# near Utility Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50







Document serial number: 000004892117

# Table of Contents

# 1 "near" Utility Overview

## Scope of This Document

This document describes the "near" utility that enables the transfer of various types of data between the "near" application and a game program running.

## Purpose and Features

The "near" utility is a library for allowing transfer of various types of data between the game program and the "near" application in order for use of the "near" game service. By transferring various types of data with the "near" application, game programs can use various in-game services such as the distribution of game data between users and the discovery of nearby users.

## Main Functions

The following are the main functions provided by the "near" utility.

- 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

## Used Resources

The "near" utility uses the following system resources.

| Resource | Description |
|---|---|
| Footprint | 44 KiB when PRX is loaded |
| Work memory | Uses the memory passed at the time of initialization |
| Thread | Unused |
| 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_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_UTIL ) != SCE_OK ){
    // Error handling
}
```

Unload loaded modules when they are no longer necessary.

Upon building the program, link llibSceNearUtil_stub.a.

It is necessary to initialize the NP library before using the "near" utility library.

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

## Sample Programs

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

**sample_code/network/api_near_util/basic/**

## Reference Material

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

# 2 Usage Procedure

## "near" Utility Features

This chapter explains the usage for each of the "near" utility features.

- Setting the Gifts to be Distributed
- Checking the Gifts to be Distributed
- Deleting Distribution of a Gift
- Sending Registered Gifts to the "near" Server
- Discovering Gifts
- Obtaining Discovered Gifts
- Receiving the Data of 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

## Setting the Gifts to be Distributed

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

(2) Set the icon image, data, name, description, receipt conditions and distribution quantity of the gifts to be distributed with `sceNearSetGift()`/`sceNearSetGift2()`.
*Use `sceNearSetGift2()` if you wish to use gift names and descriptions in multiple languages.
*If using `sceNearSetGift2()`, first it will be necessary to call `sceAppUtilInit()` and initialize application utility.

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

## Checking the Gifts to be Distributed

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

(2) Obtains the status of the last-set gift to the other user with `sceNearGetGiftStatus()`.

(3) Obtain the information of the last-set gifts to other users with `sceNearGetGift()`.

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

## Deleting Distribution of a Gift

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

(2) Delete the last-set gifts to other users with `sceNearDeleteGift()`.

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

## Sending Registered Gifts to the "near" Server

(1) Display in the game the appropriate message listed in the "Messages for Sending the Gift Information" item in Chapter 4 to request the user to accept to start up the "near" application.

(2) If the user accepts, call `sceNearLaunchNearAppForUpdate()`.

(3) The "near" application will start and prompt the user to perform an "Update"

(4) The user will perform the "Update" in the "near" application and send the gifts to the "near" server (depending on the user's intention)
*The "near" utility will not participate directly in the "Update" processing of the "near" application.

The game program enters the suspended state when the "near" application is called using `sceNearLaunchNearAppForUpdate()`, or through user operation.

When the game program enters the suspended state with the "near" utility library in the initialized state and "Update" processing is performed at the "near" application at that time, the information held by the library in the work memory becomes out of date. Re-obtain the information. For details, refer to the "Operating Notes" section.

**Note**
To launch the "near" application from game programs, use `sceNearLaunchNearAppForUpdate()` which was added to SDK 1.600. Do not use the obsolete `sceNearFinalizeAndLaunchNearApp()`. `sceNearFinalizeAndLaunchNearApp()` is scheduled for deletion in future SDK updates.

## Discovering Gifts

(1) The user starts up the "near" application.

(2) The user performs an "Update" in the "near" application and discovers the gifts that are being distributed from the "near" server.
*The "near" utility will not participate directly in the "Update" processing of the "near" application.

## Obtaining Discovered Gifts

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

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

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

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

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

(6) Open the icon image file of a given gift on the list with `sceNearOpenDiscoveredGiftImage()`, read the contents of the image with `sceNearReadDiscoveredGiftImage()`, then close the image file with `sceNearCloseDiscoveredGiftImage()`.

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

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

## Receiving the Data of Discovered Gifts

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

(2) Display the appropriate message listed in the "Messages for Receiving a Gift" section in Chapter 4 to request the user to accept to start up the "near" application.

(3) If the user accepts, call `sceNearLaunchNearAppForDownload()`.

(4) The "near" application will start, and prompt the user to perform the operations for receiving gift data in the "Discoveries" screen.

(5) The user performs the operations for gift reception in the "Discoveries" screen, and receives gift data from the "near" server (depending on the user's intention).
*The "near" utility will not participate directly in processing in the "near" application's "Discoveries" screen.
