NP Auth Library Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Purpose and Characteristics

The NP Auth library is a library for obtaining a ticket associated with a service ID or an authorization code associated with a client ID on the PSN℠ servers.

By using tickets, applications can check if a user has already obtained an entitlement (entitlement type product) distributed from PlayStation®Store, and applications will be able to allow a user to consume a consumable type entitlement.

Authorization codes are used to assign access rights for application servers to user information maintained on PSN℠ servers. The “application servers” mentioned here refers to servers (typically game servers) set up by licensees. By transferring an authorization code to an application server, the application server will be able to access user information maintained by the PSN℠ servers. Note that access rights assigned using authorization codes are assigned according to the OAuth 2.0 framework.

Main Features

The main features provided by the NP Auth library are as follows.

Ticket Related Features

- Feature for specifying the service ID and obtain a ticket
- Feature for extracting an entitlement from the obtained ticket
- Feature for extracting attached information (e.g. issued time) from the obtained ticket
- Feature for consuming (reduce the quantity/frequency of) a consumable entitlement

Authorization Code Related Features

- Feature for obtaining an authorization code by specifying a client ID

Used Resources

The NP Auth library uses the following system resources.

Resource	Description
Work memory	(The application is not required to provide memory.)
Thread	(A thread affecting the application will not be created.)
Processor time	Can be ignored

Embedding into a Program

Ticket Related Features

The following is an explanation of the method for adding the ticket related features (the APIs provided by np_auth.h) when they are to be used in a program.

Include np.h in the source program. Various header files, such as, np_auth.h, will be automatically included as well.

In addition, before calling any NP Auth library APIs in the program, load the PRX module with the relevant libsysmodule API, as follows.

```
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_NP) != SCE_OK ) {
    // Error handling
}
```

When calling an API of the NP Auth library, the modules of libhttp, libssl and libnet must also be loaded and initialized. Regarding the loading of each module, refer to the "NP Library Overview", "libhttp Overview", "libssl Overview" and "libnet Overview" documents.

Upon building the program, link libSceNpCommon_stub.a.

Authorization Code Related Features

The following is an explanation of the method for adding the authorization code related features (the APIs provided by np_auth_oauth.h) when they are to be used in a program.

Include np.h in the source program. Various header files, such as, np_auth_oauth.h, will be automatically included as well. In addition, before calling any NP Auth library APIs in the program, load the PRX module with the relevant libsysmodule API, as follows.

```
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_NP) != SCE_OK ) {
    // Error handling
}
```

When calling an API of the NP Auth library, the modules of libhttp, libssl and libnet must also be loaded and initialized. Regarding the loading of each module, refer to the "NP Library Overview", "libhttp Overview", "libssl Overview" and "libnet Overview" documents.

Upon building the program, link libSceNpManager_stub.a.

Sample Program (Ticket Related Features)

A ticket related features sample program using the NP Auth library is as follows.

sample_code/network/api_np/np_ticket/

This sample obtains a ticket and displays ticket parameters and entitlement data.

Reference Materials

Ticket Related Features

Refer to the following document regarding types of products that can be distributed via PlayStation®Store and their combinations.

- PSN℠ Commerce Service Overview

Authorization Code Related Features

Refer to the following document for an overview of the NP S2S (Server to Server) service and the PSN℠ Web APIs for accessing the PSN℠ servers from application servers.

- PSN℠ Web APIs Overview

Refer to the following document for the usage of the Auth Web APIs which obtain access tokens from application servers.

- Auth Web APIs Reference

Refer to the following for OAuth 2.0.

- RFC6749 The OAuth 2.0 Authorization Framework
  http://tools.ietf.org/html/rfc6749

(The above reference destinations have been confirmed as of February 13, 2015. Note that pages may have been subsequently moved or the contents modified.)

2 Using the Ticket Related Features

Basic Procedure: Check Entitlement

The following is an explanation of the processing that, as a basic procedure for the processing used by the NP Auth library, obtains a ticket from the PSN℠ servers and checks if a user has a specific entitlement.

