near Dialog Utility Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

Document serial number: 000004892117

1 "near" Dialog Utility Overview

Scope of This Document

This document describes the "near" Dialog utility, which allows the transfer of various types of data between game programs and the "near" application and makes it possible to use some of the features of "near" application with the dialog format.

Purpose and Features

The "near" Dialog utility is a library that has a feature to transfer various types of data between game programs and the "near" application and makes it possible to use some of the features of the "near" application with the dialog format.

By using the "near" Dialog utility, a game program supports the distribution of gift data among users that use the "near" game service as well as the use of various data such as the travel distance of users measured with the "near" application, the cumulative number of discovered games, the cumulative number of discovered items, and the cumulative number of discovered nearby users.

The "near" Dialog utility has features equivalent to those of the "near" utility in terms of features for transferring various types of data to/from the "near" application.

The dialog feature of the "near" Dialog utility is implemented by using some of the features in the Common Dialog library within the "near" Dialog utility. The dialog feature usage method is the same as that for other Common Dialogs, so refer to the "Common Dialog Overview" and "Common Dialog Reference" documents.

Main Features

The following are the main features provided by the "near" Dialog utility.

Features for transferring data to/from the "near" application ("near" utility feature)

- Setting gifts to be distributed
- Checking the gifts that have been set for distribution
- Deleting the gifts that have been set for distribution
- Obtaining lists of discovered gifts
- Obtaining information of discovered gifts
- Opening/reading/closing icon image files of discovered gifts
- Opening/reading/closing data files of received gifts
- Deleting discovered gifts
- Setting discovered gifts to the "Ignored" state
- Starting up the "near" application and prompting the user to communicate with the "near" server
- Getting a list of nearby users
- Converting "near" gift event parameters
- Obtaining usage status of the "near" application
- Re-obtaining the information of the "near" application

Features to use some of the functions of the "near" application with the dialog format ("near" Dialog Feature)

- Updating "near" with the dialog format
- Receiving discovered gifts with the dialog format

Used Resources

The "near" Dialog utility uses the following system resources.

Resource	Description
Footprint	36 KiB when PRX is loaded
Work memory	Uses the memory passed at the time of initialization plus an additional 1KiB
Thread	Used. The stack size of a thread is 4 KiB.
Processor time	With functions for starting up external processes, process startup and inter-process communication require some time. (Refer to the "Notes" chapter.)

Embedding into Program

Include near_dialog_util.h in the source program. Various header files will be automatically included as well.

Load the PRX module in the program, as follows.

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

Unload loaded modules when they are no longer necessary.

Upon building the program, link libSceNearDialogUtil_stub.a.

Use of the "near" Dialog utility requires that the NP library be initialized.

For the NP library, refer to the "NP Library Overview" document.

Sample Program

The following file is provided as a sample program that uses the "near" Dialog utility.

sample_code/network/api_near_dialog_util/basic/

Reference Materials

For the "near" game service, refer to the following document.

- near System Overview

For more information on development and debugging methods for "near" compliant game programs, refer to the following document.

- near Compliant Application Development Process Overview

For Common Dialog, refer to the following document.

- Common Dialog Overview
- Common Dialog Reference

2 Usage Procedure of the "near" Utility Feature

"near" Utility Feature

This chapter describes how to use the function for transferring various types of data to/from the "near" application ("near" utility feature). The functions related to calling this feature have the `sceNearUtility*` prefix.

The basic processing flow for the following "near" utility features is described.

- Setting the Gifts to be Distributed
- Checking the Gifts to be Distributed
- Deleting Distribution of a Gift
- Obtaining Discovered Gifts
- Using Received Gifts
- Deleting Discovered Gifts
- Setting Discovered Gifts to the "Ignored" State
- Obtaining a List of Nearby Users
- Obtaining a List of Nearby Users Discovered through the Most Recent Update Processing of the "near" Application
- Obtaining a List of Nearby Users Newly Discovered through the Most Recent Update Processing of the "near" Application
- Obtaining the Usage Status of the "near" application

For how to use the following "near" Dialog features, refer to the ""near" Dialog Feature Usage Method" chapter.

- Updating "near" with the Dialog Format (gift distribution/discovery)
- Receiving Discovered Gifts with the Dialog Format
- Aborting the Processing

Setting the Gifts to be Distributed

(1) Initialize the library with `sceNearUtilityInitialize()`.

(2) Set the icon image, data, name, description, receipt conditions and distribution quantity of the gifts to be distributed with `sceNearUtilitySetGift()`/`sceNearUtilitySetGift2()`.
*Use `sceNearUtilitySetGift2()` if you wish to use gift names and descriptions in multiple languages.
*If using `sceNearUtilitySetGift2()`, first it will be necessary to call `sceAppUtilInit()` and initialize application utility

(3) Terminate the library with `sceNearUtilityFinalize()`.

Checking the Gifts to be Distributed

(1) Initialize the library with `sceNearUtilityInitialize()`.

(2) Obtain the status of the last-set gift with `sceNearUtilityGetGiftStatus()`.

(3) Obtain the information of the last-set gifts with `sceNearUtilityGetGift()`.

(4) Terminate the library with `sceNearUtilityFinalize()`.

Deleting Distribution of a Gift

(1) Initialize the library with `sceNearUtilityInitialize()`.

(2) Delete the last-set gifts with `sceNearUtilityDeleteGift()`.

(3) Terminate the library with `sceNearUtilityFinalize()`.

Obtaining Discovered Gifts

(1) Initialize the library with `sceNearUtilityInitialize()`.

(2) Obtain the discovered gifts list with `sceNearUtilityGetDiscoveredGifts()`.

(3) Obtain the sender of a given gift on the list with `sceNearUtilityGetDiscoveredGiftSender()`.

(4) Obtain the gift status of a given gift on the list with `sceNearUtilityGetDiscoveredGiftStatus()`.

(5) Obtain gift name and description of a given gift on the list with `sceNearUtilityGetDiscoveredGiftInfo()`.

(6) Open the icon image file of a given gift on the list with `sceNearUtilityOpenDiscoveredGiftImage()`, read the contents of the image with `sceNearUtilityReadDiscoveredGiftImage()`, then close the image file with `sceNearUtilityCloseDiscoveredGiftImage()`.

(7) Open the data file of a given gift on the list with `sceNearUtilityOpenReceivedGiftData()`, read the contents of the data with `sceNearUtilityReadReceivedGiftData()`, then close the data file with `sceNearUtilityCloseReceivedGiftData()`.
*The gift's data must have already been received.

(8) Terminate the library with `sceNearUtilityFinalize()`.

Using Received Gifts

Obtain the parameters when the game program was launched from the "Discoveries" screen of the "near" application

(1) The game program handling received gifts is started up from the "near" application's "Discoveries" screen (depending on the user's intention).
*The "near" Dialog utility will not participate directly in processing in the "near" application's "Discoveries" screen.

