NP IN-GAME Commerce 2 Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Document serial number: 000004892117

1 Overview

Purpose and Characteristics

The NP IN-GAME Commerce 2 library provides browsing functions and enables products to be purchased from PlayStation®Store (Title Store). Users can access the Title Store from the application and purchase data of additional items and scenarios, as well as access rights to the game server.

The NP IN-GAME Commerce 2 library provides an in-game browsing feature as a method for purchasing products. In the in-game browsing method, the application handles the browsing, and the system software handles the purchasing. In other words, the application uses NP IN-GAME Commerce 2 library APIs to obtain product information from the server of PSN℠, and then presents the information to the user (for browsing). If the user selects a product for purchasing, the application uses the Store Checkout Dialog to invoke the necessary processing on the system software for purchases. The purchase processing is executed in the user interface provided by the system software.

Note
In addition to the above, there is a method to purchase products by calling the "Title Store application" feature embedded in the system software. In this method, the application specifies the category ID or the product ID of the browse/purchase target, and the browsing/purchase processing is carried out by a user interface provided by the system software. For details, refer to the "Application Utility Overview" document.

This document does not provide information of creating and preparing the products for the Title Store. Relevant information can be found in the documents listed in the section "Reference Materials".

When the user purchases additional items and other products, it is necessary for application behavior to be modified accordingly. The NP IN-GAME Commerce 2 library itself does not enable the application to check whether or not a product has been purchased, but the NP Auth library and NP TCM library can be used as described in this document to determine if service entitlements (subscription-type products) have been purchased.

Embedding into a Program

Include np.h in the source program. In addition to the np/np_commerce2.h, in which APIs of the NP IN-GAME Commerce 2 library are declared, several other header files will also be automatically included.

Load also the PRX module in the program as follows.

```
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_HTTPS) != SCE_OK ) {
    // Error handling
}
if ( sceSysmoduleLoadModule(SCE_SYSMODULE_NP_COMMERCE2) != SCE_OK ) {
    // Error handling
}
```

Upon building the program, link libSceNpCommerce2_stub.a.

Sample Programs

The following sample programs that use the NP IN-GAME Commerce 2 library are provided for your reference.

sample_code/network/api_np/np_gui_commerce2/

This sample exemplifies basic usage of the NP IN-GAME Commerce 2 library.

sample_code/network/api_np/np_ticket/

This sample obtains tickets, and obtains/checks entitlements.

Reference Materials

Detailed information of PlayStation®Store, and the specifications of products distributed through PlayStation®Store can be found in the following documents.

- "PSN℠ Commerce Service Overview"
- "PSN℠ Commerce Programming Guide"

Information of making products available in PlayStation®Store can be found in the following document.

- "NP Product Management Guide"

Information of the libraries to check if service entitlements have been purchased can be found in the following documents.

- "NP Auth Library Overview", "NP Auth Library Reference"
- "NP TCM Overview", "NP TCM (Linux C and Windows DLL Versions) Reference"

2 Purchase Processing (In-game Browsing)

In this chapter, the procedure for using the NP IN-GAME Commerce 2 library to obtain a product catalog and to display products in an application, have the user select a product, and then have the Store Checkout Dialog process the purchase (in-game browsing) will be described.

Preparation

(1) Load the PRX Module

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_HTTPS` and `SCE_SYSMODULE_NP_COMMERCE2` specified as the module IDs to load the PRX.

(2) Initialization

Initialize the dependent libraries in the following order.

(1) libnet [initialize with `sceNetInit()`]
(2) libnetctl [initialize with `sceNetCtlInit()`]
(3) libssl [initialize with `sceSslInit()`]
(4) libhttp [initialize with `sceHttpInit()`]
(5) NP library [initialize with `sceNpInit()`]

Then call `sceNpCommerce2Init()` to initialize the NP IN-GAME Commerce 2 library. When initialization is successful, `sceNpCommerce2Init()` returns 0.

(3) Create a Commerce 2 Context

Call `sceNpCommerce2CreateCtx()` to create a commerce 2 context. As arguments, specify the callback function to receive event notifications and a variable to receive the ID of the created context.

When `sceNpCommerce2CreateCtx()` succeeds in creating a context, it stores the context ID to the specified variable and returns 0.

Create a Commerce 2 Session

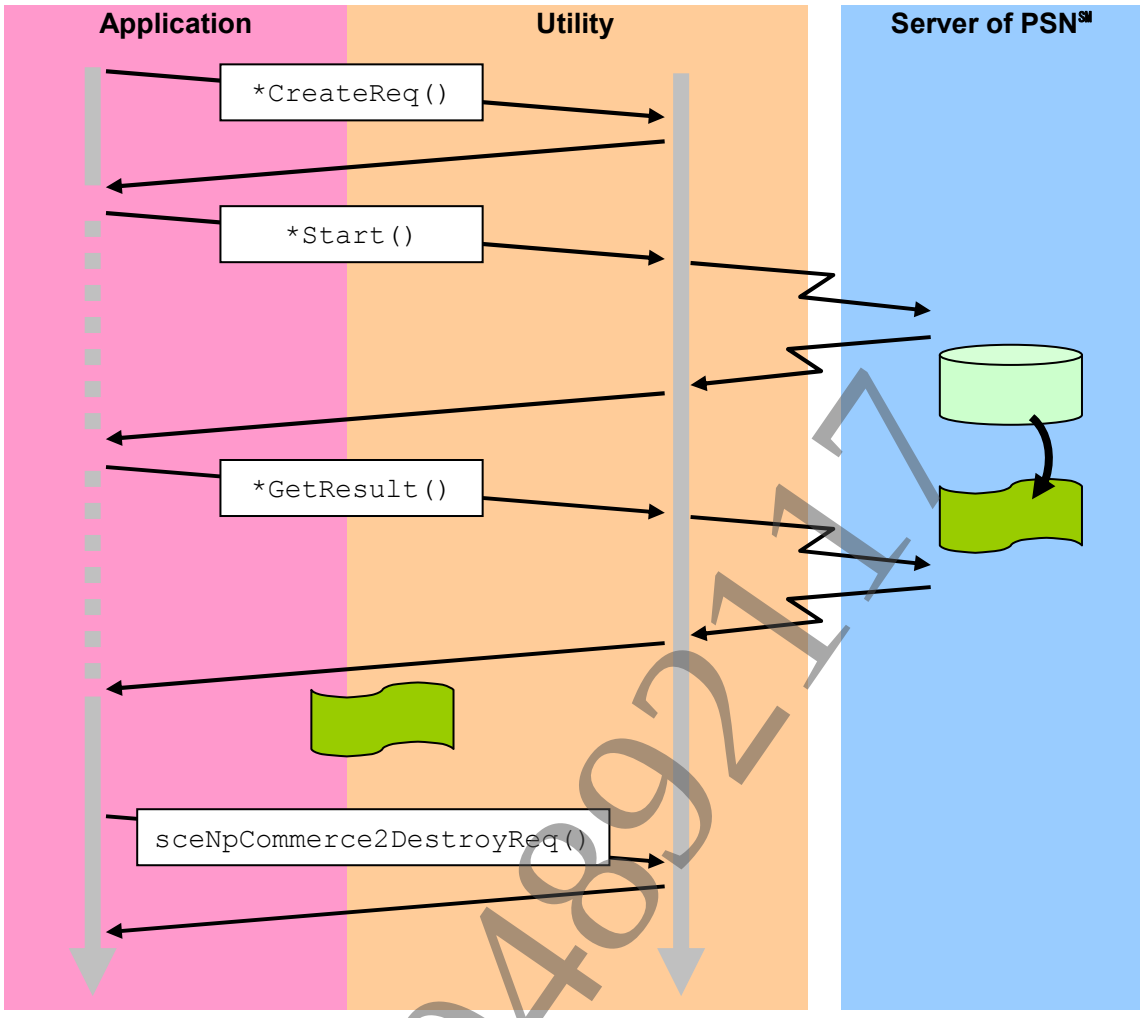
Create a commerce 2 session from the server of the PSN℠ side.

