NP Auth Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Ticket Related Features

Initialization/Termination

sceNpAuthInit

Initialize the NP Auth library

Definition

```
#include <np_auth.h>
int sceNpAuthInit(
        void
);
```

Arguments

None

Return Values

Returns 0 for normal termination.

Returns an error code upon error.

Description

This function initializes the NP Auth library.

Notes

This function is not multithread safe. Calling this function from multiple threads at the same time will lead to undefined operation.

Example

```
#include <np_auth.h>
int ret;
ret = sceNpAuthInit();
if(ret < 0){
        // Error handling
}
```

See Also

sceNpAuthTerm()

sceNpAuthTerm

Terminate the NP Auth library

Definition

```
#include <np_auth.h>
void sceNpAuthTerm(
        void
);
```

Arguments

None

Return Values

None

Description

This function terminates the NP Auth library.

Notes

This function is not multithread safe. Calling this function from multiple threads at the same time will lead to undefined operation.

Example

```
sceNpAuthTerm();
```

See Also

```
sceNpAuthInit()
```

Obtaining a Ticket

SceNpAuthRequestParameter

Ticket request parameters

Definition

```
#include <np_auth.h>
typedef struct SceNpAuthRequestParameter {
        SceSize size;
        SceNpTicketVersion version;
        const char *serviceId;
        const void *cookie;
        SceSize cookieSize;
        const char *entitlementId;
        SceUInt32 consumedCount;
        int (*ticketCb)(SceNpAuthRequestId, int, void *);
        void *cbArg;
} SceNpAuthRequestParameter;
```

Members

size	SceNpAuthRequestParameter structure size (required)
version	Ticket version (required)
serviceId	Service ID (required)
cookie	Cookie (optional)
cookieSize	Cookie size (optional)
entitlementId	Entitlement ID to execute consumption processing (optional)
consumedCount	Consumption count (optional)
ticketCb	Ticket obtainment callback function (required)
cbArg	User-specified callback argument (optional)

Description

This structure specifies the parameters to obtain a ticket for.

Meaningful values must be specified for (required) items. If (optional) items are not required, specify 0 or NULL.

For *version*, specify the version of the ticket you want to obtain.

cookie and *cookieSize* are members for specifying cookie data to add to the ticket. An arbitrary data in the binary format of up to 1024 bytes can be specified; this data will then be included as cookie data in the ticket issued by the server of PSN℠. A cookie can be used for various purposes - such as, managing a session when the application communicates with an external server.

If a cookie is not required, specify NULL to *cookie* and 0 to *cookieSize*.

entitlementId and *consumedCount* are members for consuming a service entitlement with a usable quantity/frequency set to it. When these members are specified, the server of PSN℠ checks the usable quantity/frequency of the specified service entitlement. If a usable quantity/frequency of *consumedCount* or more remains, a deduction of *consumedCount* is made and a ticket representing the state after the deduction (remaining quantity/frequency of the service entitlement) is issued.

When it is not necessary to consume a service entitlement, specify NULL to *entitlementId* and 0 to *consumedCount*.

For *ticketCb*, specify the callback function to be called when a response is obtained from the server of PSN℠. For *cbArg*, specify any arguments to provide the callback function, if any. For specifications on the callback function, refer to the explanation under the "npauthhandler" section.

See Also

SceNpTicketVersion, sceNpAuthCreateStartRequest(), npauthhandler

SceNpTicketVersion

Ticket version

Definition

```
#include <np_auth.h>
#define SCE_NP_AUTH_LATEST_TICKET_VERSION_MAJOR     3
#define SCE_NP_AUTH_LATEST_TICKET_VERSION_MINOR     0
typedef struct SceNpTicketVersion {
        unsigned short major;
        unsigned short minor;
} SceNpTicketVersion;
```

Members

major Major version
minor Minor version

Description

This structure represents the ticket version.

A currently usable ticket version is as follows.

Ticket Version	Value to Specify to *version*	
	Major Version (*version.major*)	Minor Version (*version.minor*)
3.0	3	0

Depending on the ticket version, ticket data contents issued by the server of PSN℠ slightly change. Unless there is a special reason to do so, specify version 3.0.

See Also

`sceNpAuthCreateStartRequest()`

sceNpAuthCreateStartRequest

Start ticket request processing

Definition

```
#include <np_auth.h>
int sceNpAuthCreateStartRequest(
        const SceNpAuthRequestParameter *param
);
```

Arguments

param Ticket request parameters

Return Values

Returns a request ID (1 or more) for normal termination.

Returns an error code (negative value) upon error.

Description

This function starts the processing to obtain a ticket. Call this function with appropriate values set to each member of *param*. For optional items, make sure to specify 0 or NULL if they are not required.

This function returns immediately after sending a request to the server of PSN℠ without waiting for a response.

When a response is gained from the server of PSN℠, the callback function specified via *param*->*ticketCb* will be called and the size of the obtained ticket (or the error code returned by the server of PSN℠) will be passed.