(2) The game program starts, and the "near" gift event parameters are received. (For details, refer to the ""near" Gift Event Parameters".)

(3) Parse the "near" gift event parameters with `sceAppUtilAppEventParseNearGift()` of the application utility.

Proceed with the processing in the next section, "Using Received Gifts With Game Programs

Using Received Gifts With Game Programs

(1) Initialize the library with `sceNearUtilityInitialize()`.

(2) Obtain the list of discovered gifts with `sceNearUtilityGetDiscoveredGifts()`.

(3) Once the parsed "near" gift event parameters have been obtained in the previous process, convert the parsed "near" gift event parameters with `sceNearUtilityConvertDiscoveredGiftParam()` and obtain the `SceNearGiftDiscoveringId` type ID of a specified gift.

(4) Open data file of the gift with the ID specified using `SceNearGiftDiscoveringId` type with `sceNearUtilityOpenReceivedGiftData()`, read the contents of the data with `sceNearUtilityReadReceivedGiftData()`, then close the data file with `sceNearUtilityCloseReceivedGiftData()`.

(5) Terminate the library with `sceNearUtilityFinalize()`.

Deleting Discovered Gifts

(1) Initialize the library with `sceNearUtilityInitialize()`.
(2) Obtain the list of discovered gifts with `sceNearUtilityGetDiscoveredGifts()`.
(3) Delete the gifts on the list with `sceNearUtilityDeleteDiscoveredGift()`.
(4) Terminate the library with `sceNearUtilityFinalize()`.

Setting Discovered Gifts to the "Ignored" State

(1) Initialize the library with `sceNearUtilityInitialize()`.
(2) Obtain the list of discovered gifts with `sceNearUtilityGetDiscoveredGifts()`.
(3) Set the gifts in the list to the "Ignored" state with `sceNearUtilityIgnoreDiscoveredGift()`.
(4) Terminate the library with `sceNearUtilityFinalize()`.

Obtaining a List of Nearby Users

(1) Initialize the library with `sceNearUtilityInitialize()`.
(2) Obtain a list of nearby users with `sceNearUtilityGetNeighbors()`.
(3) Terminate the library with `sceNearUtilityFinalize()`.

Obtaining a List of Nearby Users Discovered through the Most Recent Update Processing of the "near" Application

(1) Initialize the library with `sceNearUtilityInitialize()`.
(2) Obtain the time at which the "near" application has last discovered a nearby user with `sceNearUtilityGetLastNeighborFoundDateTime()`.
(3) Specify the time obtained through the step (2), call `sceNearUtilityGetRecentNeighbors()`, and obtain a list of discovered nearby users.
(4) Terminate the library with `sceNearUtilityFinalize()`.

Obtaining a List of Nearby Users Newly Discovered through the Most Recent Update Processing of the "near" Application

(1) Initialize the library with `sceNearUtilityInitialize()`.
(2) Obtain the time at which the "near" application has last discovered a nearby user with `sceNearUtilityGetLastNeighborFoundDateTime()`.
(3) Specify the time obtained through the step (2), call `sceNearUtilityGetNewNeighbors()`, and obtain a list of nearby users newly discovered.
(4) Terminate the library with `sceNearUtilityFinalize()`.

Obtaining the Usage Status of the "near" application

(1) Initialize the library with `sceNearUtilityInitialize()`.
(2) Obtain the usage status of the "near" application with `sceNearUtilityGetMyStatus()`.
(3) Terminate the library with `sceNearUtilityFinalize()`.

Main APIs Used in Basic Processing

API	Description
`sceNearUtilityInitialize()`	Initializes the library
`sceNearUtilityFinalize()`	Terminates the library
`sceNearUtilitySetGift()`	Sets gifts to be distributed
`sceNearUtilitySetGift2()`	Sets gifts to be distributed (supports character strings in multiple languages)
`sceNearUtilityGetGift()`	Obtains the information of the gift that has last been set
`sceNearUtilityGetGiftStatus()`	Obtains the status of the gift that has last been set
`sceNearUtilityDeleteGift()`	Deletes the gift that has last been set
`sceNearUtilityGetDiscoveredGifts()`	Obtains the list of discovered gifts
`sceNearUtilityDeleteDiscoveredGift()`	Deletes discovered gifts
`sceNearUtilityIgnoreDiscoveredGift()`	Sets discovered gifts to the "Ignored" state
`sceNearUtilityGetDiscoveredGiftSender()`	Obtains the sender of discovered gifts
`sceNearUtilityGetDiscoveredGiftInfo()`	Obtains character string information of discovered gifts
`sceNearUtilityGetDiscoveredGiftStatus()`	Obtains the storage status of discovered gifts
`sceNearUtilityOpenDiscoveredGiftImage()`	Opens the image files of discovered gifts
`sceNearUtilityReadDiscoveredGiftImage()`	Reads the image files of discovered gifts
`sceNearUtilityCloseDiscoveredGiftImage()`	Closes the image files of discovered gifts
`sceNearUtilityOpenReceivedGiftData()`	Opens the data files of received gifts
`sceNearUtilityReadReceivedGiftData()`	Reads the data files of received gifts
`sceNearUtilityCloseReceivedGiftData()`	Closes the data files of received gifts
`sceNearUtilityLaunchNearAppForUpdate()`	Prompts the user to update the information by launching the "near" application
`sceNearUtilityLaunchNearAppForDownload()`	Prompts receiving of discovered gifts by launching the "near" application
`sceNearUtilityGetNeighbors()`	Gets the list of nearby users
`sceNearUtilityGetRecentNeighbors()`	Obtains a list of nearby users discovered at or after the specified time
`sceNearUtilityGetNewNeighbors()`	Obtains a list of nearby users newly discovered at or after the specified time
`sceNearUtilityGetLastNeighborFoundDateTime()`	Obtains the time at which the "near" application has last discovered a nearby user
`sceNearUtilityConvertDiscoveredGiftParam()`	Converts the "near" gift event parameters and extracts the required values
`sceNearUtilityGetMyStatus()`	Obtains the usage status of the "near" application
`sceNearUtilityRefresh()`	Obtains the latest information of the "near" application, saving it to the work memory of the library

"near" Gift Event Parameters

When the "near" application receives a gift, it is possible to start up a game program that can handle the received gift from the "Discoveries" screen's list of discovered gifts.

At this point, "near" gift event parameters will be passed to the game program that has been started up.

"near" gift event parameters can be obtained using the application utility. By giving `SceAppUtilAppEventParam` received by using `sceAppUtilReceiveAppEvent()` to the parse function `sceAppUtilAppEventParseNearGift()` for "near", it is possible to receive "near" gift event parameters as the `SceAppUtilNearGiftParam` structure.