To create a commerce 2 session, first call `sceNpCommerce2CreateSessionCreateReq()` to create a request. Next, call `sceNpCommerce2CreateSessionStart()`.

Call `sceNpCommerce2CreateSessionGetResult()` with the buffer for receiving the data specified.

Lastly, call `sceNpCommerce2DestroyReq()` to delete the request.

Figure 1 Create a Commerce 2 Session



* The prefix `sceNpCommerce2CreateSession` has been omitted from function names.

Note that `sceNpCommerce2CreateSessionStart()` and `sceNpCommerce2CreateSessionGetResult()` do not execute processing in the background but instead execute communications in the calling thread. For this reason, these functions are blocking until communication completes. To abort the communication, call `sceNpCommerce2AbortReq()`.

Obtaining the Product Catalog

The product catalog is organized into hierarchical categories, and the information is obtained as either category content information or product information. Also, because the data structure is undisclosed, this data can only be obtained in two steps. First, data is obtained from the server, and then this data is parsed so that information can be taken out.

(1) Obtain Category Content Data

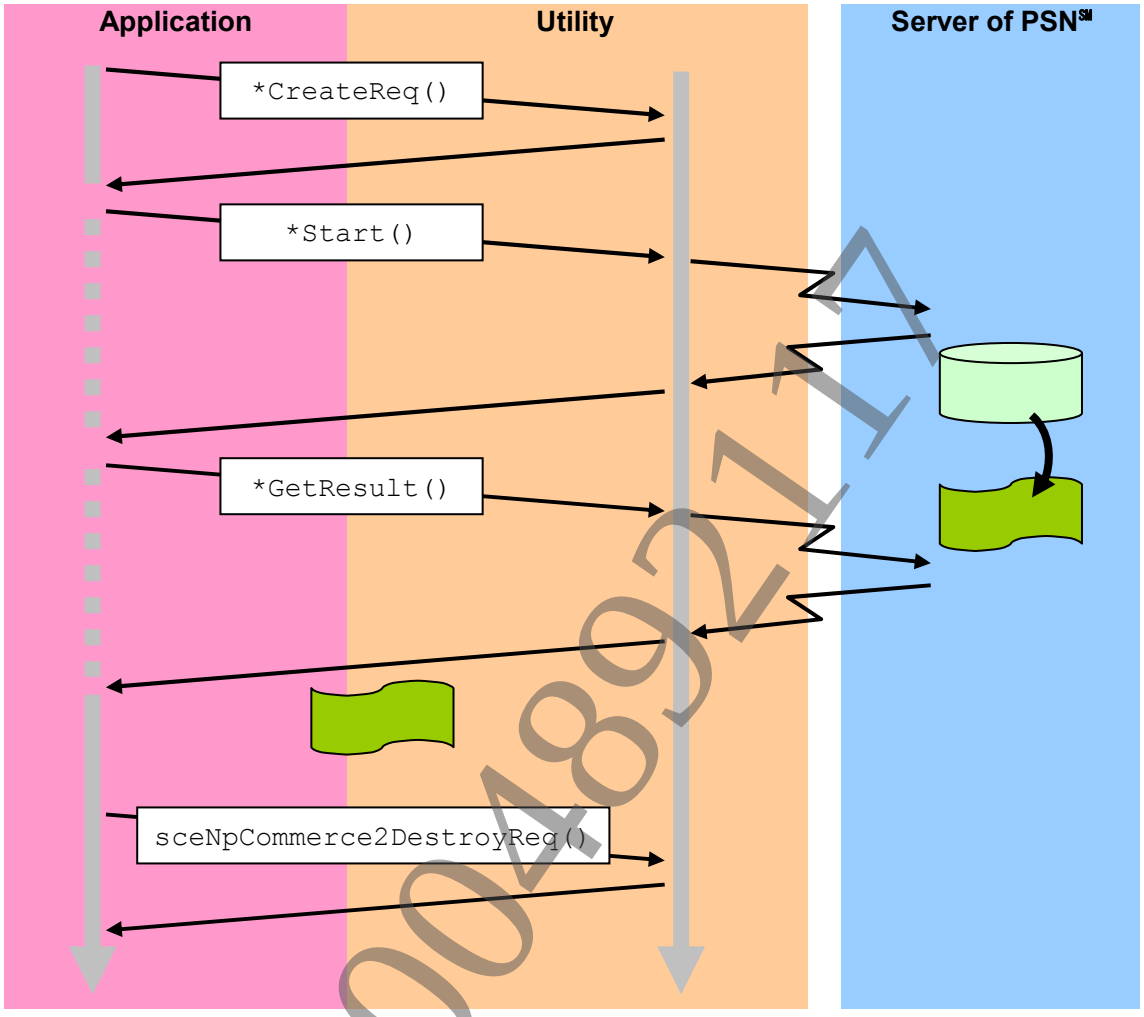
To obtain category content data, first call `sceNpCommerce2GetCategoryContentsCreateReq()` to create a request. Then call `sceNpCommerce2GetCategoryContentsStart()`, specifying the ID of the target category and the required range of content.

> **Note**
> The category ID of the main category is the same as the service ID. For example, if the service ID is IV0002-NPXS00004_00, the category ID of the main category is also IV0002-NPXS00004_00.

Then call `sceNpCommerce2GetCategoryContentsGetResult()`, specifying the buffer to receive the data.

Finally, call `sceNpCommerce2DestroyReq()` to delete the request.

Figure 2 Obtain Category Content Data



* The prefix `sceNpCommerce2GetCategoryContents` has been omitted from function names.

Note that `sceNpCommerce2GetCategoryContentsStart()` and `sceNpCommerce2GetCategoryContentsGetResult()` do not execute processing in the background but instead execute communications in the calling thread. For this reason, these functions are blocking until communication completes. To abort the communication, call `sceNpCommerce2AbortReq()`.

