NP TUS Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Datatypes

SceNpTusSlotId

Type definition for a TUS slot ID

Definition

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

Description

This datatype is for specifying the ID of a TUS slot (TUS variable or TUS data). Slots are prepared per NP Communication ID on the title user storage server according to settings that are made in advance.

SceNpTusVirtualUserId

Type definition for a virtual user ID

Definition

```
#include <np.h>
#define SCE_NP_ONLINEID_MIN_LENGTH        3
#define SCE_NP_ONLINEID_MAX_LENGTH        16
typedef struct SceNpTusVirtualUserId {
        char data[SCE_NP_ONLINEID_MAX_LENGTH];
        char term;
        char dummy[3];
} SceNpTusVirtualUserId;
```

Description

This datatype is for specifying the ID of a virtual user. It is the same datatype as the Online ID. However, the virtual user ID is not pursuant to the specifications of the Online ID (starts from the underscore), and the virtual user ID cannot be specified in functions where the Online ID should be specified.

The virtual user is prepared per NP Communication ID, on the title user storage server, according to settings made in advance.

SceNpTusVariable

Structure for representing a TUS variable

Definition

```
#include <np.h>
typedef struct SceNpTusVariable{
        SceNpId ownerId;
        SceInt32 hasData;
        SceUInt8 pad[4];
        SceRtcTick lastChangedDate;
        SceNpId lastChangedAuthorId;
        SceInt64 variable;
        SceInt64 oldVariable;
        SceUInt8 reserved[16];
} SceNpTusVariable;
```

Members

ownerId	Owner
hasData	Flag indicating whether a value has been set. 1 if set, 0 if not
pad	Padding
lastChangedDate	Last update date
lastChangedAuthorId	Last updated by
variable	Currently set value
oldVariable	Previously set value
reserved	Area for future extension

Description

This structure represents the status and value of a TUS variable.

oldVariable can only be obtained with the following APIs.

- `sceNpTusAddAndGetVariable()`, `sceNpTusAddAndGetVariableVUser()`
- `sceNpTusAddAndGetVariableAsync()`, `sceNpTusAddAndGetVariableVUserAsync()`
- `sceNpTusTryAndSetVariable()`, `sceNpTusTryAndSetVariableVUser()`
- `sceNpTusTryAndSetVariableAsync()`, `sceNpTusTryAndSetVariableVUserAsync()`

SceNpTusDataInfo

Structure for representing the accessory information of a TUS data

Definition

```
#include <np.h>

#define SCE_NP_TUS_DATA_INFO_MAX_SIZE      (384U)
typedef struct SceNpTusDataInfo{
        SceSize infoSize;
        SceUInt8 pad[4];
        SceUInt8 data[SCE_NP_TUS_DATA_INFO_MAX_SIZE];
} SceNpTusDataInfo;
```

Members

infoSize	Size of valid data within *data*
pad	Padding
data	Area for storing the accessory information of a TUS data

Description

This structure stores the accessory information of a TUS data.

It is assumed that accessory information will be used as index data. Functions are provided whereby accessory information of multiple TUS data, for a single user, can be obtained. Functions are also provided whereby the accessory information of 1 TUS data for multiple users can be obtained all at once.

Notes

The maximum size of accessory information is `SCE_NP_TUS_DATA_INFO_MAX_SIZE`. However, because it is directly connected to the storage size used by the server, avoid filling its unused space with 0s or with empty spaces, and only specify the required size in *infoSize*.

See Also

`SceNpTusDataStatus`, `SCE_NP_TUS_DATA_INFO_MAX_SIZE`

SceNpTusDataStatus

Structure for representing the status of TUS data

Definition

```
#include <np.h>
typedef struct SceNpTusDataStatus{
        SceNpId ownerId;
        SceInt32 hasData;
        SceRtcTick lastChangedDate;
        SceNpId lastChangedAuthorId;
        void *data;
        SceSize dataSize;
        SceUInt8 pad[4];
        SceNpTusDataInfo info;
} SceNpTusDataStatus;
```

Members

ownerId	Owner
hasData	Flag indicating whether data has been set. 1 if set, 0 if not
lastChangedDate	Last update time
lastChangedAuthorId	Last updated by
data	Pointer to area where the data is stored
dataSize	Total size of TUS data
pad	Padding
info	Area storing the accessory information

Description

This structure is for representing the status and data of a TUS data.

`data` will be NULL when obtaining the status of TUS data. Actual data will be set to this member when a function, such as `sceNpTusGetData()`, is used to obtain the data itself. When obtaining data in installments using multiple function calls, the pointer set in the first call will be returned in the subsequent calls.

SceNpTusGetFriendsVariableOptParam

Extended parameters for obtaining friends' TUS variables

Definition

```
#include <np.h>
typedef struct SceNpTusGetFriendsVariableOptParam {
        SceSize size;
        SceUInt32 *startSerialRank;
        SceUInt32 *hits;
} SceNpTusGetFriendsVariableOptParam;
```

Members

size	Size of this structure (IN)
startSerialRank	Pointer to start position (ranking among friends) to obtain (IN). If NULL is specified, obtaining will be performed from the first ranked friend
hits	Total number of friends with TUS variables registered in the target slot (OUT)

Description

This is an option structure for handling the 100th and later ranked TUS variables among friends with `sceNpTusGetFriendsVariable()` and `sceNpTusGetFriendsVariableAsync()`.

For *size*, specify `sizeof(SceNpTusGetFriendsVariableOptParam)`.

Notes

Web APIs are provided for the purpose of preparing test environments where more than 100 friends have TUS variables registered. For details, refer to the "TUS Management Overview" document.

SceNpTusGetFriendsDataStatusOptParam

Extended parameters for obtaining friends' TUS data statuses

Definition

```
#include <np.h>
typedef struct SceNpTusGetFriendsDataStatusOptParam {
        SceSize size;
        SceUInt32 *startSerialRank;
        SceUInt32 *hits;
} SceNpTusGetFriendsDataStatusOptParam;
```

Members

size	Size of this structure (IN)
startSerialRank	Pointer to start position (ranking among friends) to obtain (IN). If NULL is specified, obtaining will be performed from the first ranked friend
hits	Total number of friends with TUS data registered in the target slot (OUT)

Description

This is an option structure for handling the 100th and later ranked TUS data statuses among friends with `sceNpTusGetFriendsDataStatus()` and `sceNpTusGetFriendsDataStatusAsync()`.

For *size*, specify `sizeof(SceNpTusGetFriendsDataStatusOptParam)`.

Notes

Web APIs are provided for the purpose of preparing test environments where more than 100 friends have TUS data statuses registered. For details, refer to the "TUS Management Overview" document.

Initialization and Termination Functions

sceNpTusInit

Initialize the NP TUS library

Definition

```
#include <np.h>
int sceNpTusInit(
        SceInt32 threadPriority,
        SceInt32 cpuAffinityMask,
        void *option
);
```

Arguments

threadPriority	Priority of the thread created for communication between processes
cpuAffinityMask	CPU affinity mask of the thread used for communication between processes
option	Option reserved for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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_ALREADY_INITIALIZED`	0x80550701	Already initialized. `sceNpTusInit()` may have already been executed. Check the calling order

Description

This function initializes the NP TUS library. A thread on a system process will be started for handling communication with the server. A thread on the game process will be started for handling communication between processes, with the specified priority and affinity mask. Memory required for process communication will be allocated internally. This function must be called before using the NP TUS library.

Because communication is performed on a system process, processing on the game process is limited to communication between processes. The speed of the response will also affect the processing speed on the system process side. Thus, set `SCE_KERNEL_THREAD_CPU_AFFINITY_MASK_DEFAULT` for the affinity mask and ensure that whenever a core becomes free, a processing can be assigned to it.

Notes

This function is not multithread safe.

See Also

`sceNpTusTerm()`

sceNpTusTerm

Terminate the NP TUS library

Definition

```
#include <np.h>
int sceNpTusTerm(void);
```

Arguments

None

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized

Description

This function terminates the NP TUS library.

Notes

This function is not multithread safe.

See Also

`sceNpTusInit()`

Context Operation Functions

sceNpTusCreateTitleCtx

Create a title context

Definition

```
#include <np.h>
int sceNpTusCreateTitleCtx(
        const SceNpCommunicationId *communicationId,
        const SceNpCommunicationPassphrase *passphrase,
        const SceNpId *selfNpId,
);
```

Arguments

communicationId	Pointer to an NP Communication ID. If NULL was specified, the value of `SceNpCommunicationConfig` set with `sceNpInit()` is used
passphrase	Pointer to a structure storing authentication information of the title user storage allocated to the NP Communication ID. If NULL was specified, the value of `SceNpCommunicationConfig` set with `sceNpInit()` is used
selfNpId	Pointer to a structure storing the NP ID of the logging-in user. If NULL was specified, the NP ID obtained with `sceNpManagerGetNpId()` is used

Return Values

Returns the title context ID (>0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_OUT_OF_MEMORY	0x80550703	Not enough free memory
SCE_NP_COMMUNITY_ERROR_TOO_MANY_OBJECTS	0x80550706	Creating more than 32 title contexts at one time was attempted Check to see that `sceNpTusDeleteTitleCtx()` was called as necessary

Description

This function creates a title context. A title context is used to create a request ID, which is then used to perform communication processing of the NP TUS library.

For *passphrase*, specify a pointer to the authentication information (NP communication passphrase) allocated to the specified NP Communication ID.

Notes

Up to 32 title contexts can be created at one time, however, try to minimize the number of title contexts by reusing one whenever possible.

All title contexts will automatically be deleted when `sceNpTusTerm()` is called. As a general rule, however, program your application so that title contexts are explicitly destroyed using `sceNpTusDeleteTitleCtx()` before `sceNpTusTerm()` is called.

See Also

`sceNpTusDeleteTitleCtx()`,`SCE_NP_TUS_MAX_CTX_NUM`

sceNpTusDeleteTitleCtx

Delete a title context

Definition

```
#include <np.h>
int sceNpTusDeleteTitleCtx(
        SceInt32 titleCtxId
);
```

Arguments

titleCtxId	ID of title context to delete

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	Specified ID does not exist

Description

This function deletes a title context.

All title contexts will automatically be deleted when `sceNpTusTerm()` is called. As a general rule, however, program your application so that title contexts are explicitly destroyed using this function before `sceNpTusTerm()` is called.

See Also

`sceNpTusCreateTitleCtx()`

sceNpTusCreateRequest

Create a request ID

Definition

```
#include <np.h>
int sceNpTusCreateRequest(
        SceInt32 titleCtxId
);
```

Arguments

titleCtxId	Title context ID

Return Values

Returns a request ID (>0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_OUT_OF_MEMORY`	0x80550703	Not enough free memory
`SCE_NP_COMMUNITY_ERROR_TOO_MANY_OBJECTS`	0x80550706	Creating more than 32 request IDs at one time was attempted Make sure that you have not forgotten to call `sceNpTusDeleteRequest()`
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	Specified ID does not exist

Description

This function creates a request ID to be used for the NP TUS library.

A request ID is used up per communication processing function. It must be created before calling a communication processing function; and it must be deleted by `sceNpTusDeleteRequest()` after it has been used.

Up to 32 request IDs can exist at one time.

Notes

All request IDs will automatically be deleted when `sceNpTusTerm()` is called. As a general rule, however, program your application so that request IDs are explicitly destroyed using `sceNpTusDeleteRequest()` before `sceNpTusTerm()` is called.

See Also

`sceNpTusDeleteRequest(),SCE_NP_TUS_MAX_CTX_NUM`

sceNpTusDeleteRequest

Delete a request ID

Definition

```
#include <np.h>
int sceNpTusDeleteRequest (
        SceInt32 reqId
);
```

Arguments

reqId	Request ID to delete

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_OUT_OF_MEMORY`	0x80550703	Not enough free memory
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	Specified ID does not exist

Description

This function deletes a used request ID.

Up to 32 request IDs can exist at one time.

All request IDs will automatically be deleted when `sceNpTusTerm()` is called. As a general rule, however, program your application so that request IDs are explicitly destroyed using this function before `sceNpTusTerm()` is called.

See Also

`sceNpTusCreateRequest()`

sceNpTusSetTimeout

Set forced timeout time

Definition

```
#include <np.h>
int sceNpTusSetTimeout (
        SceInt32 ctxId,
        SceInt32 resolveRetry,
        SceUInt32 resolveTimeout,
        SceUInt32 connTimeout,
        SceUInt32 sendTimeout,
        SceUInt32 recvTimeout
);
```

Arguments

ctxId	Title context ID, or request ID
resolveRetry	Name resolution retry count
resolveTimeout	Name resolution timeout time (in microseconds)
connTimeout	Timeout time when connecting (in microseconds)
sendTimeout	Sending timeout time (in microseconds)
recvTimeout	Receiving timeout time (in microseconds)

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	Value specified for the number of retries or timeout time is too small
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	Specified ID does not exist

Description

This function sets the timeout time for communication processing.