(1) Initialize the Network Libraries

Initialize libhttp, libssl, and libnet. For the initialization procedure, refer to library document of each library.

(2) Initialize the NP Auth Library

Call `sceNpAuthInit()` to initialize the NP Auth library.

(3) Start Ticket Request Processing

Prepare the `SceNpAuthRequestParameter` structure: set the size of the structure, the version of the ticket you want to obtain, the service ID assigned to the title, and the pointer to the ticket obtainment callback function (described later) (it is also possible to additionally set a cookie for the title to manage sessions on its own, as well as to set parameters to provide to the ticket obtainment callback function).

```
#define SCE_NP_AUTH_LATEST_TICKET_VERSION_MAJOR     3
#define SCE_NP_AUTH_LATEST_TICKET_VERSION_MINOR     0
#define AUTH_SERVICE_ID "IV0002-NPXS00004_00"

SceNpAuthRequestParameter param;
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;
```

Call `sceNpAuthCreateStartRequest()` with the pointer to the above structure specified as the function's argument.

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

(4) Callback Check

Once the call of `sceNpAuthCreateStartRequest()` reaches normal termination, an internal thread of the NP Auth library sends a request to the server of PSN℠. The application must continue to carry out appropriate processing, and also periodically call `sceNpCheckCallback()` while waiting for a response to be gained from the server of PSN℠ and for the ticket obtainment callback function to be called. For `sceNpCheckCallback()`, refer to the "NP Library Reference" document.

(5) Obtain Ticket Data

When the ticket obtainment callback function is called, first check the value passed to *`result`*. A negative value is an error code, indicating that a ticket could not be obtained from the server of PSN℠. If the value is not negative, *`result`* indicates the size of the ticket data. Allocate a buffer of the required size and call `sceNpAuthGetTicket()` to obtain the ticket data.

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

// Ticket obtainment callback function
int
authCallback(SceNpAuthRequestId id, int result, void *arg)
{
    int ret = 0;
    (void)arg;
    // When a ticket cannot be obtained from the server of PSN℠,
    // a negative value will be passed to result
    if (result < 0) {
        auth_result = result;
        return 0;
    }
    // Otherwise, the ticket size will be passed to result
    ticket_len = result;
    // Allocate buffer
    ticket = (unsigned char *)malloc(ticket_len);
    if (ticket == NULL) {
        auth_result = -1;
        return 0;
    }
    // Obtain ticket data
    auth_result = sceNpAuthGetTicket(id, ticket, ticket_len);
    return 0;
}
```

Note
The validity periods of tickets for service IDs is currently set to 24 hours. For external servers that provide game services, always perform validity period checks for tickets for service IDs and the applicable processing.

(6) Check Entitlement

The ticket data obtained with `sceNpAuthGetTicket()` lists the service entitlements for the target service ID. To check whether a specific entitlement is included or not, use `sceNpAuthCheckEntitlementById()`.

```
#define AUTH_ENT_ID "IV0002-NPXS00004_00-SVC001"