The request ID returned by this function will be used for obtaining a ticket and for deleting the request – have it saved in an appropriate variable.

Notes

The callback function specified via *param*->*ticketCb* will be called after a response is gained from the server of PSN℠ and when the application calls `sceNpCheckCallback()`. After calling this function, carry out appropriate processing and also periodically call `sceNpCheckCallback()` while waiting for the callback function to be called.

Example

```
#define AUTH_SERVICE_ID "XXX-XXXXX-XX"
void
authCallback(SceNpAuthRequestId id, int result, void *arg)
{
        (abridged)
}

SceNpAuthRequestParameter param;
SceNpAuthRequestId id;

memset(&param, 0x00, sizeof(SceNpAuthRequestParameter));

param.size = sizeof(SceNpAuthRequestParameter);
param.version.major = SCE_NP_AUTH_LATEST_TICKET_VERSION_MAJOR;
param.version.minor = SCE_NP_AUTH_LATEST_TICKET_VERSION_MINOR;
param.serviceId = AUTH_SERVICE_ID;
param.ticketCb = authCallback;
param.cbArg = NULL;

ret = sceNpAuthCreateStartRequest(&param);
if (ret < 0){
        // Error handling
} else {
        id = ret;
}
```

See Also

SceNpAuthRequestParameter,npauthhandler,sceNpAuthDestroyRequest()

npauthhandler

Prototype of the ticket obtainment callback function

Definition

```
#include <np_auth.h>
int npauthhandler(
        SceNpAuthRequestId id,
        int result,
        void *arg
);
```

Arguments

id	Request ID
result	Size of the obtained ticket (positive value other than 0) or an error code (negative value) returned by the server
arg	*param->cbArg* specified in sceNpAuthCreateStartRequest()

Description

This is a prototype for the ticket obtainment callback function for receiving the result of a request sent to the server of PSN℠ using sceNpAuthCreateStartRequest().

The result of ticket obtainment is passed to *result*. When a ticket is successfully obtained, the size of the ticket in bytes will be passed to *result* as a positive value.

When a positive value is passed to *result*, the callback function can internally call sceNpAuthGetTicket() to obtain a ticket.

When *result* is a negative value, make sure to return immediately from the callback function.

In both cases, make sure the return value of the callback function is 0.

Example

```
static unsigned char *ticket=NULL;    // Buffer to store ticket
static int ticket_len = 0;            // Ticket size
static int auth_result=0;             // Callback result

int
npauthhandler(SceNpAuthRequestId id, int result, void *arg)
{
        int ret = 0;
        (void)arg;

        // Upon error, result will be a negative value
        if (result < 0) {
              auth_result = result;
              return 0;
        }

        // Obtain ticket
        ticket_len = result;
        ticket = (unsigned char *)malloc(result);
        if (ticket == NULL) {
              auth_result = -1;
              return 0;
```

```
            }
            auth_result = sceNpAuthGetTicket(id, ticket, result);

            return 0;
}
```

Notes

The ticket obtainment callback function is called from an internal thread of the NP Auth library.

Avoid a synchronized wait [`sceKernelWaitSema()` for the same semaphore, for example] between the thread calling an API of the NP library and the processing within the ticket obtainment callback function. Both the thread calling the NP library API and the internal thread of the NP Auth library may fall into a deadlock.

See Also

`SceNpAuthRequestParameter`, `sceNpAuthCreateStartRequest()`

sceNpAuthGetTicket

Get ticket data

Definition

```
#include <np_auth.h>
int sceNpAuthGetTicket(
        SceNpAuthRequestId id,
        void *buf,
        SceSize len
);
```

Arguments

id	Request ID
buf	Buffer to store ticket data
len	Ticket size to copy

Return Values

Returns 0 for normal termination.

Returns an error code upon error.

Description

This function obtains ticket data (in the form of binary data) from within the library, where information received from the server of PSN℠ was stored.

Always call this function from within the ticket obtainment callback function. When the call is successful, ticket data of the size specified in *len* will be copied to *buf*.

Notes

This function is not multithread safe. Calling this function from multiple threads at the same time for the same `SceNpAuthRequestId` will lead to undefined operation.

Example

Refer to the "`npauthhandler`" section.

See Also

`npauthhandler`

Document serial number: 000004892117

sceNpAuthAbortRequest

Abort ticket request processing

Definition

```
#include <np_auth.h>
int sceNpAuthAbortRequest(
        SceNpAuthRequestId id
);
```

Arguments

id Request ID

Return Values

Returns 0 for normal termination.

Returns an error code upon error.

Description

This function aborts the request to obtain the ticket specified in *id*.

If the specified request is being processed, it will be aborted and the ticket obtainment callback function will be called. In this case, an error indicating processing abort will be passed to *result*.

Delete an aborted request using `sceNpAuthDestroyRequest()`.

