

Returns 0 upon normal termination.

Returns a negative value for an error. The main error codes are shown below. (The application must not malfunction even if other error codes are returned.)

Value	(Number)	Description
SCE_NP_COMMUNITY_ERROR_NOT_INITIALIZED	0x80550702	Used before sceNpTusInit() was called
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_BODY_TOO_LARGE	0x80550709	Size of the TSS file placed on the server is too large SCE_NP_TSS_MAX_SIZE is the maximum size for slot 0 and SCE_NP_TSS_EXTRA_SLOT_MAX_SIZE is the maximum size for slot 1 - 15
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified to *dataStatus*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This function downloads TSS data for the specified slot.

TSS Filename

NP Communication ID-slot ID (decimal).tss

Example: `ABCD01234_00-15.tss`

For *`slotId`*, specify the slot ID of the TSS file to download.

For *`dataStatus`*, specify the pointer to the structure storing the status of the TSS data and specify `sizeof(SceNpTssDataStatus)` to *`dataStatusSize`*.

For *`data`*, specify the area to store the received data and specify its size to *`recvSize`*.

The total size to receive will be stored in the *`contentLength`* member of the *`dataStatus`* structure. If this value is larger than the value specified to *`recvSize`*, this means that only a part of the data was received; call this function again to receive the rest of the data. From the second and subsequent function calls, the same value as the first call must be specified to *`reqId`*. Values for *`dataStatus`*, *`data`*, and *`recvSize`* can vary. Specification made to *`slotId`* and *`option`* in the second and subsequent function calls will be ignored. Whether the end of the data has been reached must be determined by the application based on *`contentLength`* and the size of received data.

When NULL is specified to *`data`*, only the data status will be stored in the *`dataStatus`* structure without receiving any data. Thus, the first function call can be used to obtain the total size of the data to receive using the data status; required memory can be allocated, and the second function call can be used to receive the actual data.

This function performs synchronous processing. In other words, it is blocking until all the data of the size specified in *`recvSize`* is obtained. When returning from this function, call `sceNpTusDeleteRequest()` to delete the used request. Functions of the NP TUS library are used to handle the context of the NP TSS library.

Examples

```
int ret;
int reqId, titleCtxId;
SceNpTssSlotId slotId=TARGET_SLOTID;
SceNpTssDataStatus dataStatus;
const char *ptr =NULL;
SceSize recvdSize=0;
SceSize totalSize=0;
SceSize recvSize=0;

// The module is the same as the one for NP TUS
ret = sceSysmoduleLoadModule(SCE_SYSMODULE_NP_TUS);
if (ret < 0) {
        // Error handling
}
// Common initialization processing with NP TUS
ret = sceNpTusInit();
if (ret < 0) {
    // Error handling
}
// Context operation is the same as with NP TUS
ret = sceNpTusCreateTitleCtx(NULL, NULL, NULL);
if (ret < 0){
    // Error handling
}
```

```
titleCtxId = ret;

ret = sceNpTusCreateRequest(titleCtxId);
if (ret < 0){
   // Error handling
}
reqId = ret;
do {
        ret = sceNpTssGetData(
              reqId,
              slotId,
              &dataStatus,
              sizeof(SceNpTssDataStatus),
              ptr,
              recvSize,
              NULL);
        if (ret < 0) {
              // Error handling
              goto error;
        }
        if (dataStatus.contentLength == 0){
              // Processing when file is not set
              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 data obtained. Watch for buffer overflows

error:
if (ptr != NULL){
        free(ptr);
}
if (reqId > 0) {
        ret = sceNpTusDeleteRequest(reqId);
        printf("sceNpTusDeleteRequest () done. ret = 0x%x\n", ret);
}
```


Notes

If the file is not on the server, this will be handled as though a file of 0 bytes were on the server. This means that `SCE_NP_TSS_STATUS_TYPE_OK` will be stored in the *`statusCodeType`* member and 0 will be stored in the *`contentLength`* member of *`dataStasus`*, and 0 will return for normal termination. This situation can occur with problems in actual operation, so the application must be designed not to hang up even in such situations.