The `SceAppUtilNearGiftParam` structure includes the first 256 bytes (`SCE_NEAR_GIFT_DATA_PARAM_MAX_SIZE`) of the gift data received by the "near" application as GiftData.

These 256-byte giftData are set so as to be embedded in the gift data by the side distributing the gift, and received without any changes by the receiving side. The "near" application and the "near" server will not process their content.

The game program on the receiving side must be prepared so as to be able to interpret the giftData embedded by the game program on the sending side.

Obtain `SceNearGiftDiscoveringId` through `sceNearUtilityConvertDiscoveredGiftParam()` to find out the ID of the gift specified in the "Discoveries" screen of the "near" application.

For the application utility, refer to the "Application Utility Reference" document.

3 "near" Dialog Feature Usage Method

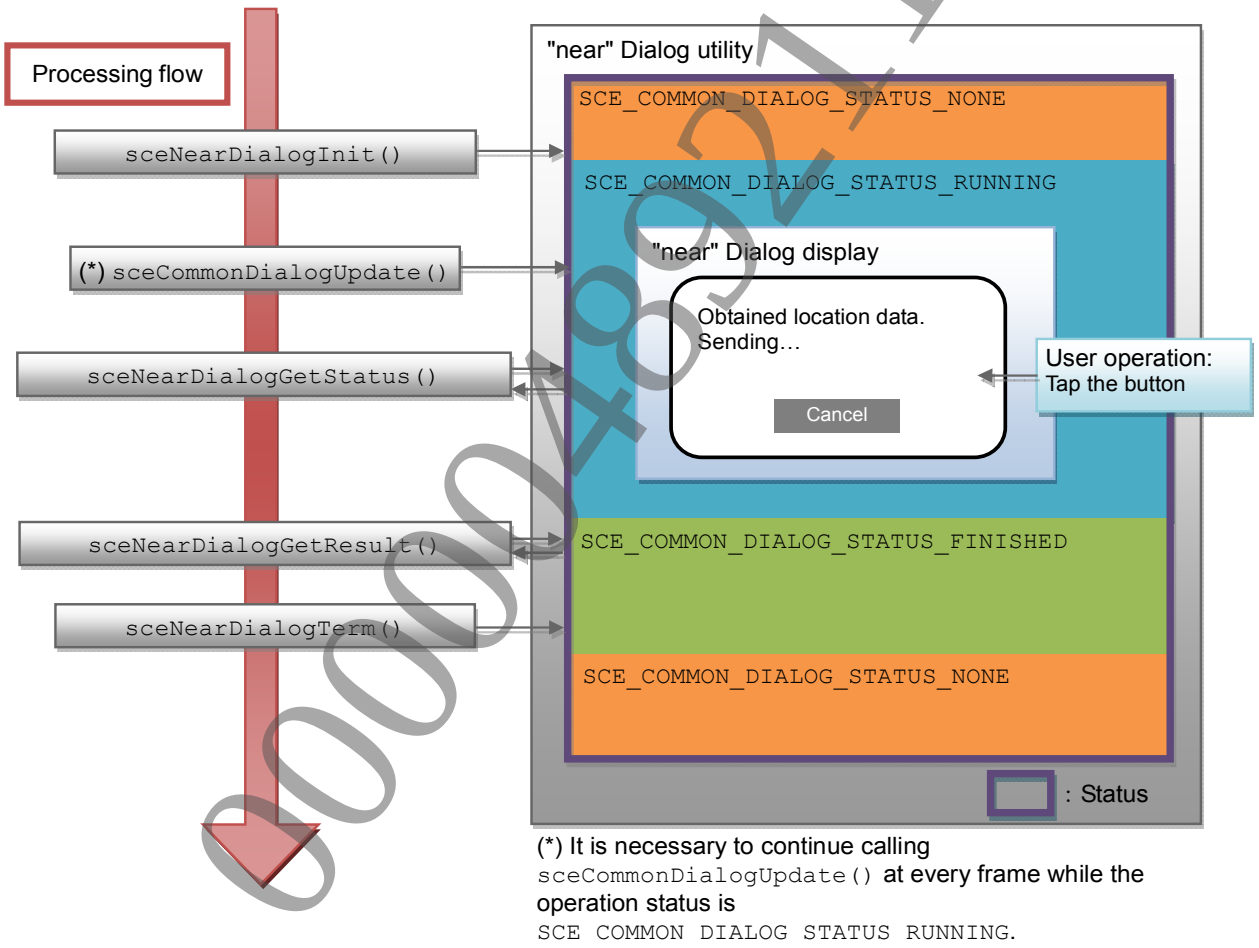
This chapter explains how to use the feature to use some of the features of the "near" application with the dialog format ("near" Dialog feature). The functions related to calling this feature have the `sceNearDialogXXXXX` prefix.

Basic Usage Procedure

The basic procedure to use the "near" Dialog feature is described. The processing flow is outlined below.

(1) Call the feature
(2) Wait for the response from the dialog
(3) Obtain the call results.
(4) Terminating the processing

Figure 1 Basic Processing Procedure



(*) It is necessary to continue calling `sceCommonDialogUpdate()` at every frame while the operation status is `SCE_COMMON_DIALOG_STATUS_RUNNING`.

Updating "near" with the Dialog Format

(1) Confirming with the user whether to update "near"

Prior to performing the update processing for "near", confirm with the user whether to update "near" using the location information. An example of the message that will be displayed at this time is given in the ""near" Update Processing Confirmation Message" chapter.

(2) Transitioning to the online state of PSN℠

The online state of PSN℠ is transitioned to by using Network Check Dialog. For details on the processing method, refer to the "NP Library Overview" document and "libnetctl Reference" document.

However, if PSN℠ is already in the online state, there is no need to call Network Check Dialog.

(3) Checking chat restrictions

Based on TRC (Technical Requirements Checklist) R3053, if required, check the chat restrictions of the Sony Entertainment Network account by getting the parental control information.

For the check method, refer to the "NP Library Reference" document.

(4) Calling the feature

Call the "near" Dialog feature with `sceNearDialogInit()`. Specify as the argument the `SceNearDialogParam` variable for which the required parameters for updating "near" have been set in advance.

*If updating "near" with the dialog format, the above-described processing must be performed in the state where `sceNearUtilityInitialize()` has not been called, or in the state where `sceNearUtilityFinalize()` has been called. For details, refer to the ""near" Dialog Utility States and Feature Call Availability" chapter.

(5) Waiting for the response from the dialog

Call `sceNearDialogGetStatus()` to poll the operation status of "near" Dialog at each frame.

> **Note**
> `sceCommonDialogUpdate()` must be called at every frame while the operation status is `SCE_COMMON_DIALOG_STATUS_RUNNING`. For details, refer to the "Common Dialog Overview" document.