Example

```
SceNpAuthRequestId id;

ret = sceNpAuthAbortRequest(id);
if (ret < 0){
        // Error handling
}
```

See Also

`sceNpAuthCreateStartRequest()`

sceNpAuthDestroyRequest

Delete a request

Definition

```
#include <np_auth.h>
int sceNpAuthDestroyRequest(
        SceNpAuthRequestId id
);
```

Arguments

id Request ID

Return Values

Returns 0 for normal termination.

Returns an error code upon error.

Description

This function deletes the ticket obtainment request specified in *id* and frees allocated resources. Use this function to delete a request after the desired ticket data is obtained, or after the ticket obtainment request is aborted.

If the specified request is being processed, this function returns SCE_NP_AUTH_ERROR_EBUSY.

Example

```
SceNpAuthRequestId id;

ret = sceNpAuthDestroyRequest(id);
if (ret < 0){
        // Error handling
}
```

See Also

sceNpAuthCreateStartRequest(),sceNpAuthAbortRequest()

Checking Entitlement

SceNpEntitlementId

Entitlement ID

Definition

```
#include <np_auth.h>
#define SCE_NP_ENTITLEMENT_ID_SIZE      (32)
typedef struct SceNpEntitlementId {
        unsigned char data[SCE_NP_ENTITLEMENT_ID_SIZE];
} SceNpEntitlementId;
```

Members

data Entitlement ID character string data

Description

This is a fixed-length structure used when obtaining a list of entitlement IDs using `sceNpAuthGetEntitlementIdList()`. *data* will be passed with the NULL terminator.

See Also

`sceNpAuthGetEntitlementIdList()`

SceNpEntitlement

Entitlement data

Definition

```
#include <np_auth.h>
typedef struct SceNpEntitlement {
        SceNpEntitlementId id;
        SceNpTime createdDate;
        SceNpTime expireDate;
        SceUInt32 type;
        SceInt32 remainingCount;  /* may be negative */
        SceUInt32 consumedCount;
        char padding[4];
} SceNpEntitlement;
```

Members

id	Entitlement ID
createdDate	Entitlement's created time
expireDate	Entitlement's expiration time
type	Entitlement type
remainingCount	Remaining count of a consumable entitlement (usable quantity/frequency)
consumedCount	Total consumed count of a consumable entitlement
padding	Padding

One of the following values will be inserted for *type*.

Value	(Number)	Description
SCE_NP_ENTITLEMENT_TYPE_NON_CONSUMABLE	0	Entitlement of the standard type or entitlement with a validity period
SCE_NP_ENTITLEMENT_TYPE_CONSUMABLE	1	Consumable entitlement (with a fixed number of usable quantity/frequency)

Description

This structure stores entitlement data obtained by sceNpAuthGetEntitlementById().

When an entitlement with a validity period reaches its expiration, the privilege to use it also expires at the same time; the entitlement will then be excluded from the ticket. In addition, consumable entitlements will no longer be included in a ticket when *remaining_count* reaches 0. The application should judge whether the user has privilege to use a specific entitlement or not by checking whether that entitlement is included in a valid ticket (regardless of *expireDate*).

See Also

sceNpAuthGetEntitlementById()

sceNpAuthGetEntitlementIdList

Get list of entitlement IDs

Definition

```
#include <np_auth.h>
int sceNpAuthGetEntitlementIdList(
        const unsigned char *ticket,
        SceSize ticketSize,
        SceNpEntitlementId *entIdList,
        SceSize entIdListNum
);
```

Arguments

ticket	Pointer to ticket data
ticketSize	Size of ticket data
entIdList	Pointer to destination array storing the obtained list of entitlement IDs
entIdListNum	Number of elements in the destination array

Return Values

Returns the number of entitlements included in the ticket (0 or more) for normal termination.

Returns an error code upon error.

Description

This function counts the number of entitlements included in the ticket data specified by *ticket* and *ticketSize*, and returns a list of entitlement IDs in the array specified by *entIdList* and *entIdListNum*.

This function is typically called 2 times in a row, as exemplified below.

For the first call, specify NULL to *entIdList*. Because the return value represents the total number of entitlements included in the ticket data, a buffer that is able to store at least that many number of SceNpEntitlementId elements must be prepared.

For the second call, specify that buffer's address to *entIdList* and the number of entitlement IDs in that buffer to *entIdListNum*. This will ensure the storage of all entitlement IDs in the ticket data.

If more numbers of entitlements are included in the ticket data than what is specified in *entIdListNum*, only the specified number of entitlement IDs will be stored in the buffer.

The maximum number of entitlements that can be stored in a ticket is 2500 for ticket version 3.0.