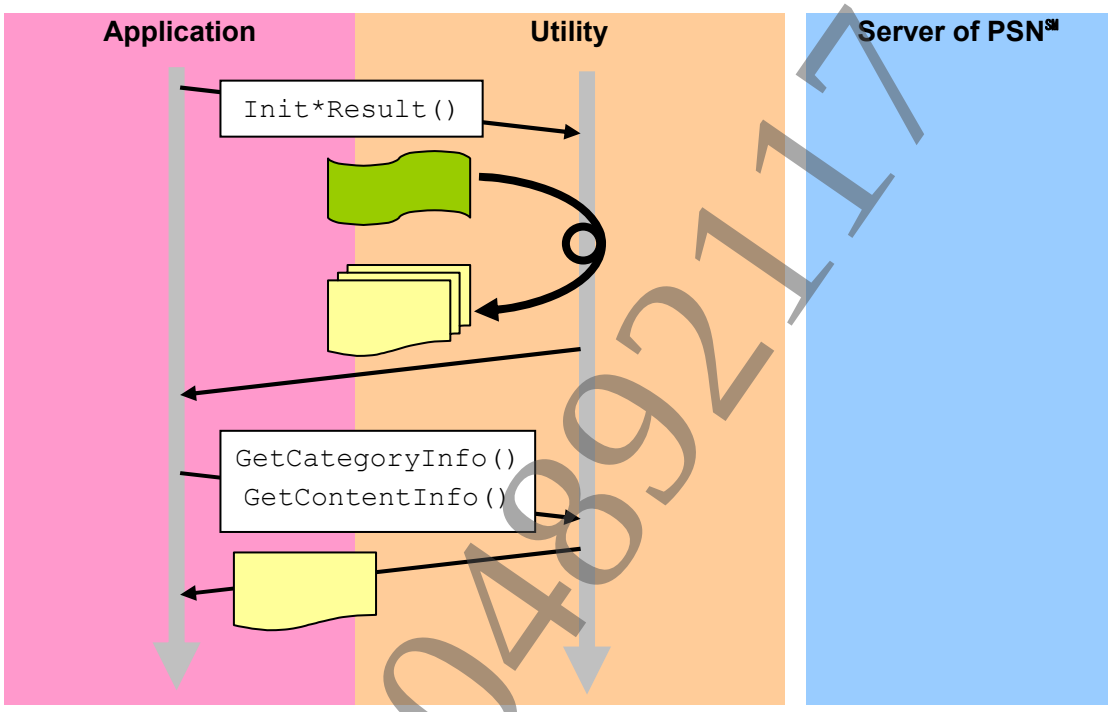
(2) Take Out Category Content Information

The data structure of the category content data obtained with `sceNpCommerce2GetCategoryContentsGetResult()` is undisclosed. The information contained is taken out as necessary by calling the appropriate function.

First call `sceNpCommerce2InitGetCategoryContentsResult()` to initialize the category content data obtained.

This enables category content information to be taken out with functions like `sceNpCommerce2GetCategoryInfo()` and `sceNpCommerce2GetContentInfo()`.

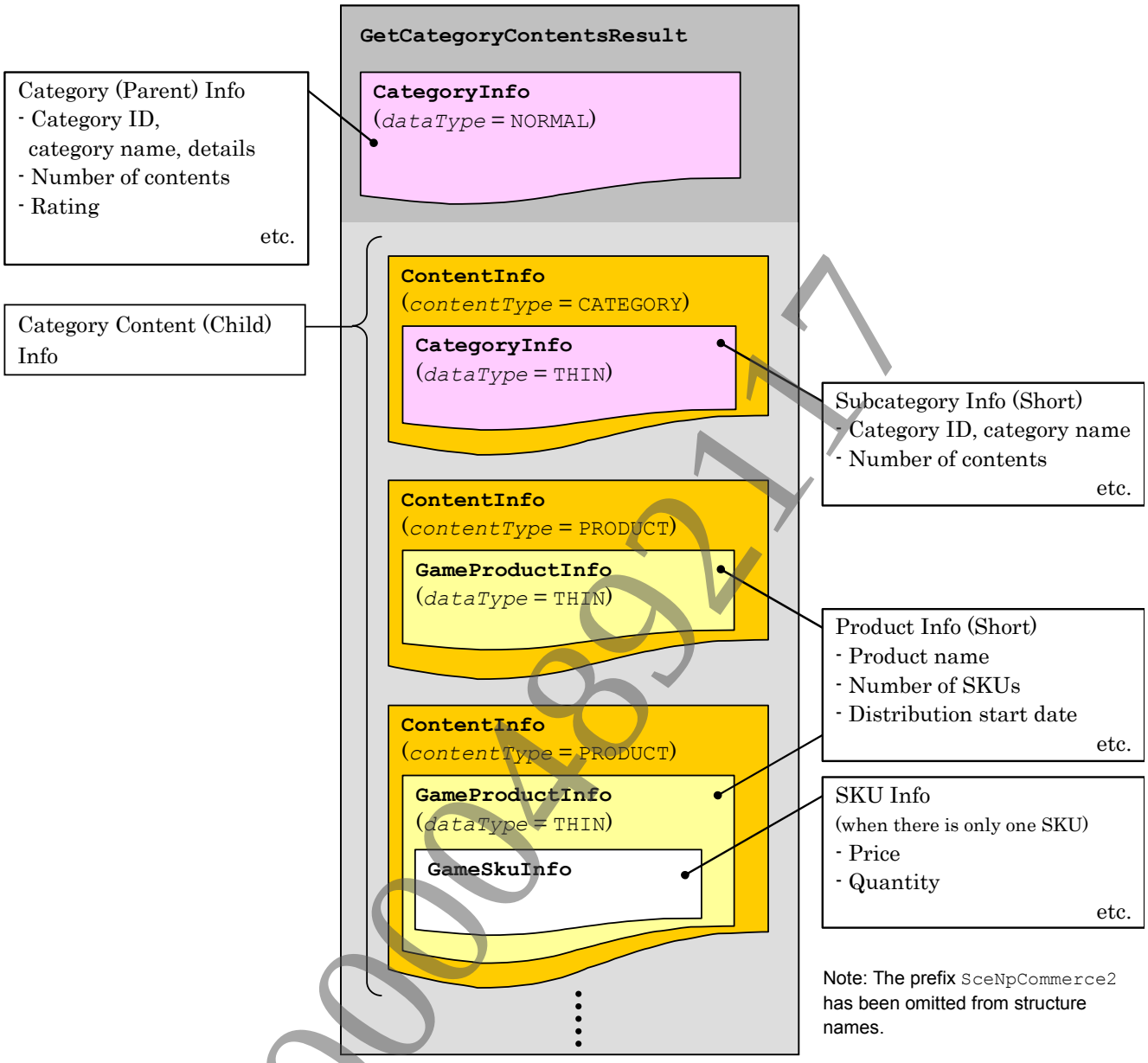
Figure 3 Take Out Category Content Information



The prefix `sceNpCommerce2` has been omitted from function names.
The asterisk (*) replaces "`GetCategoryContents`".

Category content information is in the following format.

Figure 4 Category Content Information Structure



Category content information of subcategories can be obtained to reproduce the hierarchical structure of the product catalog. Specify the category IDs of subcategories as arguments.

The product information that can be obtained from category content information is a shortened version. To obtain the complete information of a product, obtain product data first and take out product information from this data.

Note
Depending on conditions on the user-side, such as, Parental Controls settings, a product may not be provided to the user even if it is registered to the PlayStation®Store. Assume the possibility of having a situation where the number of contents/subcategories is 0 within a category, and program an appropriate processing for such situations. An API is also provided to detect whether there are any distributed items that can be provided to the user; thus, it is possible to design your program so that the above check is made before category content information is obtained.