*In the above procedure, because the "near" utility library has not been terminated yet, terminate the library with `sceNearFinalize()` once the game program has finished using the received gift data itself.

The game program enters the suspended state when the "near" application is called using `sceNearLaunchNearAppForDownload()`, or through user operation.

When the game program enters the suspended state with the "near" utility library in the initialized state and "Update" processing is performed at the "near" application at that time, the information held by the library in the work memory becomes out of date. Re-obtain the information. For details, refer to the "Operating Notes" section.

**Note**
To launch the "near" application from game programs, use `sceNearLaunchNearAppForDownload()` which was added to SDK 1.600. Do not use the obsolete `sceNearFinalizeAndLaunchNearApp()`. `sceNearFinalizeAndLaunchNearApp()` is scheduled for deletion in future SDK updates.

## 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 gifts received from the "near" application's "Discoveries" screen is started up (depending on the user's intention).
*The "near" 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 Gifts Received With Game Programs

### Using Gifts Received With Game Programs

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

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

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

(4) Open data file of the gift with the ID specified using `SceNearGiftDiscoveringId` type with `sceNearOpenReceivedGiftData()`, read the contents of the data with `sceNearReadReceivedGiftData()`, then close the data file with `sceNearCloseReceivedGiftData()`.

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

## Deleting Discovered Gifts

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

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

(3) Delete the gifts on the list with `sceNearDeleteDiscoveredGift()`.

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

## Setting Discovered Gifts to the "Ignored" State

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

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

(3) Set the gifts in the list to the "Ignored" state with `sceNearIgnoreDiscoveredGift()`.

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

## Obtaining a List of Nearby Users

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

(2) Obtain a list of nearby users with `sceNearGetNeighbors()`.

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

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

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

(2) Obtain the time at which the "near" application has last discovered a nearby user with `sceNearGetLastNeighborFoundDateTime()`.

(3) Specify the time obtained through the step (2), call `sceNearGetRecentNeighbors()`, and obtain a list of discovered nearby users.

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

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

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

(2) Obtain the time at which the "near" application has last discovered a nearby user with `sceNearGetLastNeighborFoundDateTime()`.

(3) Specify the time obtained through the step (2), call `sceNearGetNewNeighbors()`, and obtain a list of nearby users newly discovered.

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

## Obtaining the Usage Status of the "near" application

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

(2) Obtain the usage status of the "near" application with `sceNearGetMyStatus()`.

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

## Main APIs Used in Basic Processing

| API | Description |
|---|---|
| `sceNearInitialize()` | Initializes the library |
| `sceNearFinalize()` | Terminates the library |
| `sceNearSetGift()` | Sets gifts to be distributed |
| `sceNearSetGift2()` | Sets gifts to be distributed (supports character strings in multiple languages) |
| `sceNearGetGift()` | Obtains the information of the gift that has last been set |
| `sceNearGetGiftStatus()` | Obtains the status of the gift that has last been set |
| `sceNearDeleteGift()` | Deletes the gift that has last been set |
| `sceNearGetDiscoveredGifts()` | Obtains the list of discovered gifts |
| `sceNearDeleteDiscoveredGift()` | Deletes discovered gifts |
| `sceNearIgnoreDiscoveredGift()` | Sets discovered gifts to the "Ignored" state |
| `sceNearGetDiscoveredGiftSender()` | Obtains the sender of discovered gifts |
| `sceNearGetDiscoveredGiftInfo()` | Obtains character string information of discovered gifts |
| `sceNearGetDiscoveredGiftStatus()` | Obtains the storage status of discovered gifts |
| `sceNearOpenDiscoveredGiftImage()` | Opens the image files of discovered gifts |
| `sceNearReadDiscoveredGiftImage()` | Reads the image files of discovered gifts |
| `sceNearCloseDiscoveredGiftImage()` | Closes the image files of discovered gifts |
| `sceNearOpenReceivedGiftData()` | Opens the data files of received gifts |
| `sceNearReadReceivedGiftData()` | Reads the data files of received gifts |
| `sceNearCloseReceivedGiftData()` | Closes the data files of received gifts |
| `sceNearLaunchNearAppForUpdate()` | Prompts the user to update the information by launching the "near" application |
| `sceNearLaunchNearAppForDownload()` | Prompts receiving of discovered gifts by launching the "near" application |
| `sceNearGetNeighbors()` | Gets the list of nearby users |
| `sceNearGetRecentNeighbors()` | Obtains a list of nearby users discovered at or after the specified time |
| `sceNearGetNewNeighbors()` | Obtains a list of nearby users newly discovered at or after the specified time |
| `sceNearGetLastNeighborFoundDateTime()` | Obtains the time at which the "near" application has last discovered a nearby user |
| `sceNearConvertDiscoveredGiftParam()` | Converts the "near" gift event parameters and extracts the required values |
| `sceNearGetMyStatus()` | Obtains the usage status of the "near" application |
| `sceNearRefresh()` | 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 `sceNearConvertDiscoveredGiftParam()` 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 Notes

## Operating Notes

- The "near" 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:

  ```
  sceNearInitialize()
  sceNearSetGift()
  sceNearSetGift2()
  sceNearDeleteGift()
  sceNearDeleteDiscoveredGift()
  sceNearIgnoreDiscoveredGift()
  sceNearRefresh()
  ```

  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 `sceNearInitialize()` is called, the "near" utility library 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:

  ```
  sceNearGetGift()
  sceNearGetGiftStatus()
  sceNearGetDiscoveredGifts()
  sceNearGetDiscoveredGiftSender()
  sceNearGetDiscoveredGiftInfo()
  sceNearGetDiscoveredGiftStatus()
  sceNearOpenDiscoveredGiftImage()
  sceNearOpenReceivedGiftData()
  sceNearGetNeighbors()
  sceNearGetRecentNeighbors()
  sceNearGetNewNeighbors()
  sceNearGetLastNeighborFoundDateTime()
  sceNearGetMyStatus()
  ```

- While the library stores gift information or other information in the work memory by calling `sceNearInitialize()`, 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" utility library 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 `sceNearRefresh()`, or discard the outdated information with `sceNearFinalize()` and then execute `sceNearInitialize()` again.
- The state of gifts set to the "Ignored" state with `sceNearIgnoreDiscoveredGift()` will be managed by the "near" application. For details, refer to the "near System Overview" document.
- `SceNpCommunicationId` handled by 1 title
  - When using the "near" utility library, specify `SceNpCommunicationId` at initialization. In this way, it will be possible to transfer gifts among different game titles.
  - By temporarily terminating the "near" utility library 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" 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 `SceNpCommunicationIds` with a single game title
  - Relation between the game title and `SceNpCommunicationId` is recorded into the system software by calling `sceNearInitialize()`. It is recommended to record the relation through `sceNearInitialize()` wherever possible when starting up the game title for the first time.
- It is not assumed that the functions of the "near" utility will be called simultaneously from multiple threads. When functions of the "near" utility are to be used by different threads, perform exclusive control so that their respective processing do not overlap.
- The "near" utility is not designed to be used at the same time as the "near" Dialog utility. Values obtained using the "near" Dialog utility should not be passed to or used with the "near" 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_util\data\nearutilsample.gp4p
```

## Handling for When Errors Occur

Among the error codes returned by the functions of the "near" 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.
- When an unexpected error code is returned
  It is caused by a design error or implementation error of the "near" 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.

# 4 "near" Application Startup Confirmation Messages

## Before Starting the "near" Application

Prior to starting the "near" application using "near" utility, it is recommended to confirm with the user that the game will be interrupted and the "near" application will start for the purpose of usability enhancement. The recommended messages to be displayed at that time are listed below.