Example

```
unsigned char *ticket;
SceSize ticket_len, entIdListNum:
SceNpEntitlementId *entIdList;

// Assume ticket is obtained in the buffer specified by ticket and ticket_len

// If NULL is specified to the 3rd argument,
// only the number of entitlements will be returned
ret = sceNpAuthGetEntitlementIdList(ticket, ticket_len, NULL, 0);
if (ret < 0){
        // Error handling
}
entIdListNum = ret;

// Allocate area required for entIdList
entIdList = (SceNpEntitlementId *)malloc(entIdListNum *
sizeof(SceNpEntitlementId));
if(entIdList == NULL){
        // Error handling
}

// Obtain list of entitlement IDs
ret = sceNpAuthGetEntitlementIdList(ticket, ticket_len, entIdList,
entIdListNum);
if (ret < 0){
        // Error handling
}
```

sceNpAuthGetEntitlementById, sceNpAuthCheckEntitlementById

Get entitlement data/Check entitlement data

Definition

```
#include <np_auth.h>
int sceNpAuthGetEntitlementById(
        const unsigned char *ticket,
        SceSize ticketSize,
        const char *entId,
        SceNpEntitlement *ent
);

#define sceNpAuthCheckEntitlementById(ticket, ticketSize, entId) \
        sceNpAuthGetEntitlementById((ticket), (ticketSize), (entId), NULL)
```

Arguments

ticket	Pointer to ticket data
ticketSize	Size of ticket data
entId	Target entitlement ID
ent	Storage destination of obtained entitlement data

Return Values

Returns 0 for normal termination.

Returns an error code upon error.

Description

This function extracts entitlement data of the entitlement specified by *entId* from the ticket data specified by *ticket* and *ticketSize*, and returns this data to *ent*.

When NULL is specified to *ent*, the function only checks whether the entitlement exists within the ticket data or not, and does not return any entitlement data. This method can be used to check user privilege; the `sceNpAuthCheckEntitlementById()` macro is exclusively provided for this purpose.

Example

```
unsigned char *ticket;
SceSize ticket_len;
const char *entId;
SceNpEntitlement ent;

// Assume ticket is obtained in the buffer specified by ticket and ticket_len

// Assume target entitlement ID is stored in entId
ret = sceNpAuthGetEntitlementById(ticket, ticket_len, entId, &ent);
if (ret < 0){
        // Error handling
}
```

Obtaining Ticket Parameters

SceNpTicketParam

Ticket parameter

Definition

```
#include <np_auth.h>
#define SCE_NP_TICKET_PARAM_DATA_LEN    (256)
union SceNpTicketParam{
        SceInt32 i32;
        SceInt64 i64;
        SceUInt32 u32;
        SceUInt64 u64;
        SceNpDate date;
        SceUChar8 data[SCE_NP_TICKET_PARAM_DATA_LEN];
};
```

Members

i32	Signed 32-bit integer type
i64	Signed 64-bit integer type
u32	Unsigned 32-bit integer type
u64	Unsigned 64-bit integer type
date	`SceNpDate` type
data	Binary type or string type

Description

This union is used to obtain a ticket parameter using `sceNpAuthGetTicketParam()`. A ticket parameter is stored in certain members according to its data type; read value from the appropriate member.

Ticket parameters of the string type are all NULL-terminated. Binary type ticket parameters are not guaranteed to be NULL-terminated.

See Also

`sceNpAuthGetTicketParam()`

SceNpDate

Date information

Definition

```
#include <np_auth.h>
typedef struct SceNpDate {
        SceUShort16 year;
        SceUChar8 month;
        SceUChar8 day;
} SceNpDate;
```

Members

year	Year (4 digits)
month	Month (1 – 12)
day	Day (1 – 31)

Description

This structure stores date information.

See Also

`sceNpAuthGetTicketParam()`

SceNpTime

Time information

Definition

```
#include <np_auth.h>
typedef SceUInt64 SceNpTime;
```

Description

This represents time in the POSIX time_t format, as an accumulation of milliseconds from January 1, 1970 (GMT).

See Also

`sceNpAuthGetTicketParam()`

sceNpAuthGetTicketParam

Get ticket parameter

Definition

```
#include <np_auth.h>
int sceNpAuthGetTicketParam(
        const unsigned char *ticket,
        SceSize ticketSize,
        int paramId,
        union SceNpTicketParam *param
);
```

Arguments

ticket	Pointer to ticket data
ticketSize	Size of ticket data
paramId	ID of the target ticket parameter to be obtained
param	Destination to store obtained ticket parameter

Return Values

Returns 0 for normal termination.

Returns an error code upon error.

Description

This function obtains a ticket parameter specified by *paramId* from the ticket data specified by *ticket* and *ticketSize*, and stores this parameter in *param*.

Specify one of the following values for *paramId*.