To *ctxId*, specify the ID of the targeted title context ID or request ID, and set name resolution retry count and communication processing timeout times in microseconds.

When the title context is the target, this timeout value will be used as the default value of the requests created from the title context. However, it will not affect the timeout of requests that have already been created.

Notes

Instead of calling this function to set the timeout time, use the default timeout time wherever possible. If you are considering the use of this function to prevent user operation from becoming impossible for a long period of time, consider adopting a user cancellation scheme using `sceNpTusAbortRequest()` first.

The default timeout values are as follows.

Name resolution retry count	5 times
Name resolution timeout time	1 second
Timeout time when connecting	30 seconds
Sending timeout time	60 seconds
Receiving timeout time	60 seconds

The timeout value cannot be set in total time. This is because the waiting time of connection processing is included in the API calling time, in accordance with the newly introduced concept of intermittent connection. If processing were to be aborted when timeout occurs based on total time count, timeout would occur during time-consuming operations by the user, such as SSID and password input. Therefore, if setting timeout using a unique method, it is recommended not to count on a total time basis, but rather to set socket layer timeout, as in this function.

2 seconds can be assumed to be the normal time required for communication processing. However, depending on the user's network environment, it is possible for a large delay to occur; therefore we recommend setting the timeout value to 30 seconds or more. Note that this value of 30 seconds is not appropriate for general network programming. For example, in a situation where the clients communicate directly with each other, consider differences in client environments (differences in load times depending on hardware differences, for example), and keep in mind that it is necessary to set a longer timeout.

See Also

`sceNpTusCreateTitleCtx()`, `sceNpTusCreateRequest()`, `sceNpTusAbortRequest()`

sceNpTusChangeModeForOtherSaveDataOwners

Set operation restriction applied when another owner's save data is loaded

Definition

```
#include <np.h>
int sceNpTusChangeModeForOtherSaveDataOwners (
        SceInt32 ctxId,
        SceInt32 mode
);
```

Arguments

ctxId	Title context ID, or request ID
mode	Value representing the mode to be set

Return Values

Returns 0 upon normal termination.

Returns a negative value upon errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	Nonexistent value was specified for *mode*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	Specified ID does not exist

Description

This function is used to prevent the registration to TUS server with another owner's save data and keep the user from saving the TUS data and TUS variables into another owner's save data.

Since NP TUS is a system that is closed by each application, there is no TRC (Technical Requirements Checklist) requirement that prohibits the use of TUS in combination with another owner's save data. Performing a test may be difficult if using this function; thus, make sure to use this function only when it is really required according to the usage methods and policies of TUS for an application.

For example, if the account ID written in a memory card is different from the one written in the main unit, the memory card must be formatted before it is used with the main unit.

Therefore, it is not necessary to use this function when saving the save data in a memory card. Also, this function is not required if the TUS variable or the TUS data is not saved in the save data. For details, refer to the "Owner Identification When Inserting Memory Cards" chapter in the "System Software Overview" document.

For *ctxId*, specify the Title context ID or the request ID and for mode, specify the operation performed when another owner's save data is loaded.

When a title context is specified, the specified value will be used as the default value of the requests created from the title context. However, requests that already exist will not be affected.

Specify one of the following values in *mode*.

mode is handled as a bit pattern not a signed integer.

Value	Description
`SCE_NP_TUS_BINDMODE_ALL_FORBIDDEN`	When the save data is owned by another user, both reference from and update to the server are prohibited
`SCE_NP_TUS_BINDMODE_RDONLY`	When the save data is owned by another user, reference from the server is only allowed
`SCE_NP_TUS_BINDMODE_WRONLY`	When the save data is owned by another user, update to the server is only allowed
`SCE_NP_TUS_BINDMODE_RDWR`	No restriction is applied even when the save data is owned by another user
`SCE_NP_TUS_BINDMODE_DEFAULT`	Default setting. `SCE_NP_TUS_BINDMODE_RDWR` is set

Reference APIs and update APIs are as follows.

Reference APIs

`sceNpTusGetMultiSlotVariable()`, `sceNpTusGetMultiSlotVariableVUser()`, `sceNpTusGetMultiSlotVariableAsync()`, `sceNpTusGetMultiSlotVariableVUserAsync()`, `sceNpTusGetMultiUserVariable()`, `sceNpTusGetMultiUserVariableVUser()`, `sceNpTusGetMultiUserVariableAsync()`, `sceNpTusGetMultiUserVariableVUserAsync()`, `sceNpTusGetData()`, `sceNpTusGetDataVUser()`, `sceNpTusGetDataAsync()`, `sceNpTusGetDataVUserAsync()`, `sceNpTusGetMultiSlotDataStatus()`, `sceNpTusGetMultiSlotDataStatusVUser()`, `sceNpTusGetMultiSlotDataStatusAsync()`, `sceNpTusGetMultiSlotDataStatusVUserAsync()`, `sceNpTusGetMultiUserDataStatus()`, `sceNpTusGetMultiUserDataStatusVUser()`, `sceNpTusGetMultiUserDataStatusAsync()`, `sceNpTusGetMultiUserDataStatusVUserAsync()`

Update APIs

`sceNpTusSetMultiSlotVariable()`, `sceNpTusSetMultiSlotVariableVUser()`, `sceNpTusSetMultiSlotVariableAsync()`, `sceNpTusSetMultiSlotVariableVUserAsync()`, `sceNpTusAddAndGetVariable()`, `sceNpTusAddAndGetVariableVUser()`, `sceNpTusAddAndGetVariableAsync()`, `sceNpTusAddAndGetVariableVUserAsync()`, `sceNpTusTryAndSetVariable()`, `sceNpTusTryAndSetVariableVUser()`, `sceNpTusTryAndSetVariableAsync()`, `sceNpTusTryAndSetVariableVUserAsync()`, `sceNpTusDeleteMultiSlotVariable()`, `sceNpTusDeleteMultiSlotVariableVUser()`, `sceNpTusDeleteMultiSlotVariableAsync()`, `sceNpTusDeleteMultiSlotVariableVUserAsync()`, `sceNpTusSetData()`, `sceNpTusSetDataVUser()`, `sceNpTusSetDataAsync()`, `sceNpTusSetDataVUserAsync()`, `sceNpTusDeleteMultiSlotData()`, `sceNpTusDeleteMultiSlotDataVUser()`, `sceNpTusDeleteMultiSlotDataAsync()`, `sceNpTusDeleteMultiSlotDataVUserAsync()`

When a restriction is applied, an error value `SCE_NP_COMMUNITY_ERROR_TUS_INVALID_SAVEDATA_OWNER` is received as a return value for synchronous functions or as the result obtained with `sceNpTusWaitAsync()` or `sceNpTusPollAsync()` for asynchronous functions. No particular message is provided for this error; therefore, application should be designed to handle this error and inform the user that the application features are restricted because the save data is owned by another user.

Notes

It is possible to enable reference from and update to the TUS server irrespective of the owner of the save data by setting Game - Fake Trophy Earning in ★Debug Settings of system software to On. Whereas this feature can be used in the development environment and QA environment, it is disabled in the production environment.

For ★Debug Settings, refer to the "★Debug Settings Functions" section of the "System Software Overview" document.

See Also

`sceNpTusCreateTitleCtx(),sceNpTusCreateRequest()`

sceNpTusAbortRequest

Abort communication processing

Definition

```
#include <np.h>
int sceNpTusAbortRequest(
        SceInt32 reqId,
);
```

Arguments

reqId	ID of the request to abort

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *reqId* does not exist

Description

This function aborts the specified communication processing.

For *reqId*, specify the request ID for which communication processing is to be aborted.

sceNpTusWaitAsync, sceNpTusPollAsync

Get execution result of an asynchronous processing

Definition

```
#include <np.h>
int sceNpTusWaitAsync (
        SceInt32 reqId,
        SceInt32 *result
);

int sceNpTusPollAsync (
        SceInt32 reqId,
        SceInt32 *result
);
```

Arguments

reqId	Request ID
result	Pointer to variable for storing the communication processing

Return Values

Value	Description
0	Asynchronous processing completed
1	Asynchronous processing is still being executed

Returns a negative value for errors.

Description

This function obtains the execution result of an asynchronous communication processing.

For *reqId*, specify the ID of the request started by an asynchronous function [`sceNpTus`*`XXX`*`Async()`].

`sceNpTusWaitAsync()` waits for the communication processing to complete if it hasn't already. It stores the result of the processing in `*`*`result`* and returns 0.

`sceNpTusPollAsync()` returns immediately if the communication processing has not completed, with a return value of 1. In this case, the value of `*`*`result`* does not change. If the processing has already completed, the function stores the result in `*`*`result`* and returns 0.

See Also

`sceNpTusCreateRequest()`

TUS Variable Operation Functions

sceNpTusSetMultiSlotVariable, sceNpTusSetMultiSlotVariableVUser

Write values to multiple TUS variables of 1 user (synchronous)

Definition

```
#include <np.h>

int sceNpTusSetMultiSlotVariable (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        const SceInt64 variableArray[],
        SceInt32 arrayNum,
        void *option
);

int sceNpTusSetMultiSlotVariableVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        const SceInt64 variableArray[],
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the write target (IN)
targetVirtualUserId	ID of virtual user who is the write target (IN)
slotIdArray	Pointer to the array storing the write target slot IDs (IN)
variableArray	Pointer to the array storing the 64-bit integers to write (IN)
arrayNum	Valid number of elements for *slotIdArray*. Maximum is 64 (IN)
option	Option reserved for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	A negative value was specified in the array specified for *`slotIdArray`*, 0 was specified for *`arrayNum`*, or value other than NULL was specified for *`option`*
`SCE_NP_COMMUNITY_ERROR_ABORTED`	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *`targetNpId`*, *`targetVirtualUserId`*, or *`variableArray`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist
`SCE_NP_COMMUNITY_ERROR_TOO_MANY_SLOTID`	0x80550718	Value exceeding 64 was specified for *`arrayNum`*
`SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN`	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA`	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
`SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE`	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE`	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE`	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE`	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`	0x8055082b	A storage for the specified user has not been created yet
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER`	0x80550849	Specified virtual user does not exist

Description

This function writes 64-bit integer values to the TUS variables corresponding to the specified slots of the specified user.

When writing values to TUS variables of an actual user, use `sceNpTusSetMultiSlotVariable()` and specify the user's NP ID for *`targetNpId`*. When writing values to TUS variables of a virtual user, use `sceNpTusSetMultiSlotVariableVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the target slot IDs. This area must be 4-byte-aligned and continuous. Specify the number of valid elements for this array in *`arrayNum`*.

For *`variableArray`*, specify the beginning address of the array storing the values to be written. This area must be 8-byte-aligned and continuous.

The value of *`variableArray`*`[0]` will be written to the TUS variable of the slot specified in *`slotIdArray`*`[0]`. The value of *`variableArray`*`[1]` will be written to the TUS variable of the slot specified in *`slotIdArray`*`[1]`. This process is continued so that values are written all at once to the number of TUS variables as specified in *`arrayNum`*. If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and write processing will not take place. If a timeout error occurs, however, the processing result will either be that values are written to all the specified TUS variables or to none of them.

If a value to be written to a TUS variable is outside the valid range set in the target TUS variable, it will be rounded up to the valid minimum value, or rounded down to the valid maximum value, and then stored.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

sceNpTusSetMultiSlotVariableAsync, sceNpTusSetMultiSlotVariableVUserAsync

Write values to multiple TUS variables of 1 user (asynchronous)

Definition