ret = sceNpAuthCheckEntitlementById(ticket, ticket_len, entId);
if (ret == 0){
    // Processing when the user owns the specific entitlement
} else {
    // Processing when the user does not own the specific entitlement
}
```

Note
An entitlement with an expiration time set will no longer be included in the ticket data once its expiration time passes. In other words, entitlements included in the ticket data are all within their validity period. In addition, consumable entitlements will no longer be included in a ticket when *`remaining_count`* reaches 0.

Note
Although an entitlement check is possible within the callback function, it is recommended that the check be made after returning from the callback function.

(7) Delete Request

When communication processing completes, call `sceNpAuthDestroyRequest()` to delete the request. Since this processing is a counterpart of `sceNpAuthCreateStartRequest()`, the processing must be carried out even if communication processing is aborted.

(8) Terminate the Library

When you no longer need the NP Auth library, call `sceNpAuthTerm()` to terminate it. This will free the resources allocated upon initialization.

Aborting Processing

To abort processing after calling `sceNpAuthCreateStartRequest()` because of a timeout or because of the user's cancel operation, call `sceNpAuthAbortRequest()`. The ticket obtainment callback function will be called with `SCE_NP_AUTH_ERROR_ABORTED` passed to *`result`*; this will indicate that the processing has been aborted. After this confirmation, call `sceNpAuthDestroyRequest()` to delete the request.

Consuming a Consumable Entitlement

A consumable entitlement is used to realize an entitlement where the quantity or frequency by which that entitlement can be used is set to a certain number (for example, bullets). The usable quantity/frequency is managed by the server of PSN℠. When the user attempts to use the privilege, the application must specify the consumption amount to the server of PSN℠ and request a ticket. If sufficient quantity/frequency remains, the server of PSN℠ will issue a ticket (with the number of quantity/frequency remaining after subtracting the amount requested by the application).

In order to make this a practical processing flow from the user's point of view, it is recommended that a ticket first be obtained without specifying the amount of consumption. By doing so, the quantity/frequency of an entitlement that is still available to the owner can be obtained in advance, and the application can be designed so that the user is only allowed to use his/her privilege when the user still has enough quantity/frequency remaining. This procedure is explained below.

(1) Obtain Number of Usable Quantity/Frequency

In the same manner as the procedure explained in the section "Basic Procedure: Check Entitlement", set parameters to the `SceNpAuthRequestParameter` structure and call `sceNpAuthCreateStartRequest()`.

```
#define SCE_NP_AUTH_LATEST_TICKET_VERSION_MAJOR     3
#define SCE_NP_AUTH_LATEST_TICKET_VERSION_MINOR     0
#define AUTH_SERVICE_ID "IV0002-NPXS00004_00"

SceNpAuthRequestParameter param;
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;

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

When the ticket obtainment callback function is called, check the value passed to *`result`*. Allocate a buffer if there is no error occurrence and call `sceNpAuthGetTicket()` to obtain the ticket data.

Next, use `sceNpAuthGetEntitlementById()` and obtain the target consumable entitlement from the ticket data.

```
#define AUTH_ENT_ID "IV0002-NPXS00004_00-SVC001"
SceNpEntitlement ent;

ret = sceNpAuthGetEntitlementById(ticket, ticket_len, AUTH_ENT_ID, &ent);
if (ret < 0){
    // Error handling
}
```

The *`remaining_count`* member of the obtained `SceNpEntitlement` structure represents the usable quantity/frequency of the entitlement. If there is no longer enough entitlement quantity/frequency left, gray-out the feature to use this entitlement or prompt the user to make more purchase of entitlements.

(2) Consume an Entitlement

Prepare the `SceNpAuthRequestParameter` structure: set the size of the structure, the version of the ticket you want to obtain, the service ID assigned to the title, the pointer to the ticket obtainment callback function, as well as the ID of the entitlement to be consumed and its quantity/frequency. (It is also possible to additionally set a cookie for the title to manage sessions on its own, and to set parameters to provide to the ticket obtainment callback function.)

```
#define AUTH_ENT_CONSUME_COUNT 1

SceNpAuthRequestParameter param;
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.entitlementId = AUTH_ENT_ID;
param.consumeCount = AUTH_ENT_CONSUME_COUNT;
```

Use the pointer to this structure as an argument and call `sceNpAuthCreateStartRequest()`.

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

When the call reaches normal termination, carry out appropriate processing and also poll `sceNpCheckCallback()` while waiting for the ticket obtainment callback function to be called.

(3) Obtain the Remaining Usable Quantity/Frequency after Consumption

When the ticket obtainment callback function is called, first check the value passed to *`result`*. A negative value is an error code, indicating that a ticket could not be obtained from the server of PSN℠ (although there should always be enough usable quantity/frequency left in the procedure explained here, note that `SCE_NP_AUTH_ERROR_S_INVALID_CONSUMED_COUNT` will be passed if there is not enough usable quantity/frequency left of the entitlement).