(6) Obtaining the call results

When the operation status changes to `SCE_COMMON_DIALOG_STATUS_FINISHED`, the results can be obtained with `sceNearDialogGetResult()`.

(7) Terminating the processing

When the operation status becomes `SCE_COMMON_DIALOG_STATUS_FINISHED`, call `sceNearDialogTerm()` to terminate the processing. As a result, the resources acquired during calling are released, and the operation status becomes `SCE_COMMON_DIALOG_STATUS_NONE`.

Receiving Discovered Gifts with the Dialog Format

(1) Initializing the library

Initialize the library with `sceNearUtilityInitialize()`.

(2) Obtaining the list of discovered gifts

Obtain the list of discovered gifts with `sceNearUtilityGetDiscoveredGifts()`.

*The gifts to be received are those indicated by the ID acquired here.

(3) Transitioning to the online state of PSN℠

Transition to the online state of PSN℠ using Network Check Dialog. For the detailed processing method, refer to the "NP Library Overview" document and "libnetctl Reference" document.

However, if PSN℠ is already in the online state, there is no need to call Network Check Dialog.

(4) Calling the feature

Call the "near" Dialog feature with `sceNearDialogInit()`. Specify as the argument the `SceNearDialogParam` type variable for which the parameters required for receiving gifts have been set.

*To receive discovered gifts with the dialog format, the feature must be called in the state where the library has been initialized with `sceNearUtilityInitialize()`. For details, refer to the ""near" Dialog Utility States and Feature Call Availability" chapter.

*The IDs of discovered gifts obtained in step (2) become invalid when `sceNearUtilityFinalize()` is called, so be careful not to perform operations that would disable the IDs of discovered gifts, such as calling `sceNearUtilityFinalize()` prior to calling `sceNearDialogInit()`.

(5) Waiting for the response from the dialog

Call `sceNearDialogGetStatus()` and poll the operation status of "near" Dialog at each frame.

> **Note**
> `sceCommonDialogUpdate()` must be called at every frame while the operation status is `SCE_COMMON_DIALOG_STATUS_RUNNING`. For details, refer to the "Common Dialog Overview" document.

(6) Obtaining the call results

When the operation status changes to `SCE_COMMON_DIALOG_STATUS_FINISHED`, the results can be retrieved with `sceNearDialogGetResult()`.

(7) Terminating the processing

When the operation status becomes `SCE_COMMON_DIALOG_STATUS_FINISHED`, call `sceNearDialogTerm()` to terminate the processing. As a result, the resources acquired during calling are released, and the operation status becomes `SCE_COMMON_DIALOG_STATUS_NONE`.

(8) Reflecting the received gift data to the game

Reflect the received gift data to the game by using `sceNearUtilityOpenReceivedGiftData()`, `sceNearUtilityReadReceivedGiftData()`, or `sceNearUtilityCloseReceivedGiftData()`.

(9) Set the library to its uninitialized state

Return the state of the library initialized in step (1) to the uninitialized state with `sceNearUtilityFinalize()`.

Aborting the Processing

To abort the display of the "near" Dialog feature from the application side on an emergency basis, for example, when quitting the application, call `sceNearDialogAbort()`. Display of "near" Dialog will immediately end and the operation status will change to `SCE_COMMON_DIALOG_STATUS_FINISHED`. In this case, too, the call result will be obtained with `sceNearDialogGetResult()`. `SCE_COMMON_DIALOG_RESULT_ABORTED` is returned as retrieved result.

Main APIs Used for "near" Dialog Feature

API	Description
`SceNearDialogParam`	Parameter structure including mode setting
`sceNearDialogParamInit()`	Initializes parameter structure
`sceNearDialogInit()`	Performs initialization to use the "near" Dialog feature
`sceNearDialogGetStatus()`	Gets the status of the "near" Dialog feature
`sceNearDialogAbort()`	Aborts the processing of the "near" Dialog feature
`sceNearDialogGetResult()`	Gets the processing results of the "near" Dialog feature
`sceNearDialogTerm()`	Terminates use of the "near" Dialog feature

4 "near" Dialog Utility States and Feature Call Availability

This chapter describes the internal states of the "near" Dialog utility, and the "near" utility features and the "near" Dialog features, which can be called in each state.

States of "near" Utility Feature

The "near" Dialog utility has two main internal states, the "near" utility feature's initialized state and uninitialized state.

Uninitialized state of "near" utility features

This is the state where, after the "near" Dialog utility has been loaded, `sceNearUtilityInitialize()` is not called. Alternatively, this is the state where, after `sceNearUtilityInitialize()` has been called, `sceNearUtilityFinalize()` is called.

Initialized state of "near" utility features

This is the state where, after the "near" Dialog utility has been loaded, `sceNearUtilityInitialize()` is called but `sceNearUtilityFinalize()` is not subsequently called.

Common Dialog States

The "near" Dialog utility having a function as a Common Dialog, it has the same three states as a Common Dialog (refer to Figure 2).

SCE_COMMON_DIALOG_STATUS_NONE

Initialized state. This is the state in which the "near" Dialog feature is not used.

SCE_COMMON_DIALOG_STATUS_RUNNING

This is the state during which the "near" Dialog feature is being used.

SCE_COMMON_DIALOG_STATUS_FINISHED

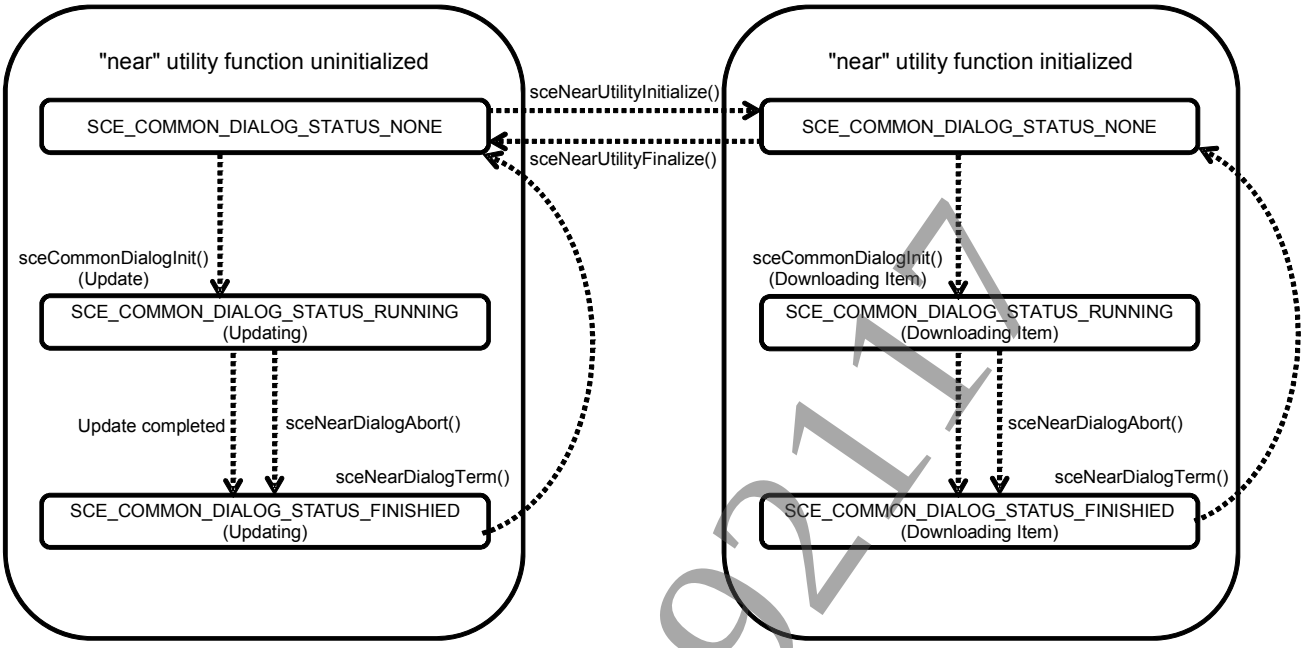
This state is entered when the processing of "near" Dialog ends. The GUI of "near" Dialog disappears from the screen. The application can obtain the "near" Dialog execution results in this state.