```
#include <np.h>

int sceNpTusSetMultiSlotVariableAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        const SceInt64 variableArray[],
        SceInt32 arrayNum,
        void *option
);

int sceNpTusSetMultiSlotVariableVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        const SceInt64 variableArray[],
        SceInt32 arrayNum,
        void *option
);
```

Arguments

`reqId`	Request ID (IN)
`targetNpId`	NP ID of user who is the write target (IN)
`targetVirtualUserId`	ID of virtual user who is the write target (IN)
`slotIdArray`	Pointer to the array storing the write target slot IDs (IN)
`variableArray`	Pointer to the array storing the 64-bit integers to write (IN)
`arrayNum`	Valid number of elements for `slotIdArray`. Maximum is 64 (IN)
`option`	Option reserved for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	0 was specified for `arrayNum`, or value other than NULL was specified for `option`
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for `reqId` does not exist

Description

This function writes 64-bit integer values to the TUS variables corresponding to the specified slots of the specified user.

When writing values to TUS variables of an actual user, use `sceNpTusSetMultiSlotVariableAsync()` and specify the user's NP ID for *`targetNpId`*. When writing values to TUS variables of a virtual user, use `sceNpTusSetMultiSlotVariableVUserAsync()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the target slot IDs. This area must be 4-byte-aligned and continuous. Specify the number of valid elements for this array in *`arrayNum`*.

For *`variableArray`*, specify the beginning address of the array storing the values to be written. This area must be 8-byte-aligned and continuous.

The value of *`variableArray`*`[0]` will be written to the TUS variable of the slot specified in *`slotIdArray`*`[0]`. The value of *`variableArray`*`[1]` will be written to the TUS variable of the slot specified in *`slotIdArray`*`[1]`. This process is continued so that values are written all at once to the number of TUS variables as specified in *`arrayNum`*. If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and write processing will not take place. If a timeout error occurs, however, the processing result will either be that values are written to all the specified TUS variables or to none of them.

If a value to be written to a TUS variable is outside the valid range set in the target TUS variable, it will be rounded up to the valid minimum value, or rounded down to the valid maximum value, and then stored.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained after confirming that processing has completed using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

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

sceNpTusGetMultiSlotVariable, sceNpTusGetMultiSlotVariableVUser

Read out multiple TUS variables of 1 user (synchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiSlotVariable (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiSlotVariableVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the read source (IN)
targetVirtualUserId	ID of virtual user who is the read source (IN)
slotIdArray	Pointer to the array storing the read source slot IDs (IN)
variableArray	Pointer to the array for storing the read TUS variable information (OUT)
variableArraySize	Memory size of *variableArray* (IN)
arrayNum	Valid number of elements for *slotIdArray*. Maximum is 64. (IN)
option	Option reserved for future extension. Always specify NULL

Return Values

Returns the number of the read TUS variables (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	A negative value was specified in the array specified for *slotIdArray*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId* or *targetVirtualUserId*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT	0x80550714	Structure size specified in *variableArraySize* is invalid
SCE_NP_COMMUNITY_ERROR_TOO_MANY_SLOTID	0x80550718	Value exceeding 64 was specified for *arrayNum*
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER	0x80550849	Specified virtual user does not exist

Description

This function reads the TUS variables corresponding to the specified slots of the specified user.

When reading values from TUS variables of an actual user, use `sceNpTusGetMultiSlotVariable()` and specify the user's NP ID for *`targetNpId`*. When reading values from TUS variables of a virtual user, use `sceNpTusGetMultiSlotVariableVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the read source slot IDs. This area must be 4-byte-aligned and continuous. For *`variableArray`*, specify the beginning address of the array for storing the read TUS values. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *`variableArraySize`*. Specify the number of valid elements for these arrays in *`arrayNum`*, and specify *`arrayNum`*`*sizeof(SceNpTusVariable)` for *`variableArraySize`*.

The TUS variable of the slot specified in *`slotIdArray`*`[0]` will be read and stored in *`variableArray`*`[0]`. The TUS variable of the slot specified in *`slotIdArray`*`[1]` will be read and stored in *`variableArray`*`[1]`. This process is continued so that values are read all at once from the number of TUS variables as specified in *`arrayNum`* and stored in *`variableArray`*. If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and read processing will not take place. If a timeout error occurs, however, the processing result will either be that values are read from all the TUS variables or from none of them.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

sceNpTusGetMultiSlotVariableAsync, sceNpTusGetMultiSlotVariableVUserAsync

Read out multiple TUS variables of 1 user (asynchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiSlotVariableAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiSlotVariableVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the read source (IN)
targetVirtualUserId	ID of virtual user who is the read source (IN)
slotIdArray	Pointer to the array storing the read source slot IDs (IN)
variableArray	Pointer to the array for storing the read TUS variable information (OUT)
variableArraySize	Memory size of `variableArray` (IN)
arrayNum	Valid number of elements for `slotIdArray`. Maximum is 64 (IN)
option	Option reserved for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	0 was specified for *arrayNum*, or value other than NULL was specified for *option*
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *targetNpId* or *targetVirtualUser*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *reqId* does not exist

Description

This function reads the TUS variables corresponding to the specified slots of the specified user.

When reading values from TUS variables of an actual user, use `sceNpTusGetMultiSlotVariableAsync()` and specify the user's NP ID for *targetNpId*. When reading values from TUS variables of a virtual user, use `sceNpTusGetMultiSlotVariableVUserAsync()` and specify the virtual user's ID for *targetVirtualUserId*.

For *slotIdArray*, specify the beginning address of the array storing the read source slot IDs. This area must be 4-byte-aligned and continuous. For *variableArray*, specify the beginning address of the array for storing the read TUS values. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *variableArraySize*. Specify the number of valid elements for these arrays in *arrayNum*, and specify *arrayNum*`*sizeof(SceNpTusVariable)` for *variableArraySize*.

The TUS variable of the slot specified in *slotIdArray*`[0]` will be read and stored in *variableArray*`[0]`. The TUS variable of the slot specified in *slotIdArray*`[1]` will be read and stored in *variableArray*`[1]`. This process is continued so that values are read all at once from the number of TUS variables as specified in *arrayNum* and stored in *variableArray*. If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and read processing will not take place. If a timeout error occurs, however, the processing result will either be that values are read from all the TUS variables or from none of them.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained after confirming that processing has completed using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

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

sceNpTusGetMultiUserVariable, sceNpTusGetMultiUserVariableVUser

Read 1 TUS variable of multiple users (synchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiUserVariable (
        SceInt32 reqId,
        const SceNpId targetNpIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiUserVariableVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId targetVirtualUserIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpIdArray	Pointer to the array storing the NP IDs of users who are the read source (IN)
targetVirtualUserIdArray	Pointer to the array storing the NP IDs of virtual users who are the read source (IN)
slotId	Read source slot ID (IN)
variableArray	Pointer to the array for storing the read TUS variable information (OUT)
variableArraySize	Memory size of `variableArray` (IN)
arrayNum	Valid number of elements for `targetNpIdArray` or `targetVirtualUserIdArray`. Maximum is 101 (IN)
option	Option reserved for future extension. Always specify NULL

Return Values

Returns the number of the read TUS variables (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	a negative value was specified for *`slotId`*, 0 was specified for *`arrayNum`*, or value other than NULL was specified for *`option`*
`SCE_NP_COMMUNITY_ERROR_ABORTED`	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *`targetNpIdArray`*, *`targetVirtualUserIdArray`*, or *`variableArray`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist
`SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT`	0x80550714	Structure size specified in *`variableArraySize`* is invalid
`SCE_NP_COMMUNITY_ERROR_TOO_MANY_NPID`	0x80550719	A value exceeding `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM` has been specified for *`arrayNum`*
`SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN`	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA`	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
`SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE`	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE`	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE`	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE`	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER`	0x80550849	Specified virtual user does not exist

Description

This function reads 1 TUS variable corresponding to the specified slot of the specified users.

When reading the TUS variable of actual users, use `sceNpTusGetMultiUserVariable()` and specify the array storing the NP IDs of those users for *`targetNpIdArray`*. When reading the TUS variable of virtual users, use `sceNpTusGetMultiUserVariableVUser()` and specify the virtual user ID array for *`targetVirtualUserIdArray`*.

For *`arrayNum`*, specify the number of valid elements for these arrays.

For *`slotId`*, specify the read source slot ID. For *`variableArray`*, specify the beginning address of the array for storing the read TUS variable values. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *`variableArraySize`*. Specify *`arrayNum`*`*sizeof(SceNpTusVariable)` to *`variableArraySize`*.

The TUS variable of the user specified in *`targetNpIdArray`*`[0]` or *`targetVirtualUserIdArray`*`[0]` will be read and stored in *`variableArray`*`[0]`. The TUS variable of the user specified in *`targetNpIdArray`*`[1]` or *`targetVirtualUserIdArray`*`[1]` will be read and stored in *`variableArray`*`[1]`. This process is continued so that the TUS variables for the number of users as specified in *`arrayNum`* are read all at once, and stored in *`variableArray`*.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

Notes

There are no counterparts of these functions, by which a write is performed to the TUS variable of multiple users all at once. This is because, given the design of the title user storage server, processing performed over multiple users cannot be efficiently handled.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusGetMultiUserVariableAsync()`, `sceNpTusGetMultiUserVariableVUserAsync()`, `SCE_NP_TUS_MAX_USER_NUM_PER_REQUEST`

sceNpTusGetMultiUserVariableAsync, sceNpTusGetMultiUserVariableVUserAsync

Read 1 TUS variable of multiple users (asynchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiUserVariableAsync (
        SceInt32 reqId,
        const SceNpId targetNpIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiUserVariableVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId targetVirtualUserIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpIdArray	Pointer to the array storing the NP IDs of users who are the read source (IN)
targetVirtualUserIdArray	Pointer to the array storing the NP IDs of virtual users who are the read source (IN)
slotId	Read source slot ID (IN)
variableArray	Pointer to array for storing the read TUS variable information (OUT)
variableArraySize	Memory size of `variableArray` (IN)
arrayNum	Valid number of elements for `targetNpIdArray` or `targetVirtualUserIdArray`. Maximum is 101 (IN)
option	Option reserved for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	a negative value was specified for *`slotId`*, 0 was specified for *`arrayNum`*, or value other than NULL was specified for *`option`*
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *`targetNpIdArray`*, *`targetVirtualUserIdArray`*, or *`variableArray`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist

Description

This function reads 1 TUS variable corresponding to the specified slot of the specified users.

When reading the TUS variable of actual users, use `sceNpTusGetMultiUserVariableAsync()` and specify the array storing the NP IDs of those users for *`targetNpIdArray`*. When reading the TUS variable of virtual users, use `sceNpTusGetMultiUserVariableVUserAsync()` and specify the virtual user ID array for *`targetVirtualUserIdArray`*.

For *`arrayNum`*, specify the number of valid elements for these arrays.

For *`slotId`*, specify the read source slot ID. For *`variableArray`*, specify the beginning address of the array for storing the read TUS variable values. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *`variableArraySize`*. Specify *`arrayNum`*`*sizeof(SceNpTusVariable)` to *`variableArraySize`*.

The TUS variable of the user specified in *`targetNpIdArray`*`[0]` or *`targetVirtualUserIdArray`*`[0]` will be read and stored in *`variableArray`*`[0]`. The TUS variable of the user specified in *`targetNpIdArray`*`[1]` or *`targetVirtualUserIdArray`*`[1]` will be read and stored in *`variableArray`*`[1]`. This process is continued so that the TUS variables for the number of users as specified in *`arrayNum`* are read all at once, and stored in *`variableArray`*.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

Notes

There are no counterparts of these functions, by which a write is performed to the TUS variable of multiple users all at once. This is because, given the design of the title user storage server, processing performed over multiple users cannot be efficiently handled.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusGetMultiUserVariable()`, `sceNpTusGetMultiUserVariableVUser()`, `SCE_NP_TUS_MAX_USER_NUM_PER_REQUEST`

sceNpTusGetFriendsVariable

Specify sort conditions and read TUS variables for the top user results (up to 100) from among friends (synchronous)

Definition