If the value is not negative, *`result`* indicates the size of the ticket data. Allocate a buffer of the required size and call `sceNpAuthGetTicket()` to obtain the ticket data. Then, by obtaining the target consumable entitlement from the ticket data with `sceNpAuthGetEntitlementById()`, you can obtain the quantity/frequency that still remains usable for the entitlement.

Obtaining Information Attached to a Ticket

Call `sceNpAuthGetTicketParam()` for the ticket data that was obtained using `sceNpAuthGetTicket()` to obtain information attached to that ticket – such as, the ID of the server that issued the ticket, the issued time and expiration time of the ticket, the account ID of the user who requested the ticket, etc. This attached information can be utilized to perform a strict check on the authenticity of the ticket - for example, to set up measures against "replay attacks" (preventing unauthorized user action, such as, repeatedly using the same obtained ticket via the proxy server in game play).

3 Usage of Authorization Code Related Features

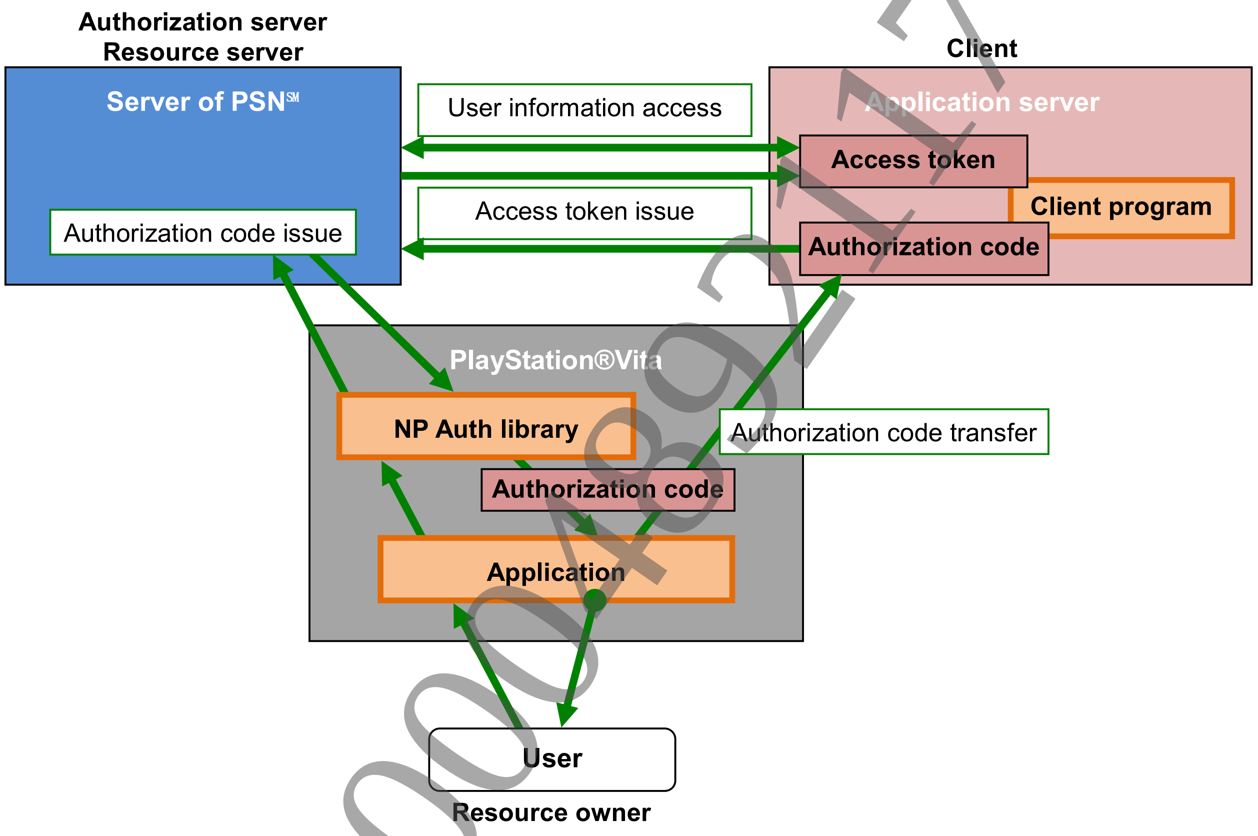
Overview

PSN℠ supports OAuth 2.0 in order to assign access rights to application servers.