Value	(Number)	Description	Data Type
SCE_NP_TICKET_PARAM_SERIAL_ID	0	Ticket serial ID	Binary
SCE_NP_TICKET_PARAM_ISSUER_ID	1	Ticket issuer ID	SceUInt32
SCE_NP_TICKET_PARAM_ISSUED_DATE	2	Ticket issued time	SceNpTime
SCE_NP_TICKET_PARAM_EXPIRE_DATE	3	Ticket expiration time	SceNpTime
SCE_NP_TICKET_PARAM_SUBJECT_ACCOUNT_ID	4	User account ID	SceUInt64
SCE_NP_TICKET_PARAM_SUBJECT_ONLINE_ID	5	User Online ID	String
SCE_NP_TICKET_PARAM_SUBJECT_REGION	6	User's region	Binary
SCE_NP_TICKET_PARAM_SUBJECT_DOMAIN	7	User's domain	String
SCE_NP_TICKET_PARAM_SERVICE_ID	8	Service ID	String
SCE_NP_TICKET_PARAM_SUBJECT_STATUS	9	User status	SceUInt32
SCE_NP_TICKET_PARAM_STATUS_DURATION	10	User status (connection duration)	SceNpTime
SCE_NP_TICKET_PARAM_SUBJECT_DOB	11	Birth date	SceNpDate
SCE_NP_TICKET_PARAM_MAX	12	Maximum parameter value. Specifying a value greater than this will result in an error.	-

The data stored in *param* is a union. Appropriately perform processing for a specific parameter according to its data type as indicated in the above table. Several macros are provided to parse ticket parameters – such as, one to extract the user's age from the user status parameter.

Example

```
unsigned char *ticket;
SceSize ticket_len:
union SceNpTicketParam p;

// Assume ticket is obtained in the buffer specified by ticket and ticket_len

ret = sceNpAuthGetTicketParam(ticket, ticket_len,
SCE_NP_TICKET_PARAM_SUBJECT_STATUS, &p);
if (ret < 0){
        // Error handling
}

printf("Age: %u\n", SCE_NP_SUBJECT_STATUS_GET_AGE(p.u32));
```

See Also

SceNpTicketParam, SceNpDate, SceNpTime

SCE_NP_SUBJECT_ONLINE_ID_GET_ONLINE_ID

Get the Online ID

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_ONLINE_ID_GET_ONLINE_ID(raw, onlineid) do { \
        for (int i=0; i<SCE_NP_ONLINEID_MAX_LENGTH; i++) { \
          (onlineid)->data[i] = (raw)[i]; \
        } \
        (onlineid)->term = '\0'; \
        } while (0)
```

Description

This macro parses a ticket parameter and obtains the user's Online ID as a value of the `SceNpOnlineId` type.

Specify the Online ID string (`SceNpTicketParam.data`) - obtained by specifying `SCE_NP_TICKET_PARAM_SUBJECT_ONLINE_ID` to *`paramId`* in `sceNpAuthGetTicketParam()` - to *`raw`*; and specify the pointer to the `SceNpOnlineId` structure to *`onlineid`*. The Online ID can be obtained from `*`*`onlineid`*.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam`

SCE_NP_SUBJECT_STATUS_GET_AGE

Get age

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_STATUS_GET_AGE(u32) (((u32)>>24)&0x7f)
```

Description

This macro parses a ticket parameter and obtains the user's age.

Specify the user status (`SceNpTicketParam.u32`) - obtained by specifying `SCE_NP_TICKET_PARAM_SUBJECT_STATUS` to *`paramId`* in `sceNpAuthGetTicketParam()` – to *`u32`*. The user's age can be obtained.

Notes

The day when the user's age is incremented is pursuant to the tradition of the user's residing country and region; however, the timing is based on GMT. For example, in a country and region where age is added by 1 on one's birth date and the time zone of this country and region is GMT+9, age will be incremented 9 hours after the date becomes one's birthday in that country and region.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam`

SCE_NP_SUBJECT_STATUS_IS_SUSPENDED

Get service suspended flag

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_STATUS_IS_SUSPENDED(u32)    ((u32)&0x80)
```

Description

This macro parses a ticket parameter and obtains information on whether the user is temporarily suspended from using the PSN℠ services.

Specify the user status (`SceNpTicketParam.u32`) - obtained by specifying `SCE_NP_TICKET_PARAM_SUBJECT_STATUS` to *`paramId`* in `sceNpAuthGetTicketParam()` – to *`u32`*. A flag indicating whether the user is temporarily suspended from using the PSN℠ services or not can be obtained.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam`

SCE_NP_SUBJECT_STATUS_IS_CHAT_DISABLED

Get chat disabled flag

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_STATUS_IS_CHAT_DISABLED(u32)      ((u32)&0x100)
```

Description

This macro parses a ticket parameter and obtains information on whether chat is disabled for the user or not.