```
#include <np.h>
int sceNpTusGetFriendsVariable (
        SceInt32 reqId,
        SceNpTusSlotId slotId,
        SceBool includeSelf,
        SceInt32 sortType,
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        SceNpTusGetFriendsVariableOptParam *option
);
```

Arguments

reqId	Request ID (IN)
slotId	Read source slot ID (IN)
includeSelf	Flag to indicate whether the current user's TUS variable is to be included in the target variables to obtain or not. Specify any number besides 0 to include and 0 to exclude (IN)
sortType	Friend sort conditions (IN)
variableArray	Pointer to array for storing the read TUS variable information (OUT)
variableArraySize	Memory size of *variableArray* (IN)
arrayNum	Number of TUS variables to obtain. Up to `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM` (100) (IN)
option	Pointer to extended option structure. Specify NULL if not required

Return Values

Returns the number of the read TUS variables (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	A negative value was specified for *`slotId`*, or 0 was specified for *`arrayNum`*, or value other than NULL was specified for *`option`*
`SCE_NP_COMMUNITY_ERROR_ABORTED`	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *`variableArray`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist
`SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT`	0x80550714	Structure size specified in *`variableArraySize`* is invalid
`SCE_NP_COMMUNITY_ERROR_TOO_MANY_NPID`	0x80550719	Value exceeding `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM` was specified for *`arrayNum`*
`SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN`	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA`	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
`SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE`	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE`	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE`	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE`	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server

Description

This function specifies a single slot and sort conditions, then reads the TUS variables for the top friend results.

For *includeSelf*, specify any number besides 0 to include the current user in the target TUS variables to obtain, and specify 0 to exclude the current user in the target variables.

For *arrayNum*, specify the maximum number of users whose TUS variables will be obtained in the top results.

For *slotId*, specify the read source slot ID.

For *variableArray*, specify the beginning address of the array for storing the read TUS variables. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *variableArraySize*.

Specify *arrayNum*`*sizeof(SceNpTusVariable)` to *variableArraySize*.

For *sortType*, specify the friend sort conditions with one of the following values.

Value	Description
`SCE_NP_TUS_VARIABLE_SORTTYPE_DESCENDING_VALUE`	Sorts the TUS variable numerical values in descending order
`SCE_NP_TUS_VARIABLE_SORTTYPE_ASCENDING_VALUE`	Sorts the TUS variable numerical values in ascending order
`SCE_NP_TUS_VARIABLE_SORTTYPE_DESCENDING_DATE`	Sorts by last update date/time in descending order
`SCE_NP_TUS_VARIABLE_SORTTYPE_ASCENDING_DATE`	Sorts by last update date/time in ascending order

In, *variableArray*, the friend TUS variables will be stored from the top sort result up to the number specified with *arrayNum*.

The number of stored TUS variables will return as the function return value.

If not even one friend TUS variable was registered, the return value will be 0 and normal termination will occur.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusGetFriendsVariableAsync()`, `sceNpTusGetFriendsDataStatus()`, `sceNpTusGetFriendsDataStatusAsync()`, `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM`, `SceNpTusGetFriendsVariableOptParam`

sceNpTusGetFriendsVariableAsync

Specify sort conditions and read TUS variables for the top user results (up to 100) from among friends (asynchronous)

Definition

```
#include <np.h>
int sceNpTusGetFriendsVariableAsync (
        SceInt32 reqId,
        SceNpTusSlotId slotId,
        SceBool includeSelf,
        SceInt32 sortType,
        SceNpTusVariable variableArray[],
        SceSize variableArraySize,
        SceInt32 arrayNum,
        SceNpTusGetFriendsVariableOptParam *option
);
```

Arguments

reqId	Request ID (IN)
slotId	Read source slot ID (IN)
includeSelf	Flag to indicate whether the current user's TUS variable is to be included in the target variables to obtain or not. Specify any number besides 0 to include and 0 to exclude (IN)
sortType	Friend sort conditions (IN)
variableArray	Pointer to array for storing the read TUS variable information (OUT)
variableArraySize	Memory size of *variableArray* (IN)
arrayNum	Number of TUS variables to obtain. Up to SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM (100) (IN)
option	Pointer to extended option structure. Specify NULL if not required

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_ INVALID_ARGUMENT	0x80550704	A negative value was specified for *slotId*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *variableArray*
SCE_NP_COMMUNITY_ERROR_ INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_ INVALID_ALIGNMENT	0x80550714	Structure size specified in *variableArraySize* is invalid
SCE_NP_COMMUNITY_ERROR_ TOO_MANY_NPID	0x80550719	Value exceeding SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM was specified for *arrayNum*

Description

This function specifies a single slot and sort conditions, then reads the TUS variables for the top friend results.

For *`includeSelf`*, specify any number besides 0 to include the current user in the target TUS variables to obtain, and specify 0 to exclude the current user in the target variables.

For *`arrayNum`*, specify the maximum number of users whose TUS variables will be obtained in the top results.

For *`slotId`*, specify the read source slot ID.

For *`variableArray`*, specify the beginning address of the array for storing the read TUS variables. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *`variableArraySize`*. Specify *`arrayNum`*`*sizeof(SceNpTusVariable)` to *`variableArraySize`*.

For *`sortType`*, specify the friend sort conditions with one of the following values.

Value	Description
`SCE_NP_TUS_VARIABLE_SORTTYPE_DESCENDING_VALUE`	Sorts the TUS variable numerical values in descending order
`SCE_NP_TUS_VARIABLE_SORTTYPE_ASCENDING_VALUE`	Sorts the TUS variable numerical values in ascending order
`SCE_NP_TUS_VARIABLE_SORTTYPE_DESCENDING_DATE`	Sorts by last update date/time in descending order
`SCE_NP_TUS_VARIABLE_SORTTYPE_ASCENDING_DATE`	Sorts by last update date/time in ascending order

In, *`variableArray`*, the friend TUS variables will be stored from the top sort result up to the number specified with *`arrayNum`*.

The number of TUS variables stored in *`variableArray`* will be stored in *`result`* in `sceNpTusWaitAsync()`/`sceNpTusPollAsync()`. If not even one friend TUS variable was registered, 0 will be stored in *`result`* and normal termination will occur.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusGetFriendsVariable()`, `sceNpTusGetFriendsDataStatus()`, `sceNpTusGetFriendsDataStatusAsync()`, `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM`, `SceNpTusGetFriendsVariableOptParam`

sceNpTusAddAndGetVariable, sceNpTusAddAndGetVariableVUser

Add a 64-bit integer to a TUS variable and get the result of the addition (synchronous)

Definition

```
#include <np.h>

int sceNpTusAddAndGetVariable (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceInt64 inVariable,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *outVariable,
        SceSize outVariableSize,
        void *option
);

int sceNpTusAddAndGetVariableVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceInt64 inVariable,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *outVariable,
        SceSize outVariableSize,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotId	Target slot ID (IN)
inVariable	64-bit integer to add (IN)
isLastChangedAuthor	Specification of the author of the update for conflict prevention. Addition is only executed when the author of the last update of the TUS variable currently registered on the server is identical with the specified NP ID. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary (IN)
isLastChangedDate	Specification of the date and time for conflict prevention. Addition is only executed when the time of the last update of the TUS variable currently registered on the server is identical with or older than the specified time. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary (IN)
outVariable	Pointer to structure for storing the TUS variable information after the addition (OUT)
outVariableSize	Memory size of *outVariable* (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	Value other than NULL was specified for *`option`*, or a negative value was specified for *`slotId`*
`SCE_NP_COMMUNITY_ERROR_ABORTED`	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *`targetNpId`* or *`targetVirtualUserId`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist
`SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT`	0x80550714	Structure size specified for *`outVariableSize`* is invalid
`SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN`	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA`	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
`SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE`	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE`	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE`	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE`	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`	0x8055082b	A storage for the specified user has not been created yet
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER`	0x80550849	Specified virtual user does not exist
`SCE_NP_COMMUNITY_SERVER_ERROR_CONDITIONS_NOT_SATISFIED`	0x80550873	Processing could not be executed because a conflict was detected against the specification of *`isLastChangedAuthor`* and/or *`isLastChangedDate`*

Description

This function adds a 64-bit integer to a TUS variable corresponding to the specified slot of the specified user.

When adding a value to the TUS value of an actual user, use `sceNpTusAddAndGetVariable()` and specify the user's NP ID for *`targetNpId`*. When adding a value to the TUS value of a virtual user, use `sceNpTusAddAndGetVariableVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotId`*, specify the slot ID indicating the target TUS variable. Specify the 64-bit value to add to that TUS variable in *`invariable`*. It is also possible to subtract from the TUS variable by specifying a negative value for *`invariable`*.

The result of the addition will be stored in the area specified by *`outVariable`* (the *`variable`* member will represent the value after the addition). Specify `sizeof(SceNpTusVariable)` for *`outVariableSize`*.

If the value after the addition is outside the valid value range set beforehand to the TUS variable, it will be rounded down to the valid maximum value or rounded up to the valid minimum value, and then stored.

If a value has not been set to the TUS variable yet, this function will handle processing as though 0 had been set. However, an error will occur if the specified user has never accessed the title user storage server, and the function will return `SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`.

Also, when the author and time of the last update are specified in *`isLastChangedAuthor`* and *`isLastChangedDate`*, respectively, these will be checked against the values that have been set on the server, and, if they are different, writing will fail and an error will return. An error will also return if no values have been set on the server (if they have been deleted). This feature was devised to be used in cases where you wish to have only the operations of the first user reflected when multiple users have performed operations in the same slot. If exclusive processing is not necessary, specify NULL.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusAddAndGetVariableAsync()`, `sceNpTusAddAndGetVariableVUserAsync()`

sceNpTusAddAndGetVariableAsync, sceNpTusAddAndGetVariableVUserAsync

Add a 64-bit integer to a TUS variable and get the result of the addition (asynchronous)

Definition

```
#include <np.h>

int sceNpTusAddAndGetVariableAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceInt64 inVariable,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *outVariable,
        SceSize outVariableSize,
        void *option
);

int sceNpTusAddAndGetVariableVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceInt64 inVariable,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *outVariable,
        SceSize outVariableSize,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotId	Target slot ID (IN)
inVariable	64-bit integer to add (IN)
isLastChangedAuthor	Specification of the author of the update for conflict prevention. Addition is only executed when the author of the last update of the TUS variable currently registered on the server is identical with the specified NP ID. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary
isLastChangedDate	Specification of the date and time for conflict prevention. Addition is only executed when the time of the last update of the TUS variable currently registered on the server is identical with or older than the specified time. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary (IN)
outVariable	Pointer to structure for storing the TUS variable information after the addition (OUT)
outVariableSize	Memory size of `outVariable` (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	Value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED	0x8055082b	A storage for the specified user has not been created yet

Description

This function adds a 64-bit integer to a TUS variable corresponding to the specified slot of the specified user.

When adding a value to the TUS value of an actual user, use `sceNpTusAddAndGetVariableAsync()`, and specify the user's NP ID for *targetNpId*. When adding a value to the TUS value of a virtual user, use `sceNpTusAddAndGetVariableVUserAsync()` and specify the virtual user's ID for *targetVirtualUserId*.

For *slotId*, specify the slot ID indicating the target TUS variable. Specify the 64-bit value to add to that TUS variable in *invariable*. It is also possible to subtract from the TUS variable by specifying a negative value for *invariable*.

The result of the addition will be stored in the area specified by *outVariable* (the *variable* member will represent the value after the addition). Specify `sizeof(SceNpTusVariable)` for *outVariableSize*.

If the value after the addition is outside the valid value range set beforehand to the TUS variable, it will be rounded down to the valid maximum value or rounded up to the valid minimum value, and then stored.

If a value has not been set to the TUS variable yet, this function will handle processing as though 0 had been set. However, an error will occur if the specified user has never accessed the title user storage server, and the function will return `SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`.

Also, when the author and time of the last update are specified in *isLastChangedAuthor* and *isLastChangedDate*, respectively, these will be checked against the values that have been set on the server, and, if they are different, writing will fail and an error will return. An error will also return if no values have been set on the server (if they have been deleted). This feature was devised to be used in cases where you wish to have only the operations of the first user reflected when multiple users have performed operations in the same slot. If exclusive processing is not necessary, specify NULL.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusAddAndGetVariable()`, `sceNpTusAddAndGetVariableVUser()`

sceNpTusTryAndSetVariable, sceNpTusTryAndSetVariableVUser

Conditionally write a 64-bit integer to a TUS variable (synchronous)

Definition

```
#include <np.h>

int sceNpTusTryAndSetVariable (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceInt32 opeType,
        SceInt64 variable,
        const SceInt64 *compareValue,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *resultVariable,
        SceSize resultVariableSize,
        void *option
);

int sceNpTusTryAndSetVariableVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceInt32 opeType,
        SceInt64 variable,
        const SceInt64 *compareValue,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *resultVariable,
        SceSize resultVariableSize,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotId	Target slot ID (IN)
opeType	Value representing the write condition (IN)
variable	64-bit integer to write (IN)
compareValue	64-bit integer to be compared with the value registered on the server. If NULL is specified, the value of `variable` will be used for comparison
isLastChangedAuthor	Specification of the author of the update for conflict prevention. Processing is only executed when the author of the last update of the TUS variable currently registered on the server is identical with the specified NP ID. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary

isLastChangedDate	Specification of the date and time for conflict prevention. Processing is only executed when the time of the last update of the TUS variable currently registered on the server is identical with or older than the specified time. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary (IN)
resultVariable	Pointer to structure for storing the TUS variable information after the processing (OUT)
resultVariableSize	Size of the `SceNpTusVariable` structure (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. The main error codes are shown below. (The application must not malfunction even if other error codes are returned.) This function does not handle it as an error when the specified condition is not met and the write does not take place.