(3) Obtain Product Data

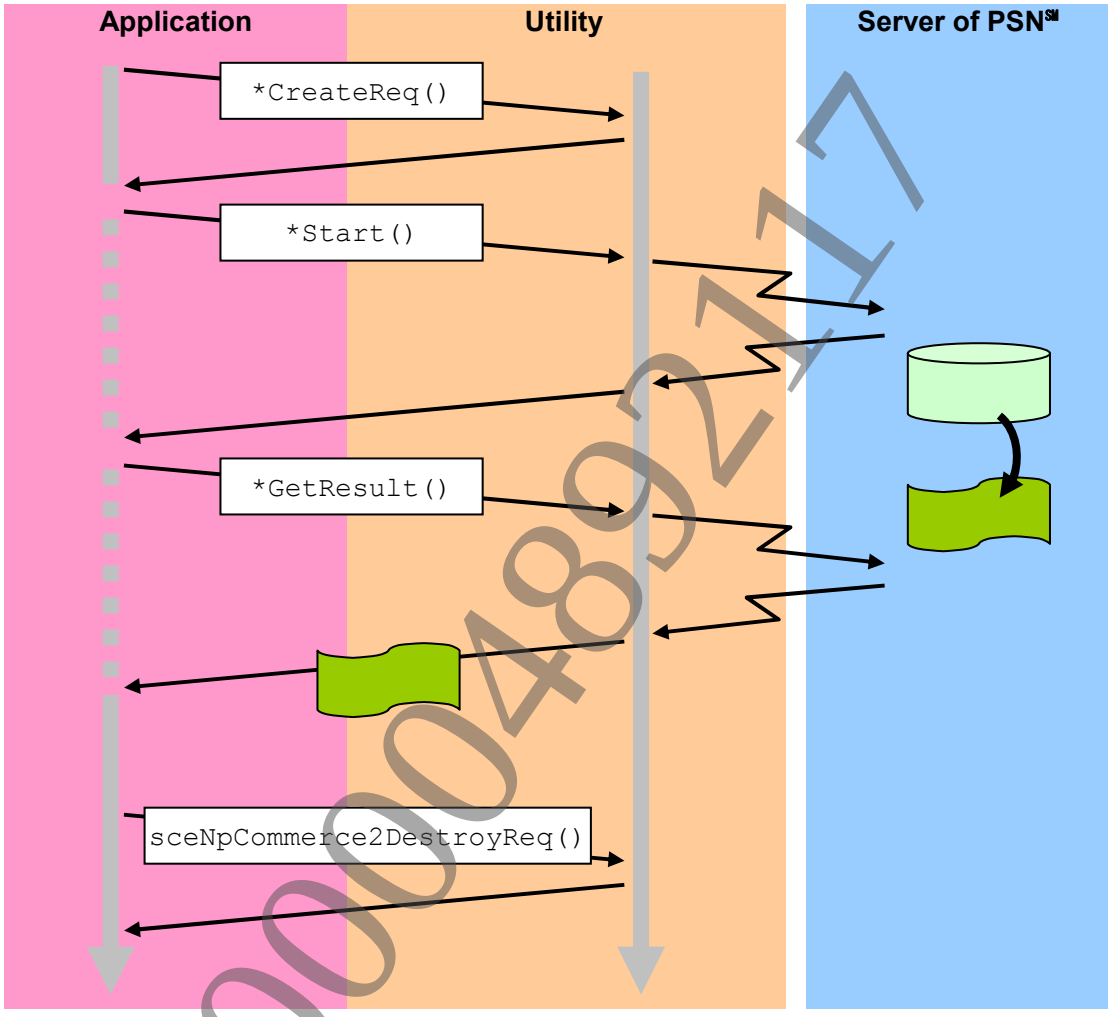
The procedure for obtaining product data is much like the procedure for obtaining category content data.

First call `sceNpCommerce2GetProductInfoCreateReq()` to create a request. Then call `sceNpCommerce2GetProductInfoStart()` with the product ID specified.

Then call `sceNpCommerce2GetProductInfoGetResult()`, specifying the buffer to receive the data.

Finally, call `sceNpCommerce2DestroyReq()` to delete the request.

Figure 5 Obtain Product Data



* The prefix `sceNpCommerce2GetProductInfo` has been omitted from function names.

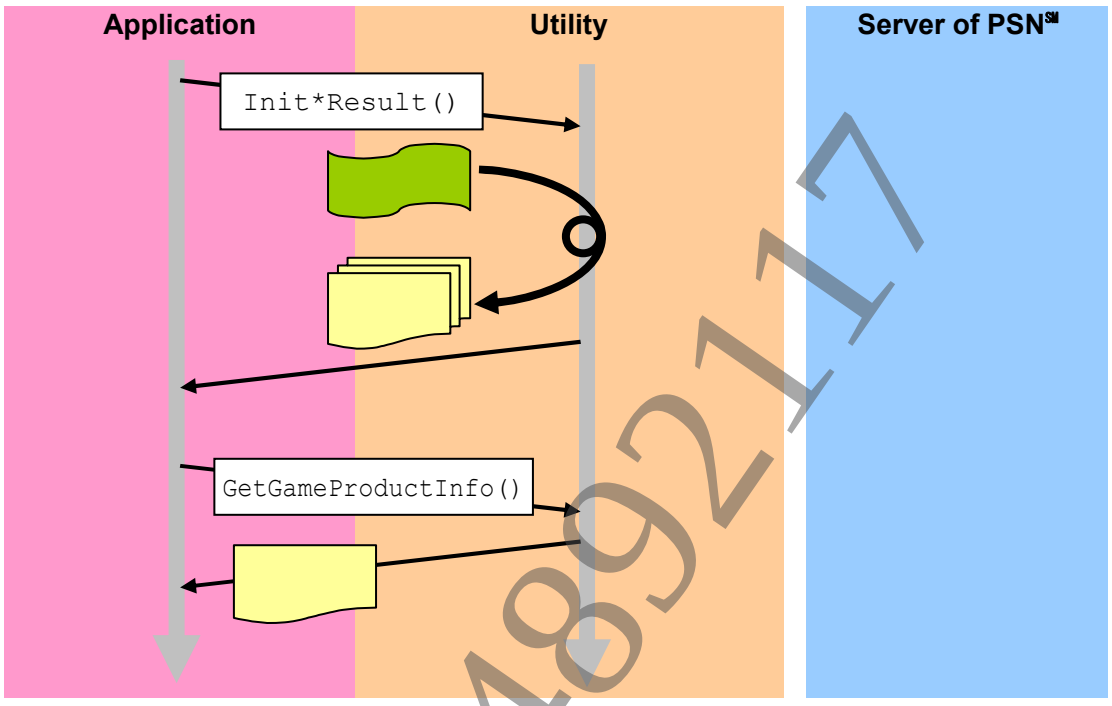
Note that `sceNpCommerce2GetProductInfoStart()` and `sceNpCommerce2GetProductInfoGetResult()` do not execute processing in the background but instead execute communications in the calling thread. For this reason, these functions are blocking until communication completes. To abort the communication, call `sceNpCommerce2AbortReq()`.

(4) Take Out Game Product Information

The data structure of the product data obtained with `sceNpCommerce2GetProductInfoGetResult()` is undisclosed. The information contained is taken out as necessary by calling the appropriate function.

First call `sceNpCommerce2InitGetProductInfoResult()` to initialize the data obtained. This enables game product information to be taken out with `sceNpCommerce2GetGameProductInfo()`.