Specify the user status (`SceNpTicketParam.u32`) - obtained by specifying `SCE_NP_TICKET_PARAM_SUBJECT_STATUS` to *`paramId`* in `sceNpAuthGetTicketParam()` – to *`u32`*. A flag indicating whether the master account (user's parent or guardian) is prohibiting the user from using the chat feature or not, can be obtained.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam`

SCE_NP_SUBJECT_STATUS_CONTENT_RATING

Get parental control flag

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_STATUS_CONTENT_RATING(u32)        ((u32)&0x200)
```

Description

This macro parses a ticket parameter and obtains whether parental control is required or not.

Specify the user status (`SceNpTicketParam.u32`) - obtained by specifying `SCE_NP_TICKET_PARAM_SUBJECT_STATUS` to *`paramId`* in `sceNpAuthGetTicketParam()` – to *`u32`*. A flag indicating whether parental control should be placed in effect according to the user's age and the rating of the content can be obtained.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam`

SCE_NP_SUBJECT_REGION_GET_COUNTRY_CODE

Get country/region code

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_REGION_GET_COUNTRY_CODE(raw, cc) do { \
        (cc)->data[0] = *((char *)(raw)); \
        (cc)->data[1] = *((char *)(raw)+1); \
        (cc)->term = '\0'; \
        } while (0)
```

Description

This macro parses a ticket parameter and obtains the user's country/region code.

Specify `SCE_NP_TICKET_PARAM_SUBJECT_REGION` to *`paramId`* in `sceNpAuthGetTicketParam()`; then specify the obtained user region (`SceNpTicketParam.data`) to *`raw`* and the pointer to the `SceNpCountryCode` structure to *`cc`*. This will enable you to obtain the country/region code representing the user's area of residence in *`*cc`*.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam, SceNpCountryCode`

SCE_NP_SUBJECT_REGION_GET_LANGUAGE_CODE

Obtain the language code

Definition

```
#include <np_auth.h>
#define SCE_NP_SUBJECT_REGION_GET_LANGUAGE_CODE(raw) \
                                    (*((SceUChar8 *)(raw)+3))
```

Description

This macro parses a ticket parameter and obtains the language code used by the user.

Specify `SCE_NP_TICKET_PARAM_SUBJECT_REGION` to *`paramId`* in `sceNpAuthGetTicketParam()` and specify the obtained user region (`SceNpTicketParam.data`) to *`raw`*. This will enable you to obtain the locale of the display message.

See Also

`sceNpAuthGetTicketParam(),SceNpTicketParam`

Constants

Maximum Values

Constant representing maximum lengths used in ticket related features

Definition

Value	(Number)	Description
SCE_NP_TICKET_SERIAL_ID_SIZE	20	Maximum length of the ticket serial ID
SCE_NP_ONLINEID_MAX_LENGTH	16	Maximum length of the user Online ID
SCE_NP_SUBJECT_REGION_SIZE	4	Maximum length of the user's region
SCE_NP_SUBJECT_DOMAIN_SIZE	4	Maximum length of the user domain
SCE_NP_SERVICE_ID_SIZE	24	Maximum length of the service ID

Return Codes

List of error codes returned by APIs for ticket related features

Definition

Value	(Number)	Description
SCE_NP_AUTH_ERROR_ALREADY_INITIALIZED	0x80550301	Attempted to initialize an already initialized library
SCE_NP_AUTH_ERROR_NOT_INITIALIZED	0x80550302	Attempted to call API in a state where the library has not yet been initialized
SCE_NP_AUTH_ERROR_ENOMEM	0x80550304	There is not enough memory
SCE_NP_AUTH_ERROR_ESRCH	0x80550305	Context or entitlement of the specified ID does not exist
SCE_NP_AUTH_ERROR_EBUSY	0x80550306	Internal thread is busy and the API cannot be called
SCE_NP_AUTH_ERROR_INVALID_SERVICE_ID	0x80550308	Service ID is invalid
SCE_NP_AUTH_ERROR_INVALID_CREDENTIAL	0x80550309	Sign-in ID or password is invalid
SCE_NP_AUTH_ERROR_INVALID_ENTITLEMENT_ID	0x8055030a	Entitlement ID is invalid
SCE_NP_AUTH_ERROR_INVALID_DATA_LENGTH	0x8055030b	Cookie data length is invalid
SCE_NP_AUTH_ERROR_UNSUPPORTED_TICKET_VERSION	0x8055030c	Version of the specified ticket is invalid
SCE_NP_AUTH_ERROR_STACKSIZE_TOO_SHORT	0x8055030d	Stack size of the internal thread is small
SCE_NP_AUTH_ERROR_TICKET_STATUS_CODE_INVALID	0x8055030e	Response from the server was an unexpected invalid value
SCE_NP_AUTH_ERROR_TICKET_PARAM_NOT_FOUND	0x8055030f	Specified parameter does not exist for the ticket