Value	(Number)	Description
SCE_NP_COMMUNITY_ERROR_NOT_INITIALIZED	0x80550702	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	A negative value was specified for *slotId*, a nonexistent value was specified for *opeType*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *resultVariable*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_INVALID_TYPE	0x80550711	Value not defined was specified for *opeType*
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED	0x8055082b	A storage for the specified user has not been created yet

Value	(Number)	Description
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER`	0x80550849	Specified virtual user does not exist
`SCE_NP_COMMUNITY_SERVER_ERROR_CONDITIONS_NOT_SATISFIED`	0x80550873	Processing could not be executed because a conflict was detected against the specification of *`isLastChangedAuthor`* and/or *`isLastChangedDate`*

Description

This function writes a 64-bit integer to a TUS variable corresponding to the specified slot of the specified user. Before writing, the function compares the current value and the value to be written, and performs the write if the specified condition is met.

When writing a value to the TUS variable of an actual user, use `sceNpTusTryAndSetVariable()` and specify the user's NP ID for *`targetNpId`*. When writing a value to the TUS variable of a virtual user, use `sceNpTusTryAndSetVariableVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotId`*, specify the slot ID indicating the target TUS variable.

For *`opeType`*, specify one of the following values as the condition for performing a write.

Value	Description
`SCE_NP_TUS_OPETYPE_EQUAL`	Writes if *`variable`* is equal to the current value
`SCE_NP_TUS_OPETYPE_NOT_EQUAL`	Writes if *`variable`* is not equal to the current value
`SCE_NP_TUS_OPETYPE_GREATER_THAN`	Writes if *`variable`* is greater than the current value
`SCE_NP_TUS_OPETYPE_GREATER_OR_EQUAL`	Writes if *`variable`* is greater than or equal to the current value
`SCE_NP_TUS_OPETYPE_LESS_THAN`	Writes if *`variable`* is less than the current value
`SCE_NP_TUS_OPETYPE_LESS_OR_EQUAL`	Writes if *`variable`* is equal to or less than the current value

Specify the 64-bit integer for comparison with the existing value in *`compareValue`*. If NULL is specified in *`compareValue`*, the value specified in *`variable`* will be compared with the existing value instead. For *`variable`*, specify a 64-bit integer to be used as a value to write to the TUS variable if the specified condition is met.

The TUS variable after the processing will be stored in the area specified with *`resultVariable`*, regardless of whether the condition is met or not. (The *`variable`* member represents the value after the processing.) Specify `sizeof(SceNpTusVariable)` for *`resultVariableSize`*.

When the value to be written is outside the valid value range set beforehand to the TUS variable, it will be used as is for the comparison, but rounded down to the valid maximum value or rounded up to the valid minimum value for the write.

If a value has not been set to the TUS variable yet, this function will handle processing as though the specified condition has been met. However, an error will occur if the specified user has never accessed the title user storage server, and the function will return `SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`.

Also, when the author and time of the last update are specified in *`isLastChangedAuthor`* and *`isLastChangedDate`*, respectively, these will be checked against the values that have been set on the server, and, if they are different, writing will fail and an error will return. An error will also return if no values have been set on the server (if they have been deleted). This feature was devised to be used in cases where you wish to have only the operations of the first user reflected when multiple users have performed operations in the same slot. If exclusive processing is not necessary, specify NULL.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusTryAndSetVariableAsync()`, `sceNpTusTryAndSetVariableVUserAsync()`

sceNpTusTryAndSetVariableAsync, sceNpTusTryAndSetVariableVUserAsync

Conditionally write a 64-bit integer to a TUS variable (asynchronous)

Definition

```
#include <np.h>

int sceNpTusTryAndSetVariableAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceInt32 opeType,
        SceInt64 variable,
        const SceInt64 *compareValue,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *resultVariable,
        SceSize resultVariableSize,
        void *option
);

int sceNpTusTryAndSetVariableVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceInt32 opeType,
        SceInt64 variable,
        const SceInt64 *compareValue,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        SceNpTusVariable *resultVariable,
        SceSize resultVariableSize,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotId	Target slot ID (IN)
opeType	Value representing the write condition (IN)
variable	64-bit integer to write (IN)
compareValue	64-bit integer to be compared with the value registered on the server. If NULL is specified, the value of `variable` will be used for comparison
isLastChangedAuthor	Specification of the author of the update for conflict prevention. Processing is only executed when the author of the last update of the TUS variable currently registered on the server is identical with the specified NP ID. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary

isLastChangedDate	Specification of the date and time for conflict prevention. Processing is only executed when the time of the last update of the TUS variable currently registered on the server is identical with or older than the specified time. Processing is not performed when no TUS variables are registered on the server. Specify NULL if comparison is not necessary (IN)
resultVariable	Pointer to structure for storing the TUS variable information after the processing (OUT)
resultVariableSize	Size of the `SceNpTusVariable` structure (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	Value other than NULL was specified for *option*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *reqId* does not exist
`SCE_NP_COMMUNITY_ERROR_INVALID_TYPE`	0x80550711	Value not defined was specified for *opeType*
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`	0x8055082b	A storage for the specified user has not been created yet

Description

This function writes a 64-bit integer to a TUS variable corresponding to the specified slot of the specified user. Before writing, the function compares the current value and the value to be written, and performs the write if the specified condition is met.

When writing a value to the TUS variable of an actual user, use `sceNpTusTryAndSetVariableAsync()` and specify the user's NP ID for *targetNpId*. When writing a value to the TUS variable of a virtual user, use `sceNpTusTryAndSetVariableVUserAsync()` and specify the virtual user's ID for *targetVirtualUserId*.

For *slotId*, specify the slot ID indicating the target TUS variable.

For *opeType*, specify one of the following values as the condition for performing a write.

Value	Description
`SCE_NP_TUS_OPETYPE_EQUAL`	Writes if *variable* is equal to the current value
`SCE_NP_TUS_OPETYPE_NOT_EQUAL`	Writes if *variable* is not equal to the current value
`SCE_NP_TUS_OPETYPE_GREATER_THAN`	Writes if *variable* is greater than the current value
`SCE_NP_TUS_OPETYPE_GREATER_OR_EQUAL`	Writes if *variable* is greater than or equal to the current value
`SCE_NP_TUS_OPETYPE_LESS_THAN`	Writes if *variable* is less than the current value
`SCE_NP_TUS_OPETYPE_LESS_OR_EQUAL`	Writes if *variable* is equal to or less than the current value

Specify the 64-bit integer for comparison with the existing value in *`compareValue`*. If NULL is specified in *`compareValue`*, the value specified in *`variable`* will be compared with the existing value instead. For *`variable`*, specify a 64-bit integer to be used as a value to write to the TUS variable if the specified condition is met.

The TUS variable after the processing will be stored in the area specified with *`resultVariable`*, regardless of whether the condition is met or not. (The *`variable`* member represents the value after the processing.) Specify `sizeof(SceNpTusVariable)` for *`resultVariableSize`*.

When the value to be written is outside the valid value range set beforehand to the TUS variable, it will be used as is for the comparison, but rounded down to the valid maximum value or rounded up to the valid minimum value for the write.

If a value has not been set to the TUS variable yet, this function will handle processing as though the specified condition has been met. However, an error will occur if the specified user has never accessed the title user storage server, and the function will return `SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED`.

Also, when the author and time of the last update are specified in *`isLastChangedAuthor`* and *`isLastChangedDate`*, respectively, these will be checked against the values that have been set on the server, and, if they are different, writing will fail and an error will return. An error will also return if no values have been set on the server (if they have been deleted). This feature was devised to be used in cases where you wish to have only the operations of the first user reflected when multiple users have performed operations in the same slot. If exclusive processing is not necessary, specify NULL.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusTryAndSetVariable()`, `sceNpTusTryAndSetVariableVUser()`

sceNpTusDeleteMultiSlotVariable, sceNpTusDeleteMultiSlotVariableVUser

Delete multiple TUS variables of 1 user (synchronous)

Definition

```
#include <np.h>

int sceNpTusDeleteMultiSlotVariable (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);

int sceNpTusDeleteMultiSlotVariableVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotIdArray	Pointer to the array storing the target slot IDs (IN)
arrayNum	Valid number of elements for *slotIdArray*. Maximum is 64 (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	A negative value was specified for one of the elements of *slotIdArray*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *slotIdArray*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_TOO_MANY_SLOTID	0x80550718	Value exceeding 64 was specified for *arrayNum*

Value	(Number)	Description
`SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN`	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA`	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
`SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE`	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE`	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE`	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE`	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER`	0x80550849	Specified virtual user does not exist

Description

This function deletes multiple TUS variables corresponding to the specified slots of the specified user. The deleted TUS variables will return to a state where values are not set to them.

When deleting the TUS variables of an actual user, use `sceNpTusDeleteMultiSlotVariable()` and specify the user's NP ID for *`targetNpId`*. When deleting the TUS variables of a virtual user, use `sceNpTusDeleteMultiSlotVariableVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the slot IDs of the TUS variables to delete. This area must be 4-byte-aligned and continuous. Specify the number of specified slot IDs for *`arrayNum`*.

If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and delete processing will not take place. If a timeout error occurs, however, the processing result will either be that values are deleted for all the TUS variables or for none of them.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusDeleteMultiSlotVariableAsync()`, `sceNpTusDeleteMultiSlotVariableVUserAsync()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

sceNpTusDeleteMultiSlotVariableAsync, sceNpTusDeleteMultiSlotVariableVUserAsync

Delete multiple TUS variables of 1 user (asynchronous)

Definition

```
#include <np.h>

int sceNpTusDeleteMultiSlotVariableAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);

int sceNpTusDeleteMultiSlotVariableVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);
```

Arguments

`reqId`	Request ID (IN)
`targetNpId`	NP ID of the target user (IN)
`targetVirtualUserId`	ID of the target virtual user (IN)
`slotIdArray`	Pointer to the array storing the target slot IDs (IN)
`arrayNum`	Valid number of elements for `slotIdArray`. Maximum is 64 (IN)
`option`	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	0 was specified for `arrayNum`, or value other than NULL was specified for `option`
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for `targetNpId`, `targetVirtualUserId`, or `slotIdArray`
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for `reqId` does not exist

Description

This function deletes multiple TUS variables corresponding to the specified slots of the specified user. The deleted TUS variables will return to a state where values are not set to them.

When deleting the TUS variables of an actual user, use `sceNpTusDeleteMultiSlotVariableAsync()` and specify the user's NP ID for *`targetNpId`*. When deleting the TUS variables of a virtual user, use `sceNpTusDeleteMultiSlotVariableVUserAsync()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the slot IDs of the TUS variables to delete. This area must be 4-byte-aligned and continuous. Specify the number of specified slot IDs for *`arrayNum`*.

If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and delete processing will not take place. If a timeout error occurs, however, the processing result will either be that values are deleted for all the TUS variables or for none of them.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusDeleteMultiSlotVariable()`, `sceNpTusDeleteMultiSlotVariableVUser()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

TUS Data Operation Functions

sceNpTusSetData, sceNpTusSetDataVUser

Upload TUS data (synchronous)

Definition

```
#include <np.h>

int sceNpTusSetData (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceSize totalSize,
        SceSize sendSize,
        const void *data,
        const SceNpTusDataInfo *info,
        SceSize infoStructSize,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        void *option
);

int sceNpTusSetDataVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceSize totalSize,
        SceSize sendSize,
        const void *data,
        const SceNpTusDataInfo *info,
        SceSize infoStructSize,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the update target (IN)
targetVirtualUserId	ID of virtual user who is the update target (IN)
slotId	Update target slot ID (IN)
totalSize	Total size of data to upload (IN)
sendSize	Size of data to send this time (IN)
data	Pointer to data to send this time (IN)
info	Pointer to structure storing accessory information (IN)
infoStructSize	Size of the accessory information (IN)
isLastChangedAuthor	Specification of the author of the update for conflict prevention. Processing is only executed when the author of the last update of the TUS data currently registered on the server is identical with the specified NP ID. Processing is not performed when no TUS data is registered on the server. Specify NULL if comparison is not necessary (IN)
isLastChangedDate	Specification of the date and time for conflict prevention. Processing is only executed when the time of the last update of the TUS data currently registered on the server is identical with or older than the specified time. Processing is not performed when no TUS data is registered on the server. Specify NULL if comparison is not necessary (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	Value other than NULL was specified for *option*, a negative value was specified for *slotId*, or the size specified to the *infoSize* member of the *info* structure is invalid
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *data*; or 0 was specified for *totalSize*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT	0x80550714	The size specified for *infoStructSize* is invalid
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in SceNpCommunicationPassphrase
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TOO_LARGE_DATA	0x8055081a	The size specified for *totalSize* exceeds the maximum size set to the slot
SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED	0x8055082b	A storage for the specified user has not been created yet
SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER	0x80550849	Specified virtual user does not exist
SCE_NP_COMMUNITY_SERVER_ERROR_CONDITIONS_NOT_SATISFIED	0x80550873	Processing could not be executed because a conflict was detected against the specification of *isLastChangedAuthor* and/or *isLastChangedDate*

Description

This function uploads TUS data corresponding to the specified slot of the specified user. Data can also be uploaded in installments by calling this function multiple times.

When uploading TUS data of an actual user, use `sceNpTusSetData()` and specify the user's NP ID for *`targetNpId`*. When uploading TUS data of a virtual user, use `sceNpTusSetDataVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotId`*, specify the slot ID of the upload target.

Store the data to upload in the area specified for *`data`*, and specify the data size in *`sendSize`*. Specify the total size of the data to upload in *`totalSize`*.

Accessory information that serves as the TUS index data should be stored in the area specified for *`info`*. Specify `sizeof(SceNpTusDataInfo)` for *`infoStructSize`*. Note that a valid data size must be stored to the *`infoSize`* member of the *`info`* structure. If accessory information is not required, specify NULL for *`info`*.

When uploading data in installments, specify the same values for *`reqId`*, *`targetNpId`* or *`targetVirtualUserId`*, and *`slotId`* for every function call. Data will be sent for each call according to the values set each time in *`data`* and *`sendSize`*. However, the value specified for *`totalSize`* in the first call will be used, and this argument will be ignored in the subsequent calls. Moreover, because accessory information is sent in the first function call, the values specified for *`info`* and *`infoStructSize`* in the subsequent calls will be ignored.

Also, when the author and time of the last update are specified in *`isLastChangedAuthor`* and *`isLastChangedDate`*, respectively, these will be checked against the values that have been set on the server, and, if they are different, writing will fail and an error will return. An error will also return if no values have been set on the server (if they have been deleted). This feature was devised to be used in cases where you wish to have only the operations of the first user reflected when multiple users have performed operations in the same slot. If exclusive processing is not necessary, specify NULL.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After sending all the data and returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusSetDataAsync()`, `sceNpTusSetDataVUserAsync()`

sceNpTusSetDataAsync, sceNpTusSetDataVUserAsync

Upload TUS data (asynchronous)

Definition