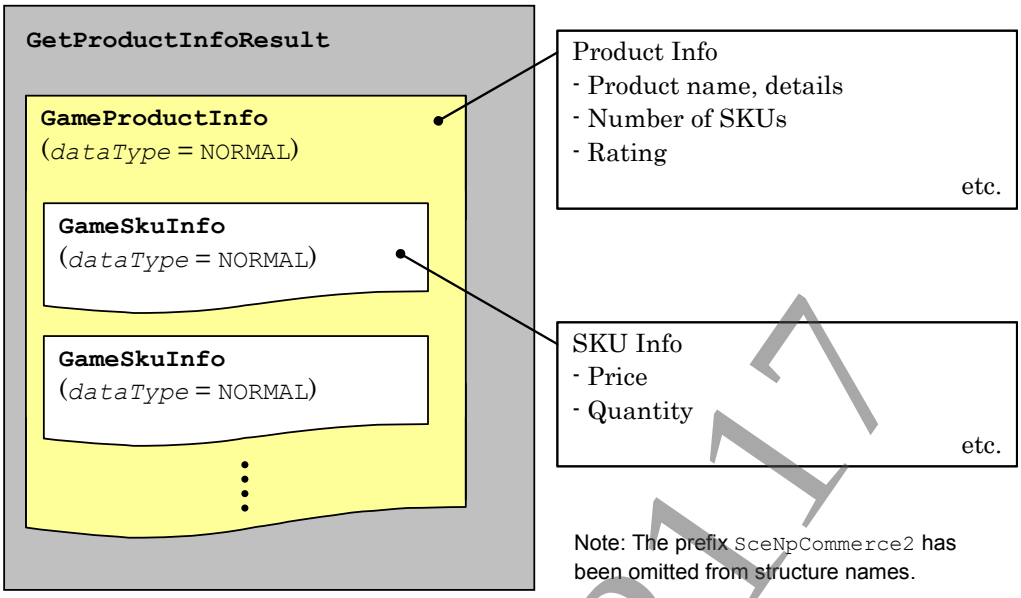
Figure 6 Take Out Game Product Information



The prefix `sceNpCommerce2` has been omitted from function names.
The asterisk (*) replaces "`GetProductInfo`".

Game product information is in the following format.

Figure 7 Game Product Information Structure

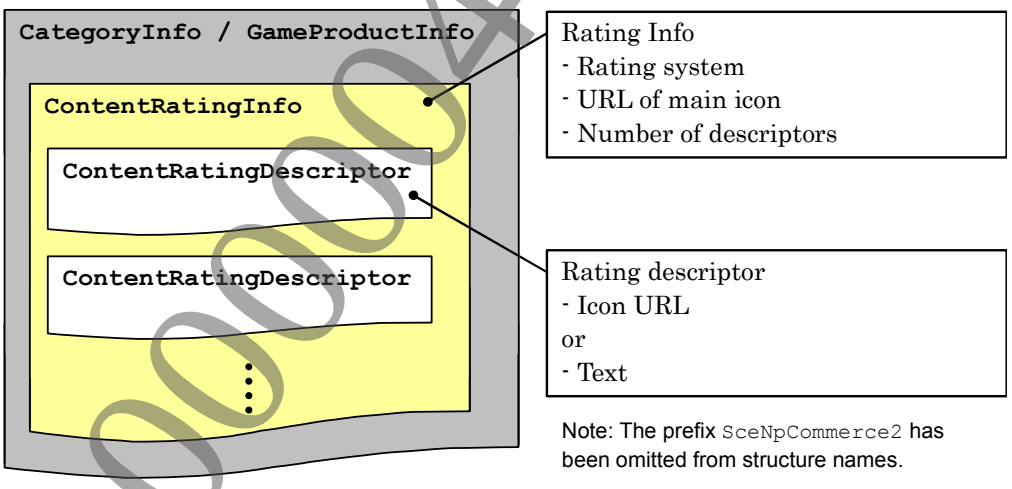


To take out SKU information from game product information, use `sceNpCommerce2GetGameSkuInfoFromGameProductInfo()`.

(5) Take Out Rating Information

Rating information is available for both categories and products, and is in the following format.

Figure 8 Rating Information Structure



Note
It is not necessary normally for the application to handle rating information. The system displays the rating and blocks inappropriate content depending on the user's age.

Obtaining Product Information from Multiple Categories

If the products are all under one category, information of multiple products can be obtained with just one request for category content information. To obtain the information of products under different categories, however, it is necessary to carry out the request for each category, or if the product IDs are known, it is possible to obtain a product info list. This method can be used to obtain the necessary product information with just one request, if a list of product IDs can be obtained from the game server (for example).

(1) Obtain a Product Info List

To obtain the product information of products in multiple categories, obtain a "product info list" from the server and then take out the game product information from the list.

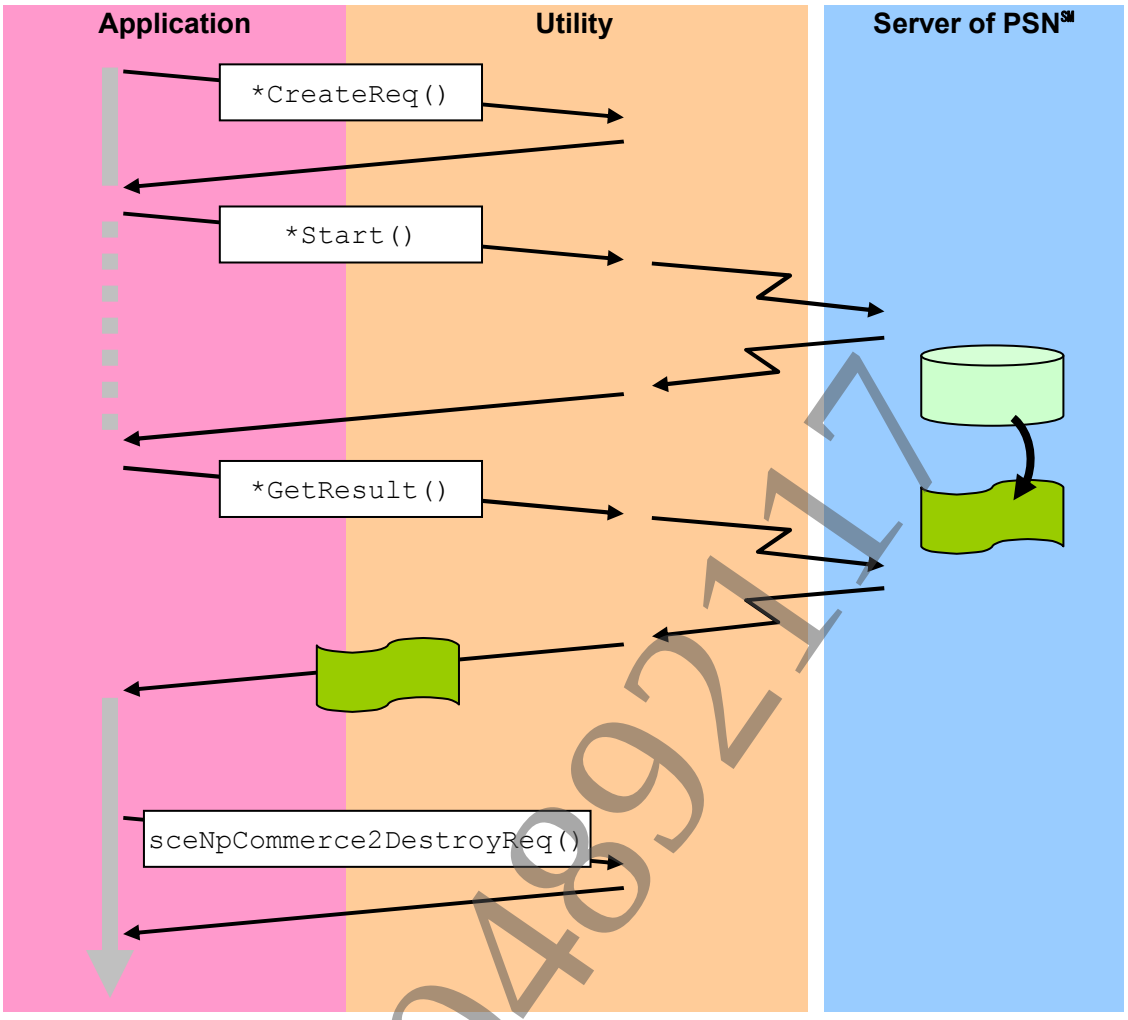
The procedure for obtaining a product info list is much like the procedure for obtaining category content data.