Through OAuth 2.0, application servers set up by game developers will be able to access user information on PSN℠ servers and will be able to provide unique services in connection with PSN℠.

The following is a general diagram for assigning access rights to application servers with PSN℠.

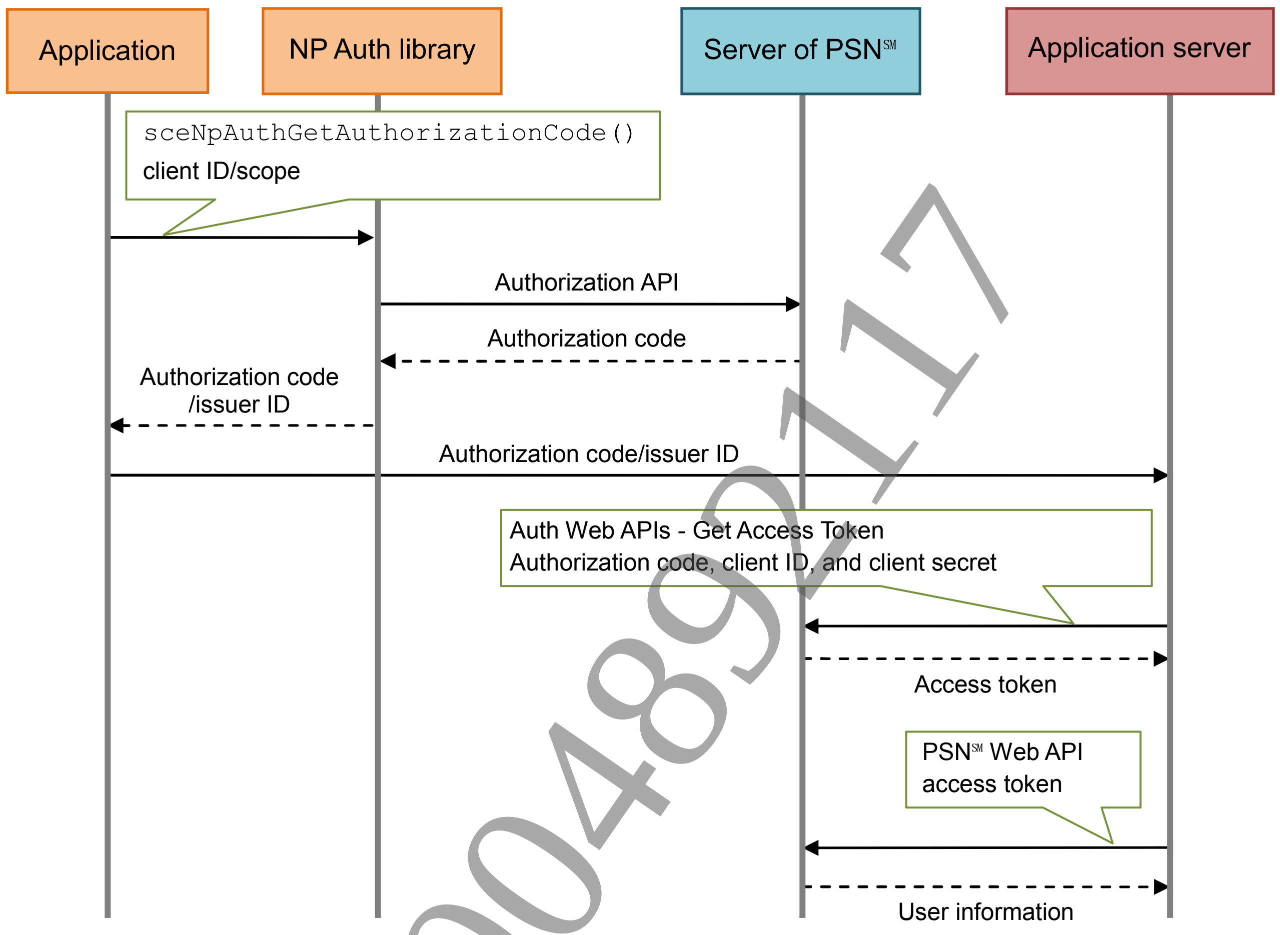
Figure 1 Assigning Access Rights to Application Servers