```
#include <np.h>

int sceNpTusSetDataAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceSize totalSize,
        SceSize sendSize,
        const void *data,
        const SceNpTusDataInfo *info,
        SceSize infoStructSize,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        void *option
);

int sceNpTusSetDataVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceSize totalSize,
        SceSize sendSize,
        const void *data,
        const SceNpTusDataInfo *info,
        SceSize infoStructSize,
        const SceNpId *isLastChangedAuthor,
        const SceRtcTick *isLastChangedDate,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the update target (IN)
targetVirtualUserId	ID of virtual user who is the update target (IN)
slotId	Update target slot ID (IN)
totalSize	Total size of data to upload (IN)
sendSize	Size of data to send this time (IN)
data	Pointer to data to send this time (IN)
info	Pointer to structure storing accessory information (IN)
infoStructSize	Size of the accessory information (IN)
isLastChangedAuthor	Specification of the author of the update for conflict prevention. Processing is only executed when the author of the last update of the TUS data currently registered on the server is identical with the specified NP ID. Processing is not performed when no TUS data is registered on the server. Specify NULL if comparison is not necessary (IN)

`isLastChangedDate`	Specification of the date and time for conflict prevention. Processing is only executed when the time of the last update of the TUS data currently registered on the server is identical with or older than the specified time. Processing is not performed when no TUS data is registered on the server. Specify NULL if comparison is not necessary (IN)
`option`	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	Value other than NULL was specified for *`option`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist

Description

This function uploads TUS data corresponding to the specified slot of the specified user. Data can also be uploaded in installments by calling this function multiple times.

When uploading TUS data of an actual user, use `sceNpTusSetDataAsync()` and specify the user's NP ID for *`targetNpId`*. When uploading TUS data of a virtual user, use `sceNpTusSetDataVUserAsync()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotId`*, specify the slot ID of the upload target.

Store the data to upload in the area specified for *`data`*, and specify the data size in *`sendSize`*. Specify the total size of the data to upload in *`totalSize`*.

Accessory information that serves as the TUS index data should be stored in the area specified for *`info`*. Specify `sizeof(SceNpTusDataInfo)` for *`infoStructSize`*. Note that a valid data size must be stored to the *`infoSize`* member of the *`info`* structure. If accessory information is not required, specify NULL for *`info`*.

When uploading data in installments, specify the same values for *`reqId`*, *`targetNpId`* or *`targetVirtualUserId`*, and *`slotId`* for every function call. Data will be sent for each call according to the values set each time in *`data`* and *`sendSize`*. However, the value specified for *`totalSize`* in the first call will be used, and this argument will be ignored in the subsequent calls. Moreover, because accessory information is sent in the first function call, the values specified for *`info`* and *`infoStructSize`* in the subsequent calls will be ignored.

Also, when the author and time of the last update are specified in *`isLastChangedAuthor`* and *`isLastChangedDate`*, respectively, these will be checked against the values that have been set on the server, and, if they are different, writing will fail and an error will return. An error will also return if no values have been set on the server (if they have been deleted). This feature was devised to be used in cases where you wish to have only the operations of the first user reflected when multiple users have performed operations in the same slot. If exclusive processing is not necessary, specify NULL.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The actual processing will be performed on an internal thread created by `sceNpTusInit()`. Note that the area specified for *`data`* must be held until the processing completes. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example.

After sending all the data and obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

Notes

Until the asynchronous processing completes, another asynchronous processing cannot be issued from the same request ID. Even when using a different request ID, note that there is a possibility of memory running out on the network layer.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusSetData()`, `sceNpTusSetDataVUser()`

sceNpTusGetData, sceNpTusGetDataVUser

Download TUS data (synchronous)

Definition

```
#include <np.h>

int sceNpTusGetData (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceNpTusDataStatus *dataStatus,
        SceSize dataStatusSize,
        void *data,
        SceSize recvSize,
        void *option
);

int sceNpTusGetDataVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceNpTusDataStatus *dataStatus,
        SceSize dataStatusSize,
        void *data,
        SceSize recvSize,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the download source (IN)
targetVirtualUserId	ID of virtual user who is the download source (IN)
slotId	Slot ID of the download source (IN)
dataStatus	Pointer to structure for storing the status of the TUS data (OUT)
dataStatusSize	Size of the status of the TUS data (IN)
data	Pointer to area for storing the 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 the size of the received data (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	Value other than NULL was specified for *option*, or a negative value was specified for *slotId*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId* or *targetVirtualUserId*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT	0x80550714	The structure size specified for *dataStatusSize* is invalid
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_USER_NOT_ASSIGNED	0x8055082b	A storage for the specified user has not been created yet
SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER	0x80550849	Specified virtual user does not exist

Description

This function downloads TUS data corresponding to the specified slot of the specified user.

When downloading TUS data of an actual user, use `sceNpTusGetData()` and specify the user's NP ID for *`targetNpId`*. When downloading TUS data of a virtual user, use `sceNpTusGetDataVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotId`*, specify the slot ID of the download source.

Specify a pointer to the structure for storing the status of the TUS data in *`dataStatus`*. Specify `sizeof(SceNpTusDataStatus)` for *`dataStatusSize`*.

For *`data`*, specify the area for storing the data to receive, and specify its size in *`recvSize`*.

The total size of the TUS data will be stored in the *`dataSize`* member of the *`dataStatus`* structure. If this value is greater than the value specified for *`recvSize`*, this means only a part of the data has been received. Call this function again to receive the remaining data. For subsequent calls, the same values should be specified as the first for *`reqId`*, *`targetNpId`* or *`targetVirtualUserId`*, and *`slotId`*. Values specified for *`dataStatus`*, *`data`*, and *`recvSize`* can differ. Determine the end of the data from the application side by comparing the total size of the TUS data and the received data size.

When this function is called with NULL specified for *`data`*, data will not be received and only the data status will be stored in the *`dataStatus`* structure. Thus, this function can be called in this manner for the first call to obtain the total size of the TUS data to receive. A sufficient amount of memory can then be allocated and the entire data received from the second call of this function.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After receiving the entire data and returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

Notes

Functions for obtaining the data status are `sceNpTusGetMultiSlotDataStatus()`/`sceNpTusGetMultiSlotDataStatusVUser()` and `sceNpTusGetMultiUserDataStatus()`/`sceNpTusGetMultiUserDataStatusVUser()`. If you want to obtain the data status for reasons other than checking the total data size before downloading, use either of the above functions instead of this one.

The *`data`* member of the *`dataStatus`* structure exists for reasons of practicality for managing data. You must be careful when receiving a single TUS data in installments, especially, because the address specified first for the *`data`* argument of this function (other than NULL) will be returned for each call of the function and this will not indicate the address of the received data.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusGetDataAsync()`, `sceNpTusGetDataVUserAsync()`

sceNpTusGetDataAsync, sceNpTusGetDataVUserAsync

Download TUS data (asynchronous)

Definition

```
#include <np.h>

int sceNpTusGetDataAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        SceNpTusSlotId slotId,
        SceNpTusDataStatus *dataStatus,
        SceSize dataStatusSize,
        void *data,
        SceSize recvSize,
        void *option
);

int sceNpTusGetDataVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        SceNpTusSlotId slotId,
        SceNpTusDataStatus *dataStatus,
        SceSize dataStatusSize,
        void *data,
        SceSize recvSize,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the download source (IN)
targetVirtualUserId	ID of virtual user who is the download source (IN)
slotId	Slot ID of the download source (IN)
dataStatus	Pointer to structure for storing the status of the TUS data (OUT)
dataStatusSize	Size of the status of the TUS data (IN)
data	Pointer to area for storing the 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 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	Value other than NULL was specified for `option`
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for `reqId` does not exist

Description

This function downloads TUS data corresponding to the specified slot of the specified user.

When downloading TUS data of an actual user, use `sceNpTusGetDataAsync()` and specify the user's NP ID for *`targetNpId`*. When downloading TUS data of a virtual user, use `sceNpTusGetDataVUserAsync()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotId`*, specify the slot ID of the download source.

Specify a pointer to the structure for storing the status of the TUS data in *`dataStatus`*. Specify `sizeof(SceNpTusDataStatus)` for *`dataStatusSize`*.

For *`data`*, specify the area for storing the data to receive, and specify its size in *`recvSize`*.

The total size of the TUS data will be stored in the *`dataSize`* member of the *`dataStatus`* structure. If this value is greater than the value specified for *`recvSize`*, this means only a part of the data has been received. Call this function again to receive the remaining data. For subsequent calls, the same values should be specified as the first for *`reqId`*, *`targetNpId`* or *`targetVirtualUserId`*, and *`slotId`*. Values specified for *`dataStatus`*, *`data`*, and *`recvSize`* can differ. Determine the end of the data from the application side by comparing the total size of the TUS data and the received data size.

When this function is called with NULL specified for *`data`*, data will not be received and only the data status will be stored in the *`dataStatus`* structure. Thus, this function can be called in this manner for the first call to obtain the total size of the TUS data to receive. A sufficient amount of memory can then be allocated and the entire data received from the second call of this function.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The actual processing will be performed on an internal thread created by `sceNpTusInit()`. Note that the content of the memory specified for *`data`* is undefined until the processing completes. The result of the processing must be obtained after confirming that the processing completed using `sceNpTusWaitAsync()`, for example.

After receiving the entire data and obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

Notes

Until the asynchronous processing completes, another asynchronous processing cannot be issued from the same request ID. Even when using a different request ID, note that there is a possibility of memory running out on the network layer.

Functions for obtaining the data status are `sceNpTusGetMultiSlotDataStatusAsync()` / `sceNpTusGetMultiSlotDataStatusVUserAsync()` and `sceNpTusGetMultiUserDataStatusAsync()` / `sceNpTusGetMultiUserDataStatusVUserAsync()`. If you want to obtain the data status for reasons other than checking the total data size before downloading, use either of the above functions instead of this one.

The *`data`* member of the *`dataStatus`* structure exists for reasons of practicality for managing data. You must be careful when receiving a single TUS data in installments, especially, because the address specified first for the *`data`* argument of this function (other than NULL) will be returned for each call of the function and this will not indicate the address of the received data.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusGetData()`, `sceNpTusGetDataVUser()`

sceNpTusGetMultiSlotDataStatus, sceNpTusGetMultiSlotDataStatusVUser

Get multiple TUS data statuses of 1 user (synchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiSlotDataStatus (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiSlotDataStatusVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the read source (IN)
targetVirtualUserId	ID of virtual user who is the read source (IN)
slotIdArray	Pointer to the array storing the read source slot IDs (IN)
statusArray	Pointer to the array for storing the read statuses of the TUS data (OUT)
statusArraySize	Memory size of *statusArray* (IN)
arrayNum	Valid number of elements for *slotIdArray*. Maximum number is 64 (IN)
option	Option of future extension. Always specify NULL

Return Values

Returns the number of the obtained TUS data statuses (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	A negative value was specified for one of the elements of *slotIdArray*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *statusArray*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT	0x80550714	The structure size specified for *statusArraySize* is invalid
SCE_NP_COMMUNITY_ERROR_TOO_MANY_SLOTID	0x80550718	A value exceeding 64 has been specified for *arrayNum*
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in SceNpCommunicationPassphrase
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER	0x80550849	Specified virtual user does not exist

Description

This function obtains the statuses of multiple TUS data corresponding to the specified slots of the specified user.