First, call `sceNpCommerce2GetProductInfoListCreateReq()` to create a request. Then call `sceNpCommerce2GetProductInfoListStart()`, specifying an array of the target product IDs.

Next, call `sceNpCommerce2GetProductInfoListGetResult()`, specifying the buffer to receive the data.

Finally, call `sceNpCommerce2DestroyReq()` to delete the request.

Figure 9 Obtain a Product Info List



* The prefix `sceNpCommerce2GetProductInfoList` has been omitted from function names.

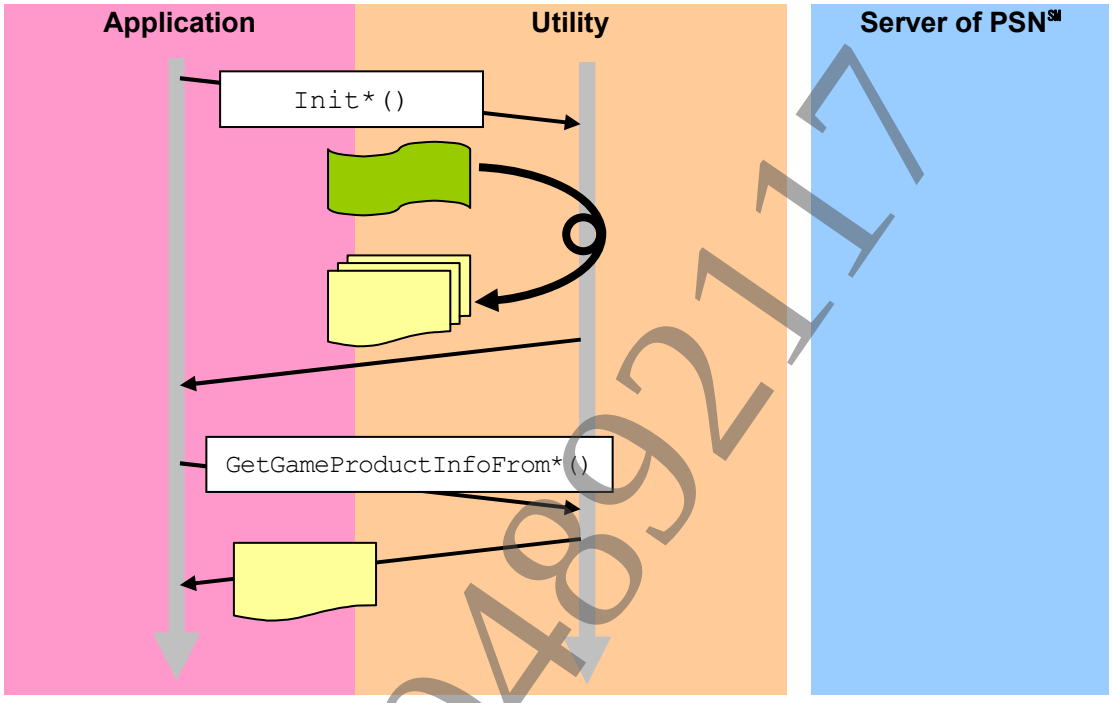
Note that `sceNpCommerce2GetProductInfoListStart()` and `sceNpCommerce2GetProductInfoListGetResult()` do not execute processing in the background but instead execute communications on the calling thread. For this reason, these functions are blocking until communication completes. To abort the communication, call `sceNpCommerce2AbortReq()`.

(2) Take Out Game Product Information

The data structure of the product info list obtained with `sceNpCommerce2GetProductInfoListGetResult()` is undisclosed. The information contained is taken out as necessary by calling the appropriate function.

First call `sceNpCommerce2InitGetProductInfoListResult()` to initialize the product info list that was obtained. This enables game product information to be taken out with the function `sceNpCommerce2GetGameProductInfoFromGetProductInfoListResult()`.

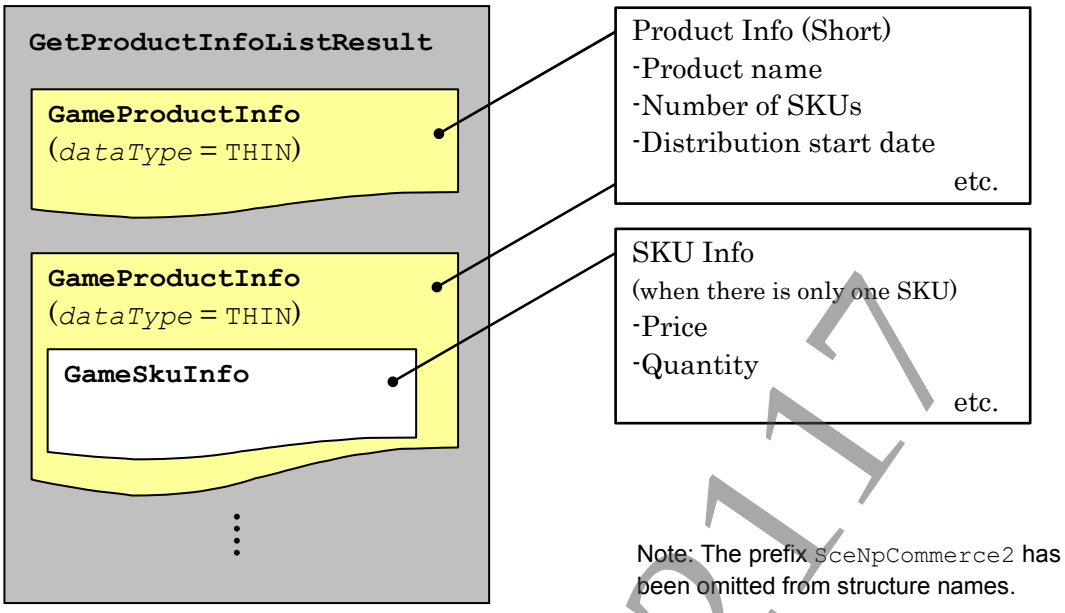
Figure 10 Take Out Game Product Information



The prefix `sceNpCommerce2` has been omitted from function names.
The asterisk (*) replaces "`GetProductInfoListResult`".

The game product information included in product info lists is in the following format.

Figure 11 Product Info List Structure



Note
Depending on conditions on the user-side, such as, Parental Controls settings, a product may not be provided to the user even if it is registered to the PlayStation®Store. Assume the possibility of the number of SKUs being 0, and program an appropriate processing for such situations. An API is also provided to detect whether there are any distributed items that can be provided to the user; thus, it is possible to design your program so that the above check is made before game product information is obtained.

To obtain the SKU information included in the game product information, call `sceNpCommerce2GetGameSkuInfoFromGameProductInfo()`.

Browsing

After obtaining and taking out the category content information and product information, display this information in the application and enable the user to choose the products to purchase. It is possible to implement a shopping cart method that enables the user to purchase multiple products at once. Refer to the sample program for an example of a shopping cart implementation.

Checkout (Purchase Processing)

Once the user has chosen the product to purchase, use the Store Checkout Dialog to carry out the purchase processing. For details, refer to the "Store Checkout Dialog Overview" and the "Store Checkout Dialog Reference" documents.

Termination

To terminate the use of the NP IN-GAME Commerce 2 library, first call `sceNpCommerce2DestroyCtx()` to destroy the commerce 2 context. When successful, `sceNpCommerce2DestroyCtx()` returns 0.

Call `sceNpCommerce2Term()` and `sceNpTerm()` to terminate the NP IN-GAME Commerce 2 library and the NP library. Perform termination processing of the libhttp with `sceHttpTerm()`, libssl with `sceSslTerm()`, and the network libraries with `sceNetCtlTerm()` and `sceNetTerm()`.