The function to obtain these states is `sceNearDialogGetStatus()`.

Feature Call Availability in Each State

The feature call availability in each state is described below.

Figure 2 "near" Dialog Utility States and Functions That Can Be Called



The functions that can be called in each state of the "near" utility feature/"near" Dialog feature are listed below.

State	Operation Status	Feature	Callable Function
"near" utility feature uninitialized	`SCE_COMMON_DIALOG_STATUS_NONE`	"near" utility	`sceNearUtilityInitialize()`

		"near" Dialog	`sceNearDialogParamInit()`
			`sceNearDialogInit()`
			(Specify `SCE_NEAR_DIALOG_MODE_UPDATE` for *mode* of `SceNearDialogParam`)
			`sceNearDialogGetStatus()`

	`SCE_COMMON_DIALOG_STATUS_RUNNING`	"near" utility	None

		"near" Dialog	`sceNearDialogParamInit()`
			`sceNearDialogGetStatus()`
			`sceNearDialogAbort()`

	`SCE_COMMON_DIALOG_STATUS_FINISHED`	"near" utility	None

		"near" Dialog	`sceNearDialogParamInit()`
			`sceNearDialogGetStatus()`
			`sceNearDialogTerm()`

State	Operation Status	Feature	Callable Function
"near" utility feature initialized	SCE_COMMON_DIALOG_STATUS_NONE	"near" utility	All functions except sceNearUtilityInitialize()
		"near" Dialog	sceNearDialogParamInit() sceNearDialogInit() (Specify SCE_NEAR_DIALOG_MODE_DOWNLOAD for *mode* of SceNearDialogParam) sceNearDialogGetStatus()
	SCE_COMMON_DIALOG_STATUS_RUNNING	"near" utility	None
		"near" Dialog	sceNearDialogParamInit() sceNearDialogGetStatus() sceNearDialogAbort()
	SCE_COMMON_DIALOG_STATUS_FINISHED	"near" utility	None
		"near" Dialog	sceNearDialogParamInit() sceNearDialogGetStatus() sceNearDialogTerm()

Uninitialized State of "near" Utility Features

"near" Dialog state: SCE_COMMON_DIALOG_STATUS_NONE

- "near" utility feature
  Only sceNearUtilityInitialize() can be called.
  When the function is successfully called, the "near" utility feature initialized state is entered, and the operation status of "near" Dialog remains SCE_COMMON_DIALOG_STATUS_NONE
- "near" Dialog feature
  sceNearDialogParamInit(), sceNearDialogInit(), and sceNearDialogGetStatus() can be called. However, sceNearDialogInit() can be called only if SCE_NEAR_DIALOG_MODE_UPDATE has been set for *mode* of SceNearDialogParam.
  When sceNearDialogInit() is successfully called, the operation status of "near" Dialog changes immediately to SCE_COMMON_DIALOG_STATUS_RUNNING, but the "near" utility feature uninitialized state remains unchanged.

"near" Dialog state: SCE_COMMON_DIALOG_STATUS_RUNNING (Updating)

- "near" utility feature
  There are no functions that can be called.
- "near" Dialog feature
  sceNearDialogParamInit(), sceNearDialogGetStatus(), and sceNearDialogAbort() can be called.
  When sceNearDialogAbort() is successfully called, the operation status of "near" Dialog changes to SCE_COMMON_DIALOG_STATUS_FINISHED after a while, but the "near" utility feature uninitialized state remains unchanged.

"near" Dialog state: SCE_COMMON_DIALOG_STATUS_FINISHED (Updating)

- "near" utility feature
  There are no functions that can be called.
- "near" Dialog feature
  sceNearDialogParamInit(), sceNearDialogGetStatus(), and sceNearDialogTerm() can be called.
  When sceNearDialogTerm() is successfully called, the operation status of "near" Dialog changes immediately to SCE_COMMON_DIALOG_STATUS_NONE, but the "near" utility feature uninitialized state remains unchanged.

"near" Utility Initialized State

"near" Dialog state: `SCE_COMMON_DIALOG_STATUS_NONE`

- "near" utility feature
  All functions except `sceNearUtilityInitialize()` can be called.
  When `sceNearUtilityFinalize()` is successfully called, the "near" utility feature uninitialized state is entered and the operation status of "near" Dialog remains `SCE_COMMON_DIALOG_STATUS_NONE`.
- "near" Dialog feature
  `sceNearDialogParamInit()`, `sceNearDialogInit()`, and `sceNearDialogGetStatus()` can be called. However, `sceNearDialogInit()` can be called only with `SCE_NEAR_DIALOG_MODE_DOWNLOAD` set for *`mode`* of `SceNearDialogParam`.
  When `sceNearDialogInit()` is successfully called, the operation status of "near" Dialog changes immediately to `SCE_COMMON_DIALOG_STATUS_RUNNING`, but the "near" utility feature initialized state remains unchanged.

"near" Dialog state: `SCE_COMMON_DIALOG_STATUS_RUNNING` (Downloading Item)

- "near" utility feature
  There are no functions that can be called.
- "near" Dialog feature
  `sceNearDialogParamInit()`, `sceNearDialogGetStatus()`, and `sceNearDialogAbort()` can be called.
  When `sceNearDialogAbort()` is successfully called, the operation status of "near" Dialog changes to `SCE_COMMON_DIALOG_STATUS_FINISHED` after a while, but the "near" utility feature initialized state remains unchanged.