When obtaining the statuses of TUS data of an actual user, use `sceNpTusGetMultiSlotDataStatus()` and specify the user's NP ID for *`targetNpId`*. When obtaining the statuses of TUS data of a virtual user, use `sceNpTusGetMultiSlotDataStatusVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the read source slot IDs. This area must be 4-byte-aligned and continuous. Specify the number of specified slot IDs in *`arrayNum`*.

For *`statusArray`*, specify the beginning address of the array for storing the read statuses. This area must be 8-byte-aligned and continuous. The number of elements as specified in *`arrayNum`* is required. Specify *`arrayNum`*`*sizeof(SceNpTusDataStatus)` for *`statusArraySize`*.

The TUS data status of the slot specified by *`slotIdArray`*`[0]` will be read and stored in *`statusArray`*`[0]`. The TUS data status of the slot specified by *`slotIdArray`*`[1]` will be read and stored in *`statusArray`*`[1]`. This process is continued so that statuses are read all at once from the number of TUS data as specified in *`arrayNum`* and stored in *`statusArray`*. If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and read processing will not take place. If a timeout error occurs, however, the processing result will either be that the status is read from all TUS data or from none of them.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusGetMultiSlotDataStatusAsync()`, `sceNpTusGetMultiSlotDataStatusVUserAsync()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

sceNpTusGetMultiSlotDataStatusAsync, sceNpTusGetMultiSlotDataStatusVUserAsync

Get multiple TUS data statuses of 1 user (asynchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiSlotDataStatusAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiSlotDataStatusVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of user who is the read source (IN)
targetVirtualUserId	ID of virtual user who is the read source (IN)
slotIdArray	Pointer to the array storing the read source slot IDs (IN)
statusArray	Pointer to the array for storing the read statuses of the TUS data (OUT)
statusArraySize	Memory size of `statusArray` (IN)
arrayNum	Valid number of elements for `slotIdArray`. Maximum number is 64 (IN)
option	Option of future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *statusArray*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This function obtains the statuses of multiple TUS data corresponding to the specified slots of the specified user.

When obtaining the statuses of TUS data of an actual user, use `sceNpTusGetMultiSlotDataStatusAsync()` and specify the user's NP ID for *`targetNpId`*.
When obtaining the statuses of TUS data of a virtual user, use `sceNpTusGetMultiSlotDataStatusVUserAsync()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the read source slot IDs. This area must be 4-byte-aligned and continuous. Specify the number of specified slot IDs in *`arrayNum`*.

For *`statusArray`*, specify the beginning address of the array for storing the read statuses. This area must be 8-byte-aligned and continuous. The number of elements as specified in *`arrayNum`* is required. Specify *`arrayNum`*`*sizeof(SceNpTusDataStatus)` for *`statusArraySize`*.

The TUS data status of the slot specified by *`slotIdArray`*`[0]` will be read and stored in *`statusArray`*`[0]`. The TUS data status of the slot specified by *`slotIdArray`*`[1]` will be read and stored in *`statusArray`*`[1]`. This process is continued so that statuses are read all at once from the number of TUS data as specified in *`arrayNum`* and stored in *`statusArray`*. If there is even one slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and read processing will not take place. If a timeout error occurs, however, the processing result will either be that the status is read from all TUS data or from none of them.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The actual processing will be performed on an internal thread created by `sceNpTusInit()`. The content of the area where the result will be returned is undefined until the processing completes; make sure to obtain the processing result after confirming that the processing completed using `sceNpTusWaitAsync()`, for example.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

Notes

Until the asynchronous processing completes, another asynchronous processing cannot be issued from the same request ID. Even when using a different request ID, note that there is a possibility of memory running out on the network layer.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusGetMultiSlotDataStatus()`, `sceNpTusGetMultiSlotDataStatusVUser()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

sceNpTusGetMultiUserDataStatus, sceNpTusGetMultiUserDataStatusVUser

Get 1 TUS data status of multiple users (synchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiUserDataStatus (
        SceInt32 reqId,
        const SceNpId targetNpIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiUserDataStatusVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId targetVirtualUserIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpIdArray	Pointer to array storing the NP IDs of users who are the read sources (IN)
targetVirtualUserIdArray	Pointer to array storing the IDs of virtual users who are the read sources (IN)
slotId	Slot ID of the read source (IN)
statusArray	Pointer to array for storing the read statuses of the TUS data (OUT)
statusArraySize	Memory size of `statusArray` (IN)
arrayNum	Valid number of elements for `targetNpIdArray` or `targetVirtualUserIdArray`. Maximum is 101 (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns the number of the obtained TUS data statuses (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
`SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT`	0x80550704	A negative value was specified for *`slotId`*, 0 was specified for *`arrayNum`*, or value other than NULL was specified for *`option`*
`SCE_NP_COMMUNITY_ERROR_ABORTED`	0x80550707	Communication processing was aborted with `sceNpTusAbortRequest()` or `sceNpTusDeleteRequest()`
`SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT`	0x8055070c	NULL was specified for *`targetNpIdArray`*, *`targetVirtualUserIdArray`*, or *`statusArray`*
`SCE_NP_COMMUNITY_ERROR_INVALID_ID`	0x8055070e	ID specified for *`reqId`* does not exist
`SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT`	0x80550714	Structure size specified for *`statusArraySize`* is invalid
`SCE_NP_COMMUNITY_ERROR_TOO_MANY_NPID`	0x80550719	A value exceeding `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM` has been specified for *`arrayNum`*
`SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN`	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA`	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE`	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE`	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE`	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
`SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE`	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND`	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
`SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER`	0x80550849	Specified virtual user does not exist

Description

This function obtains the status of TUS data corresponding to the specified slot of the specified users.

When obtaining the TUS data status of actual users, use `sceNpTusGetMultiUserDataStatus()` and specify the NP ID array for those users in *`targetNpIdArray`*. When obtaining the TUS data status of virtual users, use `sceNpTusGetMultiUserDataStatusVUser()` and specify the array for the virtual user IDs in *`targetVirtualUserIdArray`*. These arrays must be 4-byte-aligned continuous areas. Specify the number of users specified in these arrays in *`arrayNum`*.

For *`slotId`*, specify the slot ID to read out.

For *`statusArray`*, specify the beginning address of the array for storing the read statuses. This area must be 8-byte-aligned and continuous. The number of elements as specified in *`arrayNum`* is required. Specify *`arrayNum`*`*sizeof(SceNpTusDataStatus)` for *`statusArraySize`*.

The TUS data status of the user specified for *`targetNpIdArray`*`[0]` or *`targetVirtualUserIdArray`*`[0]` will be read and stored in *`statusArray`*`[0]`. The TUS data status of the user specified for *`targetNpIdArray`*`[1]` or *`targetVirtualUserIdArray`*`[1]` will be read and stored in *`statusArray`*`[1]`. This process is continued so that statuses are read all at once for the number of TUS data as specified in *`arrayNum`* and stored in *`statusArray`*.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusGetMultiUserDataStatusAsync()`, `sceNpTusGetMultiUserDataStatusVUserAsync()`, `SCE_NP_TUS_MAX_USER_NUM_PER_REQUEST`

sceNpTusGetMultiUserDataStatusAsync, sceNpTusGetMultiUserDataStatusVUserAsync

Get 1 TUS data status of multiple users (asynchronous)

Definition

```
#include <np.h>

int sceNpTusGetMultiUserDataStatusAsync (
        SceInt32 reqId,
        const SceNpId targetNpIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);

int sceNpTusGetMultiUserDataStatusVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId targetVirtualUserIdArray[],
        SceNpTusSlotId slotId,
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpIdArray	Pointer to array storing the NP IDs of users who are the read sources (IN)
targetVirtualUserIdArray	Pointer to array storing the IDs of virtual users who are the read sources (IN)
slotId	Slot ID of the read source (IN)
statusArray	Pointer to array for storing the read statuses of the TUS data (OUT)
statusArraySize	Memory size of `statusArray` (IN)
arrayNum	Valid number of elements for `targetNpIdArray` or `targetVirtualUserIdArray`. Maximum is 101 (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	0 was specified for *arrayNum*, or value other than NULL was specified for *option*

Value	(Number)	Description
SCE_NP_COMMUNITY_ERROR_ INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpIdArray*, *targetVirtualUserIdArray*, or *statusArray*
SCE_NP_COMMUNITY_ERROR_ INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This function obtains the status of TUS data corresponding to the specified slot of the specified users.

When obtaining the TUS data status of actual users, use `sceNpTusGetMultiUserDataStatusAsync()` and specify the NP ID array for those users in *targetNpIdArray*. When obtaining the TUS data status of virtual users, use `sceNpTusGetMultiUserDataStatusVUserAsync()` and specify the array for the virtual user IDs in *targetVirtualUserIdArray*. These arrays must be 4-byte-aligned continuous areas. Specify the number of users specified in these arrays in *arrayNum*.

For *slotId*, specify the slot ID to read out.

For *statusArray*, specify the beginning address of the array for storing the read statuses. This area must be 8-byte-aligned and continuous. The number of elements as specified in *arrayNum* is required. Specify *arrayNum*`*sizeof(SceNpTusDataStatus)` for *statusArraySize*.

The TUS data status of the user specified for *targetNpIdArray*`[0]` or *targetVirtualUserIdArray*`[0]` will be read and stored in *statusArray*`[0]`. The TUS data status of the user specified for *targetNpIdArray*`[1]` or *targetVirtualUserIdArray*`[1]` will be read and stored in *statusArray*`[1]`. This process is continued so that statuses are read all at once for the number of TUS data as specified in *arrayNum* and stored in *statusArray*.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The actual processing will be performed on an internal thread created by `sceNpTusInit()`. The content of the area where the result will be returned is undefined until the processing completes; make sure to obtain the processing result after confirming that the processing completed using `sceNpTusWaitAsync()`, for example.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

Notes

Until the asynchronous processing completes, another asynchronous processing cannot be issued from the same request ID. Even when using a different request ID, note that there is a possibility of memory running out on the network layer.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusGetMultiUserDataStatus()`, `sceNpTusGetMultiUserDataStatusVUser()`, `SCE_NP_TUS_MAX_USER_NUM_PER_REQUEST`

sceNpTusGetFriendsDataStatus

Specify sort conditions and read TUS data statuses for the top user results (up to 100) from among friends (synchronous)

Definition

```
#include <np.h>
int sceNpTusGetFriendsDataStatus (
        SceInt32 reqId,
        SceNpTusSlotId slotId,
        SceBool includeSelf,
        SceInt32 sortType,
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        SceNpTusGetFriendsDataStatusOptParam *option
);
```

Arguments

reqId	Request ID (IN)
slotId	Read source slot ID (IN)
includeSelf	Flag to indicate whether the current user's TUS data status is to be included in the target statuses to obtain or not. Specify any number besides 0 to include and 0 to exclude (IN)
sortType	Friend sort conditions (IN)
statusArray	Pointer to array for storing the read TUS data statuses (OUT)
statusArraySize	Memory size of *statusArray* (IN)
arrayNum	Number of TUS data statuses to obtain. Up to `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM` (100) (IN)
option	Pointer to extended option structure. Specify NULL if not required

Return Values

Returns the number of the read TUS data statuses (>=0) for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_INVALID_ARGUMENT	0x80550704	A negative value was specified for *slotId*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *statusArray*
SCE_NP_COMMUNITY_ERROR_INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_INVALID_ALIGNMENT	0x80550714	The structure size specified for *statusArraySize* is invalid
SCE_NP_COMMUNITY_ERROR_TOO_MANY_NPID	0x80550719	Value exceeding SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM was specified for *arrayNum*
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in SceNpCommunicationPassphrase
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server

Description

This function specifies a single slot and sort conditions, then reads the TUS data statuses for the top friend results.

For *`includeSelf`*, specify any number besides 0 to include the current user in the target TUS data statuses to obtain, and specify 0 to exclude the current user in the target statuses.

For *`arrayNum`*, specify the maximum number of users whose TUS data statuses will be obtained in the top results.

For *`slotId`*, specify the read source slot ID.

For *`statusArray`*, specify the beginning address of the array for storing the read TUS data statuses. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *`statusArraySize`*. Specify *`arrayNum`*`*sizeof(SceNpTusDataStatus)` to *`statusArraySize`*.

For *`sortType`*, specify the friend sort conditions with one of the following values.

Value	Description
`SCE_NP_TUS_DATASTATUS_SORTTYPE_DESCENDING_DATE`	Sorts by last update date/time in descending order
`SCE_NP_TUS_DATASTATUS_SORTTYPE_ASCENDING_DATE`	Sorts by last update date/time in ascending order

In *`statusArray`*, the friend TUS data statuses will be stored from the top sort result up to the number specified with *`arrayNum`*.

The number of stored TUS data statuses will return as the function return value.

If not even one friend TUS data status was registered, the return value will be 0 and normal termination will occur.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusGetFriendsVariable()`, `sceNpTusGetFriendsVariableAsync()`, `sceNpTusGetFriendsDataStatusAsync()`, `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM`, `SceNpTusGetFriendsDataStatusOptParam`

sceNpTusGetFriendsDataStatusAsync

Specify sort conditions and read TUS data statuses for the top user results (up to 100) from among friends (asynchronous)

Definition