Then, unload PRX by calling `sceSysmoduleUnloadModule()` with `SCE_SYSMODULE_NP_COMMERCE2` and `SCE_SYSMODULE_HTTPS` specified for the module IDs.

List of Functions Used in In-game Browsing

Initialize and Terminate the Library

Function	Description
`sceNpCommerce2Init()`	Initializes the NP IN-GAME Commerce 2 library
`sceNpCommerce2Term()`	Terminates the NP IN-GAME Commerce 2 library

Create and Delete Contexts

Function	Description
`sceNpCommerce2CreateCtx()`	Creates a commerce 2 context
`sceNpCommerce2DestroyCtx()`	Destroys a commerce 2 context
`sceNpCommerce2GetShortfallOfLibhttpPool()`	Obtains the size of the libhttp's memory pool that was lacking for this context during the previous error
`sceNpCommerce2GetShortfallOfLibsslPool()`	Obtains the size of the libssl's memory pool that was lacking for this context during the previous error

Create and Finish Sessions

Function	Description
`sceNpCommerce2CreateSessionCreateReq()`	Creates a request to obtain a commerce 2 session
`sceNpCommerce2CreateSessionStart()`	Starts creating a commerce 2 session
`sceNpCommerce2CreateSessionGetResult()`	Obtains the result of a commerce 2 session creation
`sceNpCommerce2GetSessionInfo()`	Obtains information on a commerce 2 session

Obtain Category Content Data

Function	Description
`sceNpCommerce2GetCategoryContentsCreateReq()`	Creates a request to obtain category content data
`sceNpCommerce2GetCategoryContentsStart()`	Starts obtaining category content data
`sceNpCommerce2GetCategoryContentsGetResult()`	Obtains category content data

Take Out Category Content Information

Function	Description
sceNpCommerce2InitGetCategoryContentsResult()	Initializes category content data
sceNpCommerce2GetCategoryInfo()	Takes out category information
sceNpCommerce2GetContentInfo()	Takes out information of content in the category
sceNpCommerce2GetCategoryInfoFromContentInfo()	Takes out subcategory information
sceNpCommerce2GetGameProductInfoFromContentInfo()	Takes out product information
sceNpCommerce2DestroyGetCategoryContentsResult()	Destroys category content data

Obtain Product Data

Function	Description
sceNpCommerce2GetProductInfoCreateReq()	Creates a request to obtain product data
sceNpCommerce2GetProductInfoStart()	Starts obtaining product data
sceNpCommerce2GetProductInfoGetResult()	Obtains product data

Take Out Product Information

Function	Description
sceNpCommerce2InitGetProductInfoResult()	Initializes product data
sceNpCommerce2GetGameProductInfo()	Takes out product information
sceNpCommerce2DestroyGetProductInfoResult()	Destroys product information

Take Out Rating Information

Function	Description
sceNpCommerce2GetContentRatingInfoFromGameProductInfo()	Takes out rating information of a product
sceNpCommerce2GetContentRatingInfoFromCategoryInfo()	Takes out rating information of a category
sceNpCommerce2GetContentRatingDescriptor()	Takes out rating descriptors

Take Out SKU Information

Function	Description
sceNpCommerce2GetGameSkuInfoFromGameProductInfo()	Takes out SKU information
sceNpCommerce2GetPrice()	Obtains the price string

Obtain a Product Info List

Function	Description
sceNpCommerce2GetProductInfoListCreateReq()	Creates a request to obtain a product info list
sceNpCommerce2GetProductInfoListStart()	Starts obtaining the product info list
sceNpCommerce2GetProductInfoListGetResult()	Obtains the product info list

Take Out Product Information from a Product Info List

Function	Description
sceNpCommerce2InitGetProductInfoListResult()	Initializes the product info list
sceNpCommerce2GetGameProductInfoFromGetProductInfoListResult()	Takes out product information
sceNpCommerce2DestroyGetProductInfoListResult()	Destroys the product info list

Abort and Destroy Requests

Function	Description
`sceNpCommerce2DestroyReq()`	Destroys commerce 2 request
`sceNpCommerce2AbortReq()`	Aborts commerce 2 request

Show and Hide the PlayStation®Store Icon

Function	Description
`sceNpCommerce2ShowPsStoreIcon()`	Shows PlayStation®Store Icon
`sceNpCommerce2HidePsStoreIcon()`	Hides PlayStation®Store Icon

Display of PlayStation®Store Icon During Browsing

When displaying the PlayStation®Store products or categories using the in-game browsing feature, make sure to display PlayStation®Store Icon in accordance with the display condition of TRC [R3121]. The display position can be chosen from among three kinds of the positions: bottom left, bottom center, or bottom right of the screen. Each display position and the size of the icon are as follows.

Figure 12 PlayStation®Store Icon Size

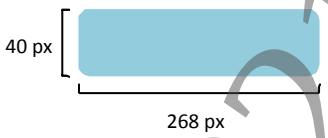
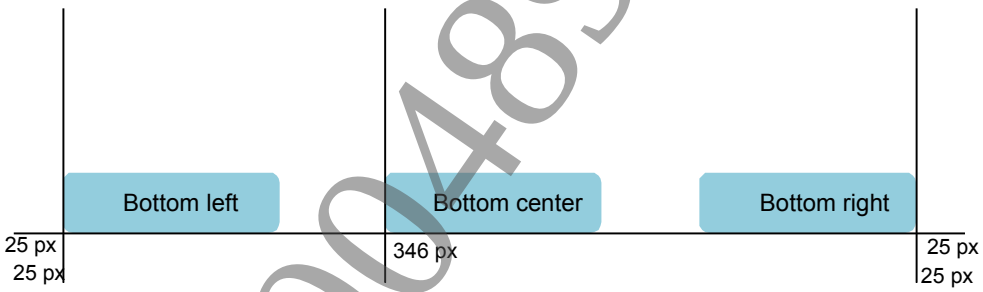


Figure 13 Display Position of PlayStation®Store Icon



PlayStation®Store Icon will be displayed on the screen during the period from the call of `sceNpCommerce2ShowPsStoreIcon()` until the call of `sceNpCommerce2HidePsStoreIcon()`. Note that the layer on which PlayStation®Store Icon is displayed is upper than the layer on which the application's screen is displayed; therefore, make sure that the icon does not hide an important display of the application.

Also, when Store Checkout Dialog or other Common Dialogs is displayed, PlayStation®Store Icon overlaps such dialogs. In this case, call `sceNpCommerce2HidePsStoreIcon()` to hide the icon as necessary.

3 Checking for Purchased Products

This chapter outlines the procedure for checking if service entitlements (subscription-type products) have been purchased.

Checking for Service Entitlements in the Application

The NP Auth library ticketing APIs are used to check for entitlements in the client application. The procedure is as follows.

(1) Load the PRX Module

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP` specified as the module ID to load the PRX.

(2) Initialization

Initialize the dependent libraries in the following order.

(1) libnet [initialize with `sceNetInit()`]
(2) libnetctl [initialize with `sceNetCtlInit()`]
(3) libssl [initialize with `sceSslInit()`]
(4) libhttp [initialize with `sceHttpInit()`]
(5) NP library [initialize with `sceNpInit()`]

Then call `sceNpAuthInit()` to initialize the NP Auth library. When initialization is successful, `sceNpAuthInit()` returns 0.