See Also

`sceNpTusInit()`, `sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`

sceNpTssGetDataAsync

Obtain TSS data of the specified slot (asynchronous)

Definition

```
#include <np.h>
int sceNpTssGetDataAsync(
        SceInt32 reqId,
        SceNpTssSlotId slotId,
        SceNpTssDataStatus *dataStatus,
        SceSize dataStatusSize,
        void *data,
        SceSize recvSize,
        SceNpTssGetDataOptParam *option
);
```

Arguments

reqId	Request ID (IN)
slotId	Slot ID of the TSS file to download. Specify 0 to 15 (IN)
dataStatus	Pointer to structure storing status of TSS data (OUT)
dataStatusSize	Size of TSS data status (IN)
data	Pointer to area storing data to receive this time (OUT)
recvSize	Size of data to receive this time (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 upon normal termination.

Returns a negative value for an error. The main error codes are shown below. (The application must not malfunction even if other error codes are returned.)

Value	(Number)	Description
SCE_NP_COMMUNITY_ERROR_NOT_INITIALIZED	0x80550702	Used before sceNpTusInit() was called
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_BODY_TOO_LARGE	0x80550709	Size of the TSS file placed on the server is too large SCE_NP_TSS_MAX_SIZE is the maximum size for slot 0 and SCE_NP_TSS_EXTRA_SLOT_MAX_SIZE is the maximum size for slot 1 - 15
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified to *dataStatus*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This function downloads TSS data for the specified slot.

TSS Filename

NP Communication ID-slot ID (decimal).tss

Example: `ABCD01234_00-15.tss`

For *`slotId`*, specify the slot ID of the TSS file to download.

For *`dataStatus`*, specify the pointer to the structure storing the status of the TSS data and specify `sizeof(SceNpTssDataStatus)` to *`dataStatusSize`*.

For *`data`*, specify the area to store the received data and specify its size to *`recvSize`*.

The total size to receive will be stored in the *`contentLength`* member of the *`dataStatus`* structure. If this value is larger than the value specified to *`recvSize`*, this means that only a part of the data was received; call this function again to receive the rest of the data. From the second and subsequent function calls, the same value as the first call must be specified to *`reqId`*. Values for *`dataStatus`*, *`data`*, and *`recvSize`* can vary. Specification made to *`slotId`* and *`option`* in the second and subsequent function calls will be ignored. Whether the end of the data has been reached must be determined by the application based on *`contentLength`* and the size of received data.

When NULL is specified to *`data`*, only the data status will be stored in the *`dataStatus`* structure without receiving any data. Thus, the first function call can be used to obtain the total size of the data to receive using the data status; required memory can be allocated, and the second function call can be used to receive the actual data.

This function performs asynchronous processing. Once it starts a communication processing, the function returns without waiting to obtain the result of the processing from the server. The result can be obtained with `sceNpTusWaitAsync()` or `sceNpTusPollAsync()`. After receiving the result from one of these functions, destroy the request used.

Notes

If the file is not on the server, this will be handled as though a file of 0 bytes were on the server. This means that `SCE_NP_TSS_STATUS_TYPE_OK` will be stored in the *`statusCodeType`* member and 0 will be stored in the *`contentLength`* member of *`dataStasus`*, and 0 will return for normal termination. This situation can occur with problems in actual operation, so the application must be designed not to hang up even in such situations.

See Also

`sceNpTusCreateRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`

Constants

SCE_NP_TSS_MAX_SIZE

Maximum size of TSS data for slot 0

Definition

```
#include <np.h>

#define SCE_NP_TSS_MAX_SIZE  (64 * 1024U)
```

Description

This constant indicates the maximum size in bytes of TSS data for slot 0.

See Also

sceNpTssGetSmallStorage(),sceNpTssGetSmallStorageAsync(),sceNpTssGetData(), sceNpTssGetDataAsync()

SCE_NP_TSS_EXTRA_SLOT_MAX_SIZE

Maximum size of TSS data for extended slots

Definition

```
#include <np.h>

#define SCE_NP_TSS_EXTRA_SLOT_MAX_SIZE     (4 * 1024 * 1024U)
```

Description

This constant represents the maximum size in bytes of TSS data for slots 1 to 15.

See Also

`sceNpTssGetData(),sceNpTssGetDataAsync()`