```
#include <np.h>
int sceNpTusGetFriendsDataStatusAsync (
        SceInt32 reqId,
        SceNpTusSlotId slotId,
        SceBool includeSelf,
        SceInt32 sortType,
        SceNpTusDataStatus statusArray[],
        SceSize statusArraySize,
        SceInt32 arrayNum,
        SceNpTusGetFriendsDataStatusOptParam *option
);
```

Arguments

reqId	Request ID (IN)
slotId	Read source slot ID (IN)
includeSelf	Flag to indicate whether the current user's TUS data status is to be included in the target statuses to obtain or not. Specify any number besides 0 to include and 0 to exclude (IN)
sortType	Friend sort conditions (IN)
statusArray	Pointer to array for storing the read TUS data statuses (OUT)
statusArraySize	Memory size of *statusArray* (IN)
arrayNum	Number of TUS data statuses to obtain. Up to SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM (100) (IN)
option	Pointer to extended option structure Specify NULL if not required

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_ INVALID_ARGUMENT	0x80550704	A negative value was specified for *slotId*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *statusArray*
SCE_NP_COMMUNITY_ERROR_ INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_ INVALID_ALIGNMENT	0x80550714	The structure size specified for *statusArraySize* is invalid
SCE_NP_COMMUNITY_ERROR_ TOO_MANY_NPID	0x80550719	Value exceeding SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM was specified for *arrayNum*

Description

This function specifies a single slot and sort conditions, then reads the TUS data statuses for the top friend results.

For *includeSelf*, specify any number besides 0 to include the current user in the target TUS data statuses to obtain, and specify 0 to exclude the current user in the target statuses.

For *arrayNum*, specify the maximum number of users whose TUS data statuses will be obtained in the top results.

For *slotId*, specify the read source slot ID.

For *statusArray*, specify the beginning address of the array for storing the read TUS data statuses. This area must be 8-byte-aligned and continuous, and its size must be larger than or equal to the value specified for *statusArraySize*. Specify *arrayNum*`*sizeof(SceNpTusDataStatus)` to *statusArraySize*.

For *sortType*, specify the friend sort conditions with one of the following values.

Value	Description
`SCE_NP_TUS_DATASTATUS_SORTTYPE_DESCENDING_DATE`	Sorts by last update date/time in descending order
`SCE_NP_TUS_DATASTATUS_SORTTYPE_ASCENDING_DATE`	Sorts by last update date/time in ascending order

In *statusArray*, the friend TUS data statuses will be stored from the top sort result up to the number specified with *arrayNum*.

The number of TUS data statuses stored in *statusArray* will be stored in *result* in `sceNpTusWaitAsync()`/`sceNpTusPollAsync()`.

If not even one friend TUS data status was registered, 0 will be stored in *result* and normal termination will occur.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained using `sceNpTusWaitAsync()`, for example. The processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusGetFriendsVariable()`, `sceNpTusGetFriendsVariableAsync()`, `sceNpTusGetFriendsDataStatus()`, `SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM`, `SceNpTusGetFriendsDataStatusOptParam`

sceNpTusDeleteMultiSlotData, sceNpTusDeleteMultiSlotDataVUser

Delete multiple TUS data of 1 user (synchronous)

Definition

```
#include <np.h>

int sceNpTusDeleteMultiSlotData (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);

int sceNpTusDeleteMultiSlotDataVUser (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotIdArray	Pointer to the array storing the target slot IDs (IN)
arrayNum	Valid number of elements for *slotIdArray*. Maximum is 64 (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_ INVALID_ARGUMENT	0x80550704	A negative value was specified for one of the elements of *slotIdArray*, 0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ ABORTED	0x80550707	Communication processing was aborted with sceNpTusAbortRequest() or sceNpTusDeleteRequest()
SCE_NP_COMMUNITY_ERROR_ INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *slotIdArray*
SCE_NP_COMMUNITY_ERROR_ INVALID_ID	0x8055070e	ID specified for *reqId* does not exist
SCE_NP_COMMUNITY_ERROR_ TOO_MANY_SLOTID	0x80550718	Value exceeding 64 was specified for *arrayNum*

Value	(Number)	Description
SCE_NP_COMMUNITY_SERVER_ERROR_FORBIDDEN	0x80550806	Specified slot does not exist, or access privilege has not been granted for a slot
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_ANTICHEAT_DATA	0x80550819	Authentication error. Usually occurs with an error in `SceNpCommunicationPassphrase`
SCE_NP_COMMUNITY_SERVER_ERROR_NO_SUCH_TITLE	0x805508a6	Specified NP Communication ID is not registered on the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_BEFORE_SERVICE	0x805508aa	Title user storage service for the specified NP Communication ID has not started yet (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_END_OF_SERVICE	0x805508ab	Title user storage service for the specified NP Communication ID has already terminated (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_TITLE_USER_STORAGE_MAINTENANCE	0x805508ac	Title user storage service for the specified NP Communication ID is currently under maintenance (Corresponding message is in the Message Dialog library)
SCE_NP_COMMUNITY_SERVER_ERROR_USER_STORAGE_TITLE_MASTER_NOT_FOUND	0x80550848	Specified NP Communication ID is not registered on the database of the title user storage server
SCE_NP_COMMUNITY_SERVER_ERROR_INVALID_VIRTUAL_USER	0x80550849	Specified virtual user does not exist

Description

This function deletes multiple TUS data corresponding to the specified slots of the specified user. The deleted TUS data will return to a state where no data is set to them.

When deleting the TUS data of an actual user, use `sceNpTusDeleteMultiSlotData()` and specify the user's NP ID for *`targetNpId`*. When deleting the TUS data of a virtual user, use `sceNpTusDeleteMultiSlotDataVUser()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the slot IDs of the TUS data to delete. This area must be 4-byte-aligned and continuous. Specify the number of specified slot IDs for *`arrayNum`*.

If there is even 1 slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and delete processing will not take place. If a timeout error occurs, however, the processing result will either be that data is deleted from all the TUS data or from none of them.

This function performs synchronous processing. In other words, other processes are blocked until communication completes and the result can be obtained. After returning from this function, delete the used request ID by calling `sceNpTusDeleteRequest()`.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusDeleteMultiSlotDataAsync()`, `sceNpTusDeleteMultiSlotDataVUserAsync()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

sceNpTusDeleteMultiSlotDataAsync, sceNpTusDeleteMultiSlotDataVUserAsync

Delete multiple TUS data of 1 user (asynchronous)

Definition

```
#include <np.h>

int sceNpTusDeleteMultiSlotDataAsync (
        SceInt32 reqId,
        const SceNpId *targetNpId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);

int sceNpTusDeleteMultiSlotDataVUserAsync (
        SceInt32 reqId,
        const SceNpTusVirtualUserId *targetVirtualUserId,
        const SceNpTusSlotId slotIdArray[],
        SceInt32 arrayNum,
        void *option
);
```

Arguments

reqId	Request ID (IN)
targetNpId	NP ID of the target user (IN)
targetVirtualUserId	ID of the target virtual user (IN)
slotIdArray	Pointer to the array storing the target slot IDs (IN)
arrayNum	Valid number of elements for `slotIdArray`. Maximum is 64 (IN)
option	Option for future extension. Always specify NULL

Return Values

Returns 0 for normal termination.

Returns a negative value for errors. 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	Library is not initialized
SCE_NP_COMMUNITY_ERROR_ INVALID_ARGUMENT	0x80550704	0 was specified for *arrayNum*, or value other than NULL was specified for *option*
SCE_NP_COMMUNITY_ERROR_ INSUFFICIENT_ARGUMENT	0x8055070c	NULL was specified for *targetNpId*, *targetVirtualUserId*, or *slotIdArray*
SCE_NP_COMMUNITY_ERROR_ INVALID_ID	0x8055070e	ID specified for *reqId* does not exist

Description

This function deletes multiple TUS data corresponding to the specified slots of the specified user. The deleted TUS data will return to a state where no data is set to them.

When deleting the TUS data of an actual user, use `sceNpTusDeleteMultiSlotDataAsync()` and specify the user's NP ID for *`targetNpId`*. When deleting the TUS data of a virtual user, use `sceNpTusDeleteMultiSlotDataVUserAsync()` and specify the virtual user's ID for *`targetVirtualUserId`*.

For *`slotIdArray`*, specify the beginning address of the array storing the slot IDs of the TUS data to delete. This area must be 4-byte-aligned and continuous. Specify the number of specified slot IDs for *`arrayNum`*.

If there is even 1 slot that does not exist, or if there is a slot for which access privilege has not been granted, the function will return an error and delete processing will not take place. If a timeout error occurs, however, the processing result will either be that data is deleted from all the TUS data or from none of them.

This function performs asynchronous processing. In other words, it returns without waiting for the processing to end. The result of the processing must be obtained by using `sceNpTusWaitAsync()`, for example. The actual processing will be performed on an internal thread created by `sceNpTusInit()`.

After obtaining the result of the processing, call `sceNpTusDeleteRequest()` to delete the used request ID.

Notes

Until the asynchronous processing completes, another asynchronous processing cannot be issued from the same request ID.

See Also

`sceNpTusCreateRequest()`, `sceNpTusDeleteRequest()`, `sceNpTusAbortRequest()`, `sceNpTusWaitAsync()`, `sceNpTusPollAsync()`, `sceNpTusDeleteMultiSlotData()`, `sceNpTusDeleteMultiSlotDataVUser()`, `SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST`

Constants

SCE_NP_TUS_DATA_INFO_MAX_SIZE

Maximum size of accessory information for TUS data

Definition

```
#include <np.h>

#define SCE_NP_TUS_DATA_INFO_MAX_SIZE      384
```

Description

This constant indicates the maximum number of bytes allowed for accessory information of TUS data.

See Also

`SceNpTusDataInfo`

SCE_NP_TUS_MAX_CTX_NUM

Maximum number of contexts

Definition

```
#include <np.h>

#define SCE_NP_TUS_MAX_CTX_NUM      (32)
```

Description

This constant indicates the maximum number of contexts that can exist at one time in the NP TUS library.

This number is both the maximum for title contexts and the maximum for request IDs.

See Also

`sceNpTusCreateTitleCtx(),sceNpTusCreateRequest()`

SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST

Maximum number of slots that can be targeted at one time

Definition

```
#include <np.h>

#define SCE_NP_TUS_MAX_SLOT_NUM_PER_REQUEST        (64)
```

Description

This constant indicates the maximum number of slot IDs that can be specified at one time in a communication processing that targets multiple slots.

SCE_NP_TUS_MAX_USER_NUM_PER_REQUEST

Maximum number of users that can be targeted at one time

Definition

```
#include <np.h>

#define SCE_NP_TUS_MAX_USER_NUM_PER_REQUEST        (101)
```

Description

This constant indicates the maximum number of users that can be specified at one time in a communication processing that targets multiple users.

`SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM` is provided separately for communication processing that targets friends.

SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM

Maximum number of users that can be obtained with communication processing that targets friends

Definition

```
#include <np.h>
#define SCE_NP_TUS_MAX_SELECTED_FRIENDS_NUM       (100)
```

Description

This constant indicates the maximum number of users that can be specified in communication processing that targets friends.

In order to not depend on changes in the maximum number of friends, it is possible to obtain the TUS variables and TUS data statuses for the users in the top 100 sort results regardless of the total number of friends.

See Also

`sceNpTusGetFriendsVariable()`, `sceNpTusGetFriendsVariableAsync()`, `sceNpTusGetFriendsDataStatus()`, `sceNpTusGetFriendsDataStatusAsync()`