(3) Obtain a Ticket

Call `sceNpAuthCreateStartRequest()` to issue a request for a ticket, specifying the service ID of the ticket as an argument. When the request starts successfully, `sceNpAuthCreateStartRequest()` returns 0. When the ticket obtainment processing completes, the callback function specified upon calling `sceNpAuthCreateStartRequest()` is called. If the processing is successful, obtain the ticket by calling `sceNpAuthGetTicket()`. For details on ticket obtainment, refer to the "NP Auth Library Overview" and the "NP Auth Library Reference" documents.

(4) Check the Subscription Entitlement

The validity of the entitlement must be checked before it can be used. Call `sceNpAuthCheckEntitlementById()` with the entitlement ID specified as an argument. If the entitlement is valid (the user owns the specific entitlement), `sceNpAuthCheckEntitlementById()` returns 0. If the entitlement is no longer valid, tickets cannot be obtained with the entitlement, and `sceNpAuthCheckEntitlementById()` returns an error.

Entitlements are seen to be valid for a certain grace period past the entitlement's expiration date to enable automatic updates of subscription services. Although the exact expiration time of an entitlement can be obtained with `sceNpAuthCheckEntitlementById()`, do not employ a rigid evaluation of the validity based on this expiration time.

(5) Obtain the Number of Usable Times

To obtain the number of times a consumable entitlement can be used, call `sceNpAuthGetEntitlementById()` with the entitlement ID specified as an argument. The remaining number of times is stored to the *`remainingCount`* member of the `SceNpEntitlement` structure, and the cumulative number of times used to *`consumedCount`*.

(6) Consume Consumable Entitlement

To consume a number of usable times left in a consumable entitlement, call `sceNpAuthCreateStartRequest()` in the same manner as when requesting to obtain a ticket. Upon this function call, specify the entitlement ID and the number of times to consume as arguments. When the consumption is successful, the same callback function as when a ticket was obtained in Step (3) will be called, and a ticket with the consumed entitlement is obtained.

(7) Termination

Call `sceNpAuthTerm()` and `sceNpTerm()` to terminate the NP Auth library and the NP library. Perform termination processing of the libhttp using `sceHttpTerm()`, the libssl using `sceSslTerm()`, and the network libraries using `sceNetCtlTerm()` and `sceNetTerm()`.

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

Checking for Service Entitlements on the Server

To check entitlements on the game server, first obtain a ticket in the client application with the ticketing API of the NP Auth library. Send the ticket to the game server, and then check the ticket using the NP Ticket Checker Module (TCM) on the server. The procedure is as follows.

(1) Load the PRX Module

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP` specified as the module ID to load the PRX.

(2) Initialization

Initialize the dependent libraries in the following order.

(1) libnet [initialize with `sceNetInit()`]
(2) libnetctl [initialize with `sceNetCtlInit()`]
(3) libssl [initialize with `sceSslInit()`]
(4) libhttp [initialize with `sceHttpInit()`]
(5) NP library [initialize with `sceNpInit()`]

Then call `sceNpAuthInit()` to initialize the NP Auth library. When initialization is successful, `sceNpAuthInit()` returns 0.

(3) Obtain a Ticket

Call `sceNpAuthCreateStartRequest()` to issue a request for a ticket. Upon this function call, specify the service ID of the ticket and the ticket version to obtain, etc., as arguments. The ticket version to specify must be the ticket version supported by TCM (to be installed to the server).

When the request starts successfully, `sceNpAuthCreateStartRequest()` returns 0. When the ticket obtainment processing completes, the callback function specified upon calling `sceNpAuthCreateStartRequest()` is called. When the processing is successful, obtain the ticket with `sceNpAuthGetTicket()`. For details on ticket obtainment, refer to the "NP Auth Library Overview" and the "NP Auth Library Reference" documents.

(4) Send Ticket to Server

Call `sceNpAuthGetTicket()` to copy the ticket to the memory allocated by the application. Send the obtained ticket data to the game server via the network.

(5) Check Entitlement with TCM

On the server, use TCM to verify the entitlement of the ticket.

(6) Termination

Call `sceNpAuthTerm()` and `sceNpTerm()` to terminate the NP Auth library and the NP library. Perform termination processing of the libhttp using `sceHttpTerm()`, the libssl using `sceSslTerm()`, and the network libraries using `sceNetCtlTerm()` and `sceNetTerm()`.

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

4 Notes

General Notes

Attribute regarding a Purchased Product That Can Be Obtained as SKU Information

In the NP IN-GAME Commerce 2 library, SKU information can be obtained as the `SceNpCommerce2GameSkuInfo` structure when obtaining game product information.

The `SceNpCommerce2GameSkuInfo` structure includes the *`annotation`* member as information accompanying the SKU; this field is a bit mask.

The bit mask of *`annotation`* has the following flags defined to indicate whether the SKU has already been purchased.

Macro	Value	Description
`SCE_NP_COMMERCE2_SKU_ANN_PURCHASED_CANNOT_PURCHASE_AGAIN`	0x80000000	Already purchased, and cannot be purchased again
`SCE_NP_COMMERCE2_SKU_ANN_PURCHASED_CAN_PURCHASE_AGAIN`	0x40000000	Already purchased, and can be purchased again

Note that this information regarding a purchased product is purely information of the SKU level – it does not indicate whether a service entitlement has been purchased. To perform this check for a service entitlement, follow the procedure described in the chapter "Checking for Purchased Products".

Obtaining Product Information and SKU Information

To use various features of the NP IN-GAME Commerce 2 library for correctly obtaining information of a desired product, the settings of the NPMT (Network Platform Management Tool) must be properly made in advance. For details, please refer to the document "NP Product Management Guide".

Notes on Application Processing

libhttp/libssl Memory Pool Size

The NP IN-GAME Commerce 2 Library internally uses the libhttp and the libssl. Because of this, upon each API call, which performs some form of communication, the memory pool size of the libhttp/libssl is checked to see if there is enough available space required for the particular processing.

The required available space on the libhttp's memory pool is defined by the `SCE_NP_COMMERCE2_LEAST_HTTP_POOL_SIZE` macro as 36KiB.

The required available space on the libssl's memory pool is defined by the `SCE_NP_COMMERCE2_LEAST_SSL_POOL_SIZE` macro as 96KiB.

If the required space is not available, an error (`SCE_NP_COMMERCE2_ERROR_HTTP_POOL_TOO_SHORT` or `SCE_NP_COMMERCE2_ERROR_SSL_POOL_TOO_SHORT`) returns. When this error returns, call either `sceNpCommerce2GetShortfallOfLibhttpPool()` or `sceNpCommerce2GetShortfallOfLibsslPool()` to check how much space is lacking.

Relationship with the libhttp

The NP IN-GAME Commerce 2 library internally uses the libhttp to communicate with the server. Because of this, when using in-game browsing, an libhttp error code may be returned in an API of the NP IN-GAME Commerce 2 library that entails network processing.

Duration of the Commerce 2 Session

When using the in-game browser, the application must first create a commerce 2 session. The created commerce 2 session will subsequently be used in obtaining category content data and product data.

Note that a created commerce 2 session will be deleted on the server-side when a specific amount of inactivity time passes. When the session is deleted from the server-side, the `SCE_NP_COMMERCE2_SERVER_ERROR_SESSION_EXPIRED` error will return for any subsequent communication attempt. To resume communication from this state, delete the commerce 2 context and recreate it.

Currently, the session time-out time set on the server–side is 15 minutes, however, this value may be changed in the future. Do not program your application in such a way that it is dependent on this time. In addition, server load may be negatively affected when communication is periodically generated to prevent a session time-out; please do not implement this type of processing in your application.