Return Codes Returned by the Authentication Server

List of return codes returned by the authentication server

Definition

Value	(Number)	Description
SCE_NP_AUTH_ERROR_SERVICE_END	0x80550400	PSN℠ services have been permanently terminated
SCE_NP_AUTH_ERROR_SERVICE_DOWN	0x80550401	PSN℠ services are temporarily unavailable
SCE_NP_AUTH_ERROR_SERVICE_BUSY	0x80550402	PSN℠ services are busy
SCE_NP_AUTH_ERROR_SERVER_MAINTENANCE	0x80550403	Server is under maintenance
SCE_NP_AUTH_ERROR_S_INVALID_DATA_LENGTH	0x80550410	Data length is invalid
SCE_NP_AUTH_ERROR_S_INVALID_USER_AGENT	0x80550411	User agent is invalid
SCE_NP_AUTH_ERROR_S_INVALID_VERSION	0x80550412	Version is invalid
SCE_NP_AUTH_ERROR_S_INVALID_SERVICE_ID	0x80550420	Service ID is invalid
SCE_NP_AUTH_ERROR_S_INVALID_CREDENTIAL	0x80550421	Sign-in ID or password is invalid
SCE_NP_AUTH_ERROR_S_INVALID_ENTITLEMENT_ID	0x80550422	Entitlement ID is invalid
SCE_NP_AUTH_ERROR_S_INVALID_CONSUMED_COUNT	0x80550423	Consumption count is invalid
SCE_NP_AUTH_ERROR_INVALID_CONSOLE_ID	0x80550424	Client console ID is invalid
SCE_NP_AUTH_ERROR_CONSOLE_ID_SUSPENDED	0x80550427	Use of the client console is temporarily suspended
SCE_NP_AUTH_ERROR_ACCOUNT_CLOSED	0x80550430	Account has been closed
SCE_NP_AUTH_ERROR_ACCOUNT_SUSPENDED	0x80550431	Account has been temporarily suspended
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_EULA	0x80550432	User must renew EULA
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT1	0x80550440	Account cannot be logged into
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT2	0x80550441	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT3	0x80550442	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT4	0x80550443	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT5	0x80550444	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT6	0x80550445	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT7	0x80550446	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT8	0x80550447	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT9	0x80550448	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT10	0x80550449	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT11	0x8055044a	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT12	0x8055044b	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT13	0x8055044c	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT14	0x8055044d	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT15	0x8055044e	Same as above
SCE_NP_AUTH_ERROR_ACCOUNT_RENEW_ACCOUNT16	0x8055044f	Same as above
SCE_NP_AUTH_ERROR_UNKNOWN	0x80550480	Internal error

Authorization Code Related Features

Obtaining Authorization codes

SceNpAuthOAuthRequestId

Request ID for authorization code obtain processing (OAuth processing)

Definition

```
#include <np_auth_oauth.h>
typedef int SceNpAuthOAuthRequestId;
```

Description

This request ID is issued when a request is created for obtaining an authorization code. By specifying the request ID and calling `sceNpAuthAbortOAuthRequest()`, a currently processing request can be aborted.

Notes

This request ID is for the authorization code related features (APIs provided by np_auth_oauth.h). Do not use it with the ticket related features.

See Also

`sceNpAuthCreateOAuthRequest()`, `sceNpAuthDeleteOAuthRequest()`, `sceNpAuthAbortOAuthRequest()`

SceNpAuthGetAuthorizationCodeParameter

Parameters for obtaining an authorization code

Definition

```
#include <np_auth_oauth.h>
Typedef struct SceNpAuthGetAuthorizationCodeParameter{
        SceSize size;
        const SceNpClientId *pClientId;
        const SceChar8 *pScope;
};
```

Members

size	Size of this structure
pClientId	Client ID assigned to application server
pScope	Scope of information requested by application server

Description

This structure represents the parameters to specify when calling `sceNpAuthGetAuthorizationCode()`.

For *size*, specify the size of this structure.

For *pClientId*, specify the client ID of the application server (client program) that will be attempting access of user information on PSN℠. Specify the client ID issued from a request made in advance.

For *pScope*, specify the scope of the user information that the application server is attempting to obtain from server of PSN℠.

See Also

`sceNpAuthGetAuthorizationCode()`

sceNpAuthCreateOAuthRequest

Create request

Definition

```
#include <np_auth_oauth.h>
SceNpAuthOAuthRequestId sceNpAuthCreateOAuthRequest(
        void
);
```

Arguments

None

Return Values

Returns a request ID for normal termination.

Returns a negative value (error code) upon error.

Description

This function is for creating a request for obtaining an authorization code.

The request ID returned by this function is used when obtaining an authorization code, when deleting a request, etc., so store it in an appropriate variable.

Examples

```
int ret;
SceNpAuthOAuthRequestId reqId;

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

See Also

sceNpAuthDeleteOAuthRequest(), sceNpAuthAbortOAuthRequest()

sceNpAuthDeleteOAuthRequest

Delete request

Definition

```
#include <np_auth_oauth.h>
Int sceNpAuthDeleteOAuthRequest(
        SceNpAuthOAuthRequestId reqId
);
```

Arguments

reqId Request ID

Return Values

Returns SCE_OK (=0) for normal termination.

Returns a negative value (error code) upon error.

Description

This function is for deleting a request to obtain an authorization code.

Examples

```
int ret;
extern SceNpAuthOAuthRequestId reqId;

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

See Also

sceNpAuthCreateOAuthRequest(),sceNpAuthAbortOAuthRequest()

sceNpAuthAbortOAuthRequest

Abort request

Definition

```
#include <np_auth_oauth.h>
int sceNpAuthAbortOAuthRequest(
        SceNpAuthOAuthRequestId reqId
);
```

Arguments

reqId Request ID

Return Values

Returns SCE_OK (=0) for normal termination.