"near" Dialog state: `SCE_COMMON_DIALOG_STATUS_FINISHED` (Downloading Item)

- "near" utility feature
  There are no functions that can be called.
- "near" Dialog feature
  `sceNearDialogParamInit()`, `sceNearDialogGetStatus()`, and `sceNearDialogTerm()` can be called.
  When `sceNearDialogTerm()` is successfully called, the operation status of "near" Dialog changes immediately to `SCE_COMMON_DIALOG_STATUS_NONE`, but the "near" utility feature initialized state remains unchanged.

5 Processing Content and Processing Results of "near" Dialog Feature

This chapter describes the interaction with the user when the "near" Dialog feature is called, and the processing on the game program side in response to the result of this interaction.

Updating "near" with the Dialog Format

Processing Flow

The main processing flow when update processing of "near" is performed with the "near" Dialog utility, the types of dialogs that are displayed in each processing phase, and the return code that is returned to the game program when the processing is aborted, are shown in Table 1.

Table 1 shows the state transitions of the processing of the "near" Dialog feature. If the content in the "Condition" column is satisfied, the corresponding dialog is displayed, and if content in the "Condition" column is not satisfied, the processing moves on to the next condition (the next line in the table).

Table 1 "near" Dialog Feature Processing and Return Codes Returned to Game Program

Condition	Content of Displayed Dialog	Return Code
Not signed in to PSN℠?	You are signed out of PSN℠. To use "near", please sign in to PSN℠.	**OK** tap SCE_NEAR_DIALOG_RESULT_NOT_SIGNIN
"near" currently updating?	Updating. Please wait.	**OK** tap SCE_NEAR_DIALOG_RESULT_ALREADY_UPDATING
"near" initialization failed?	Your "near" data was not deleted. Please launch "near" and delete all data. Launch "near?"	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_INITIALIZATION **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
Have not accepted the Terms of Service of "near"?	You have not used "near" yet. Please launch near and review the Terms of Service. Launch "near?"	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_USER_AGREEMENT **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
Connection to the Internet failed?	Could not connect to the Internet.	**OK** tap libnetctl, libnet and libhttp error codes
Found new Terms of Service of "near"?	The "near" Terms of Service has been updated. Please launch "near" to review. Launch "near" now?	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_USER_AGREEMENT **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
The "near" setting is **Do not ask my permission again**.	Game Goods cannot be accessed with the current Sharing Settings. Open "Sharing Settings" in "near" and check "Do not ask my permission again." Launch "near" now?	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_UPDATE_PERMISSION **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED

Condition	Content of Displayed Dialog	Return Code
Is "Share online ID" not set?	Game Goods cannot be accessed because you are not sharing your online ID. Please open "near" and make your ID public. Launch "near" now?	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_SHARED_USERID **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
Set to privacy content?	No sharable content available. You will need to change the Private Game Settings or play a game that isn't set as a Private Game	**Launch "near"** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_SHARED_CONTENT **Continue game** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
"near" not allowed to use location data?	You haven't given "near" permission to use your location data. To use this function, first give "near" permission to use your location data in the Settings>Location Data menu or on the "near" Settings screen.	**OK** tap SCE_NEAR_DIALOG_RESULT_LOCATION_NOT_PERMITTED
Location calculation canceled?	Could not obtain location data. This location's information will be updated when you connect to the internet again.	**OK** tap SCE_NEAR_DIALOG_RESULT_LOCATION_CANCELED
Wi-Fi setting is OFF?	To obtain location data, you must go to the home screen and select [Settings] > [Network] > [Wi-Fi Settings] and then turn [Wi-Fi] on.	**OK** tap SCE_NEAR_DIALOG_RESULT_WIFI_OFF
Transmission was canceled by the user?	N/A	SCE_NEAR_DIALOG_RESULT_SEND_CANCELED
Service currently undergoing maintenance?	This service is currently undergoing maintenance.	**OK** tap SCE_NEAR_DIALOG_RESULT_SERVER_MAINTENANCE
Service ended?	This service is no longer available.	**OK** tap SCE_NEAR_DIALOG_RESULT_SERVER_END
Transmission completed?	The following information was sent: Location Data: XX Distance Traveled: XXkm Distance Traveled Today: XXkm	**OK** tap Transition to the next dialog (next line in the table)
Returned to game?	Your data has been uploaded but the Update hasn't finished yet. Are you sure you want to return to your game? If you return to your game before the Update has finished, you might not be able to find new Game Goods or get updated data.	**Return to Game** tap SCE_SEND_OK **Continue your "near" update** tap Transition to the next dialog (next line in the table)
Continue updating of "near"?	Receiving information from "near"…	**Return to Game** tap SCE_SEND_OK If a network error or some other error occurs SCE_SEND_OK

Condition	Content of Displayed Dialog	Return Code
Was information reception successful?	The "near" update has finished.	**OK** tap SCE_OK

Types of "near" Update Processing Results and Handling by Game Program

The values that can be obtained by *result* of sceNearDialogGetResult() when the "near" update processing is performed, and the required handling by the game program side in response are described below.

Return code that allows processing to continue based on the assumption that the processing was entirely successful

```
SCE_COMMON_DIALOG_RESULT_OK
```

This is the return code that is returned when transmission/reception to/from the "near" server is successful. The game program can be made to progress based on the assumption that the "near" update processing has been successful.

Return code that allows selection of handling on game program side

If the following return code is returned, the game program side selects the handling according to the purpose of the call.

```
SCE_NEAR_DIALOG_RESULT_SEND_OK
```

This is the return code that is returned if although data transmission, such as the transmission of gifts to the "near" server, was successful, the data that should be received from the "near" server is not received in its entirety.

When "near" update processing is called for the purpose of sending gifts to a "near" server or updating the travel distance of the user, make the game progress on the assumption that the processing was successful on the game program side.

If the "near" update processing was called for the purpose of updating the number of discovered games, the number of discovered items, or the number of discovered nearby users, or of receiving gifts from nearby users or friends, that needs to be received from the "near" server, the game program should call the "near" update processing again or make the game progress.

Return codes that require the "near" application to be launched

If the user chooses to launch the "near" application in "near" Dialog in order to continue the "near" update processing, the following return codes are returned. The game program should call sceNearUtilityLaunchNearAppForUpdate().

```
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_INITIALIZATION
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_USER_AGREEMENT
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_UPDATE_PERMISSION
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_SHARED_USERID
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_SHARED_CONTENT
```

If the "near" application is called using sceNearUtilityLaunchNearAppForUpdate(), or the PS button is pressed, the game program enters the suspended state.

When the game program is suspended while the "near" Dialog utility is in the initialized state, and update processing is performed on the "near" application side during this time, the information held by the library in the work memory becomes outdated, so re-obtain the information. For details, refer to the "Notes" chapter.