### Messages for Sending the Gift Information

The "near" application must be started in order to send the set gift. Before the "near" application starts, it is recommended that a message such as the following is displayed to have the user confirm that the "near" application will start.

| language | Message |
|---|---|
| Japanese | サーバーにアイテムを送信するために"near"を起動しますか？ |
| English | Would you like to launch "near" to upload a Game Good to the server? |
| English UK | Would you like to launch "near" to upload a Game Good to the server? |
| French | Voulez-vous lancer "near" pour envoyer un élément de jeu sur le serveur ? |
| Spanish | ¿Quieres ejecutar "near" para cargar un Objeto de juego al servidor? |
| German | Möchtest du "near" starten, um einen Spielgegenstand auf den Server hochzuladen? |
| Italian | Vuoi avviare "near" per caricare un oggetto di gioco sul server? |
| Dutch | Wil je "near" opstarten om game-goodies te uploaden naar de server? |
| Portuguese | Queres iniciar o "near" para transferir um Item de Jogo para o servidor? |
| Portuguese_BR | Você deseja abrir o "near" para fazer o upload de um produto para o servidor? |
| Russian | В ы хотели бы запустить приложение "near" для подгрузки игрового товара на сервер? |
| Polish | Chcesz uruchomić "near", aby przesłać element gry na serwer? |
| Finnish | Haluatko käynnistää "near" -sovelluksen, jotta voit kopioida pelitavaran palvelimeen? |
| Danish | Vil du starte "near" for at lægge en spilgenstand op på serveren? |
| Norwegian | Har du lyst til å kjøre "near" for å laste opp en spillgodsak til serveren? |
| Swedish | Vill du starta "near" för att ladda upp spelgods till din server? |
| Turkish | Sunucuya bir Oyun Eşyası yüklemek için "near"ı çalıştırmak ister misin? |
| Korean | 서버에 아이템을 전송하기 위해 "near"를 기동하시겠습니까? |
| SimplifiedChinese | 要启动"near"将道具传送至服务器吗？ |
| TraditionalChinese | 要啟動"near", 以將道具傳送至伺服器嗎？ |

### Messages for Receiving a Gift

The "near" application must be started in order to receive the gift. Before the "near" application starts, it is recommended that a message such as the following is displayed to have the user confirm that the "near" application will start.

| language | Message |
|---|---|
| Japanese | サーバーからアイテムを受信するために"near"を起動しますか？ |
| English | Would you like to launch "near" to download a Game Good from the server? |
| English UK | Would you like to launch "near" to download a Game Good from the server? |
| French | Voulez-vous lancer "near" pour télécharger un élément de jeu du serveur ? |
| Spanish | ¿Quieres ejecutar "near" para descargar un Objeto de juego del servidor? |
| German | Möchtest du "near" starten, um einen Spielgegenstand vom Server herunterzuladen? |
| Italian | Vuoi avviare "near" per scaricare un oggetto di gioco dal server? |
| Dutch | Wil je "near" opstarten om game-goodies te downloaden van de server? |
| Portuguese | Queres iniciar o "near" para transferir um Item de Jogo do servidor? |
| Portuguese_BR | Você deseja abrir o "near" para fazer o download de um produto para o servidor? |
| Russian | В ы хотели бы запустить приложение "near" для загрузки игрового товара с сервера? |
| Polish | Chcesz uruchomić "near", aby pobrać element gry z serwera? |
| Finnish | Haluatko käynnistää "near" -sovelluksen, jotta voit ladata pelitavaran palvelimesta? |
| Danish | Vil du starte "near" for at hente en spilgenstand fra serveren? |
| Norwegian | Har du lyst til å kjøre "near" for å laste ned en spillgodsak fra serveren? |
| Swedish | Vill du starta "near" för att ladda ned spelgods från servern? |
| Turkish | Sunucudan bir Oyun Eşyası indirmek için "near"ı çalıştırmak ister misin? |
| Korean | 서버에서 아이템을 수신하기 위해 "near"를 기동하시겠습니까? |
| SimplifiedChinese | 要启动"near"从服务器接收道具吗？ |
| TraditionalChinese | 要啟動"near", 以從伺服器接收道具嗎？ |