Process Flow for Assigning Access Rights

The following shows the process flow for assigning access rights to an application server.

Figure 2 Process Flow for Assigning Access Rights


Application
NP Auth library
Server of PSN℠
Application server
sceNpAuthGetAuthorizationCode()
client ID/scope
Authorization API
Authorization code
Authorization code /issuer ID
Authorization code/issuer ID
Auth Web APIs - Get Access Token
Authorization code, client ID, and client secret
Access token
PSN℠ Web API access token
User information

Applications on PlayStation®Vita

Applications on PlayStation®Vita use the NP Auth library to obtain authorization codes from server of PSN℠. Authorization codes are temporary data that has short validity periods, therefore it should be transferred to application servers immediately. Issuer IDs, which can be similarly obtained when obtaining authorization codes from the NP Auth library, should also be transferred to application servers.

The NP Auth Library

The NP Auth library is a library that performs communication for obtaining authorization codes from PSN℠ servers. The NP Auth library provides synchronous processing APIs that perform communication with PSN℠ servers and obtain authorization codes. Applications should set the following information related to application servers to obtain authorization codes.

- Client ID: Specify the value issued from the PlayStation®Vita Developer Network (https://psvita.scedev.net/)
- Scope: The scope indicating the access scope for the user information to access

Application Servers

Application servers (client programs) use the Auth Web APIs to obtain access tokens from authorization codes. In addition to authorization codes, the following information is required in order to obtain access tokens.

- Client ID and corresponding client secret: Specify the values issued from the PlayStation®Vita Developer Network (https://psvita.scedev.net/) for both
- Redirect URI: Specify the value issued by the PlayStation®Vita Developer Network

After obtaining an access token, the PSN℠ Web APIs can be used to access user information.

Note that application servers must differentiate between PSN℠ server environments using the issuer ID transferred from applications on PlayStation®Vita.

Server of PSN℠

Server of PSN℠ provide functionality (issuing authorization codes and access tokens) as authorization servers with OAuth 2.0 and provide resource server functionality (management of user information).

Basic Procedure: Obtaining Authorization Codes

The following is an explanation of the procedure for obtaining authorization codes by performing communication with server of PSN℠.

(1) Initialize the Network Libraries

Initialize libhttp, libssl, libnet, and the NP library. Refer to the respective library documents for the initialization procedures.

Note
Unlike the ticket related features, the NP library must be initialized for the authorization code related features. On the other hand, `sceNpAuthInit()` does not need to be called.

(2) Create Requests

Create synchronous processing requests. Create a request for each communication process and delete the requests after the communication processes are complete.

```
int ret;
SceNpAuthOAuthRequestId reqId;

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

(3) Execute requests

Call `sceNpAuthGetAuthorizationCode()` to obtain the authorization code from the server of PSN℠. This function performs blocking until the authorization code is obtained.

```
// Execute request
SceNpAuthGetAuthorizationCodeParameter param;
SceNpAuthorizationCode authCode;
int issuerId;

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

(4) Delete Requests

After a request completes, specify the request ID and delete the request.

```
// Delete request
sceNpAuthDeleteOAuthRequest (reqId);
```

(5) Transfer the Authorization Code and Issuer ID to the Application Server

Transfer the authorization code and issuer ID obtained through the NP Auth library to the application server.

(6) Termination

When the authorization code related features (and the NP library features) are no longer needed, call `sceNpTerm()` to perform termination processing.