Return codes for which the processing should be continued on the assumption that the processing failed

If the "near" update processing failed owing to the decision of the user or neither the user nor the game program could handle the processing, the following return codes are returned. Make the game program progress on the assumption that the "near" update processing failed.

```
SCE_NEAR_DIALOG_RESULT_NOT_SIGNIN
SCE_NEAR_DIALOG_RESULT_ALREADY_UPDATING
SCE_NEAR_DIALOG_RESULT_LOCATION_NOT_PERMITTED
SCE_NEAR_DIALOG_RESULT_LOCATION_CANCELED
SCE_NEAR_DIALOG_RESULT_WIFI_OFF
SCE_NEAR_DIALOG_RESULT_SEND_CANCELED
SCE_NEAR_DIALOG_RESULT_SERVER_MAINTENANCE
SCE_NEAR_DIALOG_RESULT_SERVER_END
SCE_COMMON_DIALOG_RESULT_USER_CANCELED
```

Receiving Discovered Gifts with Dialog Format

Processing Flow

The main processing flow when receiving gifts, the types of dialogs displayed in each processing phase, and the return codes returned to the game program when the processing is aborted, are shown in Table 2.

Table 2 shows the state transitions when **OK** is tapped in the dialog for confirming whether to update "near". If the content in the "Condition" column is satisfied, the corresponding dialog is displayed, and if content in the "Condition" column is not satisfied, the processing moves on to the next condition (the next line in the table).

Table 2 Gift Receive Processing and Return Codes Returned to Game Program

Condition	Content of Displayed Dialog	Return Code
Not signed in to PSN℠?	You are signed out of PSN℠. To use "near", please sign in to PSN℠.	**OK** tap SCE_NEAR_DIALOG_RESULT_NOT_SIGNIN
"near" initialization failed?	Your "near" data was not deleted. Please launch "near" and delete all data. Launch "near?"	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_INITIALIZATION **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
Downloading gifts with the "near" application?	Downloading Game Goods. Please open "near" and wait for the download to finish. Launch "near?"	**Continue game** tap SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_GIFT_DOWNLOAD **Launch "near"** tap SCE_COMMON_DIALOG_RESULT_USER_CANCELED
Gift downloading completed?	Downloading Game Goods. Please open "near" and wait for the download to finish. Launch "near?"	**Cancel** tap SCE_NEAR_DIALOG_RESULT_DOWNLOAD_CANCELED
Connection to the Internet failed?	Could not connect to the Internet.	**OK** tap libnetctl, libnet and libhttp error codes
Service currently undergoing maintenance?	This service is currently undergoing maintenance.	**OK** tap SCE_NEAR_DIALOG_RESULT_SERVER_MAINTENANCE

Condition	Content of Displayed Dialog	Return Code
Service ended?	This service is no longer available.	**OK** tap SCE_NEAR_DIALOG_RESULT_SERVER_END
No gifts on the server?	That Game Good is no longer available. Try searching for more Game Goods.	**OK** tap SCE_NEAR_DIALOG_RESULT_GIFT_NOT_FOUND
Data corrupted? (data falsified, etc.)	The data is corrupt.	**OK** tap SCE_NEAR_DIALOG_RESULT_GIFT_CORRUPTED
Expired?	This item has expired.	**OK** tap SCE_NEAR_DIALOG_RESULT_GIFT_EXPIRED
Not expired?	Game Good downloaded successfully!	**OK** tap SCE_COMMON_DIALOG_RESULT_OK

Gift Receive Processing Result Type and Handling on Game Program Side

The value returned when gift receive processing is performed and the required handling on the game program side in response are described below.

Return code that allows processing to continue on the assumption that the processing was entirely successful

```
SCE_COMMON_DIALOG_RESULT_OK
```

This is the return code that is returned from the "near" server when receiving of the specified gift data has been entirely successful. The game program can be made to progress on the assumption that reception of the gift data was successful.

Return codes that require the "near" application to be launched

If the user chooses to launch the "near" application in order to continue the gift receive processing, the following return codes are returned. The game program should call `sceNearUtilityLaunchNearAppForDownload()`.

```
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_INITIALIZATION
SCE_NEAR_DIALOG_RESULT_LAUNCH_NEAR_NEED_GIFT_DOWNLOAD
```

If the "near" application is called using `sceNearUtilityLaunchNearAppForDownload()`, or if the PS button is pressed, the game program enters the suspended state.
When the game program enters the suspended state while the "near" Dialog utility is in the initialized state, and update processing is performed on the "near" application side during this time, the information held by the library in the work memory becomes outdated, so re-obtain the information. For details, refer to the "Notes" chapter.

Return codes for which the processing should be continued on the assumption that the processing failed

Following return codes are returned if the gift receive processing failed owing to the decision of the user or neither the user nor the game program could handle the processing. On the game program side, make the game program progress on the assumption that the processing failed.

```
SCE_NEAR_DIALOG_RESULT_NOT_SIGNIN
SCE_COMMON_DIALOG_RESULT_USER_CANCELED
SCE_NEAR_DIALOG_RESULT_DOWNLOAD_CANCELED
SCE_NEAR_DIALOG_RESULT_SERVER_MAINTENANCE
SCE_NEAR_DIALOG_RESULT_SERVER_END
SCE_NEAR_DIALOG_RESULT_GIFT_NOT_FOUND
SCE_NEAR_DIALOG_RESULT_GIFT_CORRUPTED
SCE_NEAR_DIALOG_RESULT_GIFT_EXPIRED
```

6 Notes

Operating Notes

- The "near" Dialog utility starts up an external process to access the information held by the system software during gift registration/discovery/deletion processing. The functions for starting up an external process are as follows:

```
sceNearUtilityInitialize()
sceNearUtilitySetGift()
sceNearUtilitySetGift2()
sceNearUtilityDeleteGift()
sceNearUtilityDeleteDiscoveredGift()
sceNearUtilityIgnoreDiscoveredGift()
sceNearUtilityRefresh()
sceNearDialogInit()
sceNearDialogGetStatus()
sceNearDialogAbort()
sceNearDialogGetResult()
sceNearDialogTerm()
```

These functions communicate with the external process that has started and block until processing is complete. Therefore, take precautions when using them, such as waiting in a thread.