Returns a negative value (error code) upon error.

Description

This function is for aborting the processing to obtain an authorization code.

Examples

```
int ret;
extern SceNpAuthOAuthRequestId reqId;

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

See Also

sceNpAuthCreateOAuthRequest(),sceNpAuthDeleteOAuthRequest()

sceNpAuthGetAuthorizationCode

Get authorization code

Definition

```
#include <np_auth_oauth.h>
int sceNpAuthGetAuthorizationCode(
        SceNpAuthOAuthRequestId reqId,
        const SceNpAuthGetAuthorizationCodeParameter *param,
        SceNpAuthorizationCode *authCode,
        int *issuerId
);
```

Arguments

reqId	Request ID
param	Parameters for obtaining authorization code
authCode	Destination to store obtained authorization code
issuerId	Destination to store obtained issuer ID

Return Values

Returns SCE_OK (=0) for normal termination.

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

Value	(Number)	Description
SCE_NP_AUTH_ERROR_EINVAL	0x80550303	Argument is invalid
SCE_NP_AUTH_ERROR_ABORTED	0x80550307	Processing aborted due to abort API
SCE_NP_AUTH_ERROR_SIGNED_OUT	0x80550310	Signed out

Description

This function obtains an authorization code from the server of PSN℠. This function informs the server of PSN℠ that a user is permitted to access user information on PSN℠ through an application server, then this function obtains the authorization code.

Transfer the obtained authorization code and the issuer ID to the application server. The application server will be able to use these to obtain an access token and access the user information maintained by the server of PSN℠.

This function is a synchronous processing type function and will be blocked until communication completes and the authorization code can be obtained. Once processing completes, call sceNpAuthDeleteOAuthRequest() to delete the used request.

Examples

```
extern SceNpAuthOAuthRequestId reqId;
SceNpAuthGetAuthorizationCodeParameter param;
SceNpAuthorizationCode authCode;
int issuerId;

memset(&param, 0, sizeof(param));
param.size = sizeof(param);  // Structure size
param.pClientId = &clientId; // Client ID assigned to application server
param.pScope = "psn:s2s";    // Scope of information requested by applicaton
server

ret = sceNpAuthGetAuthorizationCode (
        reqId,       // Request ID
        &param,      // Parameters for obtaining authorization code
        &authCode,   // Stores the obtained authorization code
        &issuerId    // Stores the obtained issuer ID
        );
if (ret < 0) {
        // Error handling
}
....
```

Notes

For the argument *reqId*, specify the request ID created by the function `sceNpAuthCreateOAuthRequest()`, which creates requests for obtaining authorization codes. Do not specify a request ID for the ticket related features.

See Also

`sceNpAuthCreateOAuthRequest()`, `sceNpAuthDeleteOAuthRequest()`, `sceNpAuthAbortOAuthRequest()`

Constants

Return Codes

List of return codes returned by the authorization code related features

Definition

Value	(Number)	Description
SCE_NP_AUTH_ERROR_SIGNED_OUT	0x80550310	Signed out

Return Codes Returned by the Authentication Server

Main return codes returned by the authentication server

Definition

In the authorization code related features, error codes where the upper 16 bits start with 0x82e0 or 0x82e1 may return (such as 0x82e01039 and 0x82e101f7). These error codes indicate authentication server errors. Codes that start with 0x82e0 are authentication server errors. Codes that start with 0x82e1 are authentication server HTTP status codes. The main codes from among these error codes are shown in the following.

Authentication Server Errors

Lower 16 bits of authentication server errors that start with 0x82e0

(Hexadecimal)	(Decimal)	Description
0x0014	20	Wrong sign-in ID and/or password
0x001b	27	Account has been disabled
0x001c	28	Account has been banned
0x001d	29	Device has been banned
0x0064	100	Password has been reset
0x0067	103	Re-agreeing to EULA is required
0x1039	4153	Invalid scope (be careful of this during development, it will occur when the wrong scope is specified)
0x1042	4162	Re-agreeing to EULA is required (sub-account)
0x104d	4173	Invalid client (be careful of this during development, it will occur when the wrong client ID is specified)
0x1050	4176	Account has been indefinitely banned

Authentication Server HTTP Status Codes

Lower 16 bits of authentication server HTTP status codes that start with 0x82e1

(Hexadecimal)	(Decimal)	Description
0x019a	410	410 Gone (service is no longer available)
0x01f7	503	503 Service Unavailable (server is undergoing maintenance)

Common Constants

Return Codes

List of common return codes shared by the NP Auth library

Definition

Value	(Number)	Description
SCE_NP_AUTH_ERROR_EINVAL	0x80550303	Argument is invalid
SCE_NP_AUTH_ERROR_ABORTED	0x80550307	Processing aborted due to abort API