- When `sceNearUtilityInitialize()` is called or "near" is updated with the dialog format, the "near" Dialog utility obtains gift information managed by the system software via external process and retains it in the work memory.
  The functions below reference information in this work memory; therefore, you will not be able to obtain new information every time you call one of these functions:

```
sceNearUtilityGetGift()
sceNearUtilityGetGiftStatus()
sceNearUtilityGetDiscoveredGifts()
sceNearUtilityGetDiscoveredGiftSender()
sceNearUtilityGetDiscoveredGiftInfo()
sceNearUtilityGetDiscoveredGiftStatus()
sceNearUtilityOpenDiscoveredGiftImage()
sceNearUtilityOpenReceivedGiftData()
sceNearUtilityGetNeighbors()
sceNearUtilityGetRecentNeighbors()
sceNearUtilityGetNewNeighbors()
sceNearUtilityGetLastNeighborFoundDateTime()
sceNearUtilityGetMyStatus()
```

- While the library stores gift information or other information in the work memory by calling `sceNearUtilityInitialize()`, the user may exit the game program by pressing the PS button and may operate the "near" application.
  There is a possibility that the information stored in the work memory is no longer the latest one when the user returns to the game program; therefore, with the "near" Dialog utility in the initialized state, it is recommended for the game program to periodically call `sceAppMgrReceiveSystemEvent()`, and check if the game program has resumed or not. When `sceAppMgrReceiveSystemEvent()` is used to receive `SCE_APPMGR_SYSTEMEVENT_ON_RESUME`, the game program is resumed, so re-obtaining the latest information for the "near" application is recommended.
  One of two methods can be used to re-obtain the information: Either call `sceNearUtilityRefresh()`, or discard the outdated information with `sceNearUtilityFinalize()` and then execute `sceNearUtilityInitialize()` again.
- The state of gifts set to the "Ignored" state with `sceNearUtilityIgnoreDiscoveredGift()` will be managed by the "near" application. For details, refer to the "near System Overview" document.
- `SceNpCommunicationId` handled by titles
  - When using the "near" Dialog utility, specify `SceNpCommunicationId` at initialization. In this way, it will be possible to transfer gifts among different game titles
  - By temporarily terminating the "near" Dialog utility and initializing by specifying a different `SceNpCommunicationId`, it will be possible to obtain gifts distributed with a different `SceNpCommunicationId`
  - The `SceNpCommunicationId` specified at initialization will be used in the "near" Dialog utility as related information of the game title, and will be retained on the system software. If this is changed and initialization is performed again, the relation between the `SceNpCommunicationId` that was used previously and the game title will be lost.
  - If the relation between the game title and `SceNpCommunicationId` is lost, the following operations will be affected.
    - If the `SceNearGiftId`'s bits meaning that the "gift cannot be discovered without a title using `SceNpCommunicationId`" are LOW when the "near" application discovers a gift on the "near" server, discovery will not be possible because there is no related game title.

  *In this version, given that, as stated above, when a gift is discovered only 1 `SceNpCommunicationId` will be related to 1 game title, 1 game title will only be able to discover gifts distributed with 1 `SceNpCommunicationId`.

  In future version upgrades, we plan to enable the discovery of gifts with multiple `SceNpCommunicationIds` by a single game title by performing simultaneous association of multiple `SceNpCommunicationId`s with a single game title
- It is not assumed that the functions of the "near" Dialog utility will be called simultaneously from multiple threads. When functions of the "near" Dialog utility are to be used by different threads, perform exclusive control so that their respective processing does not overlap.
- The "near" Dialog utility is not designed to be used at the same time as the "near" utility. Values obtained using the "near" utility should not be passed to or used with the "near" Dialog utility.
- To comply with the TRC (Technical Requirements Checklist) R3053, Game programs should react in one of the following ways when the distribution of gifts qualifying as the "user generated contents" is registered:
  - Check the chat restrictions of the user's Sony Entertainment Network account with `sceNpManagerGetChatRestrictionFlag()`, and interrupt the registration of gift distribution if the account is subjected to chat restrictions.
  - Have the "near" application/"near" Dialog utility restrict transmission and reception by setting the 0x20000000 bit of the gift ID defined by `SceNearGiftId` to HIGH.

Operating Notes for Sample Program

To install a sample program in a DevKit/TestKit without starting it up from a debugger, it must be packaged. For packaging, use the setting file of Package Generator stored in the following location.

```
sample_code\network\api_near_dialog_util\data\nearutildialogsample.gp4p
```

Handling for When Errors Occur

Among the error codes returned by the functions of the "near" Dialog utility, the error codes indicated below are not problems of the game program. The handling methods are shown for when the following errors are returned.

- When `SCE_NEAR_ERROR_NETWORK_TIME_NOT_INITIALIZED` is returned
  This is the state in which the time obtained from the network and saved to the PlayStation®Vita has not been initialized. This error code may be returned within the normal scope of operation. As the time is automatically initialized by signing in to PSN℠, the following operation on the game program side is recommended.
  - Start up Network Check Dialog in the PSN℠ mode and prompt the user to sign in using Network Check Dialog.
    For information on Network Check Dialog, refer to the "Network Overview" document and the "libnetctl Reference" document.
- Error codes not listed in the "near Dialog Utility Reference" document
  Some of the functions offered as part of the functions of the "near" Dialog utility perform network access. Even if this is within the scope of normal operation, error codes not listed in the "near Dialog Utility Reference" document may be returned. Handle these error codes based on the information given in the "libnet Reference" document, "libnetctl Reference" document, and "libhttp Reference" document.
- When an unexpected error code is returned
  It is caused by a design error or implementation error of the "near" Dialog utility. Direct enquiries to Private support at the PlayStation®Vita Developer Network (https://psvita.scedev.net/).
  However, the application must not malfunction even if unexpected error codes are returned.

7 "near" Update Processing Confirmation Message

Before Updating "near" with the Dialog Format

The "near" update processing uses the location information of PlayStation®Vita. Before performing the "near" update processing using the "near" Dialog utility, indicate to the user that the "near" update processing will be performed, and start the update processing only if the agreement of the user is obtained. This confirmation is to be performed beforehand every time update processing is to be done.

Message Displayed before "near" Update Processing is Performed

Display the following message before starting the "near" update processing using the "near" Dialog utility, to let the user know that the "near" update processing will be performed.

language	Message
Japanese	″near″更新をしますか？
English	Would you like to perform a ″near″ update?
English UK	Would you like to perform a ″near″ update?
French	Voulez-vous procéder à la mise à jour de ″near″ ?
Spanish	¿Quieres actualizar ″near″?
German	Möchtest du ″near″ aktualisieren?
Italian	Vuoi eseguire un aggiornamento di ″near″?
Dutch	Wil je ″near″ bijwerken?
Portuguese	Queres actualizar o ″near″?
Portuguese_BR	Você quer fazer uma atualização do ″near″?
Russian	В ы хотите обновить ″near″?
Polish	Czy chcesz dokonać aktualizacji ″near″?
Finnish	Haluatko suorittaa ″near″-päivityksen?
Danish	Vil du opdatere ″near″?
Norwegian	Vil du gjøre en ″near″-oppdatering?
Swedish	Vill du uppdatera ″near″?
Turkish	Bir ″near″ güncellemesi gerçekleştirmek ister misin?
Korean	"near"를 갱신하시겠습니까?
SimplifiedChinese	确定要更新″near″吗？
TraditionalChinese	確定要更新″near″嗎？