NP Message Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Scope of This Document

Game boot messages are a feature allowing users who receive the message to start up an application by opening data attachments. There are 2 types of game boot message: messages to which data to invite friends and other users to the game is attached, and messages to which data created by an application is attached.

This document describes the APIs available for use as application features, and the feature for sending/receiving game boot messages using Common Dialog.

Purpose and Characteristics

The NP Message Dialog library is used to transmit and receive game boot messages using PSN℠. The NP Message library supports the receiving processing of game boot messages. As one of the features of the Common Dialog library, the sending/receiving of messages is possible using a system GUI.

The NP Message library is a library for supporting receiving processing of game boot messages; it allows obtaining the number of messages held by the user without using the system's GUI, and implementing receiving processing of game boot messages employing the application's original UI.

> **Note**
> In addition to game boot messages, there is also a feature for sending/receiving messages with store link attachments. By using messages with store link attachments, invitations to Title Stores are possible. If you are planning to use messages with store link attachments, please contact SCE in advance to obtain approval.

NP Message Dialog Library

The NP Message Dialog library is used to transmit and receive game boot messages using PSN℠. Overlay display of the message transmission dialog, list display/selection dialog for received messages, and detailed display dialog for received messages on the screens of an application is possible by calling APIs of the NP Message Dialog library with the appropriate parameters.

The NP Message Dialog library is one of the Common Dialog library features, and GUI display and user operation handling are its main processing. The main utilization flow consists in first specifying contents to display, monitoring the closing of the dialog through polling, and last obtaining the call results.

Main Features

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

- Feature to transmit and receive messages
- Feature to display a list of received messages
- Feature to display the details of received messages
- Feature to obtain the data of received message

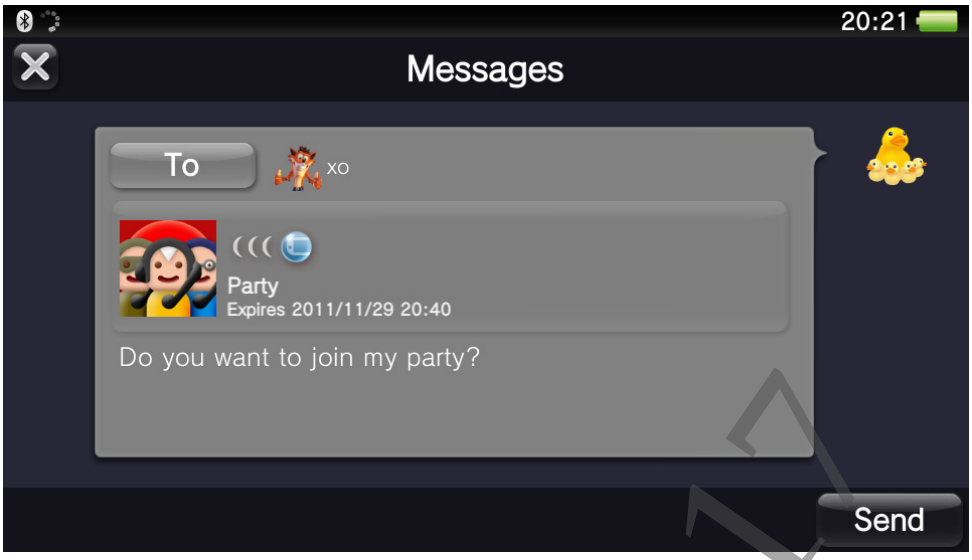
Embedding into a Program

Include np_message_dialog.h in the source program (various header files including np/np_message.h will be automatically included as well).

The PRX module need not be loaded.

Upon building the program, link libSceCommonDialog_stub.a.

Figure 1 NP Message Dialog Example



NP Message Library

Like the NP Message Dialog library, the NP Message library utilizes PSN℠ to support the receiving of game boot messages.

Since by using this library, a list of messages, a message, or attached data can be obtained, you can create original application UIs.

Main Features

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

- Feature to synchronize messages with PSN℠
- Feature to obtain the number of received NP messages
- Feature to obtain a list of received messages
- Feature to obtain a message and attached data

Used Resources

The NP Message library uses the following system resources.

Resource	Description
Footprint	Approx. 1.1 MiB upon loading the PRX
Work memory	Approx. 380 KiB upon initialization

Embedding into a Program

Include np.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_NP_MESSAGE) != SCE_OK ) {
    // Error handling
}
```

Upon building the program, link libSceNpMessage_stub.a.

Sample Programs

Sample programs are as follows.

sample_code/system/api_np_message_dialog/

This sample uses NP Message Dialog library to transmit messages and display the list of received messages.

sample_code/network/api_np/np_message/

This sample uses the NP Message library and obtains a list of received messages and attached data.

Reference Materials

For the common limitations, specifications, etc., of the Common Dialog library, refer to the following document.

- Common Dialog Overview

For a general description of PSN℠, refer to the following document.

- PSN℠ Overview

For common NP libraries required to use PSN℠ features, refer to the following documents.

- NP Library Overview
- NP Library Reference

Regarding the network check dialog - which switches the service state of the NP library - refer to the following document.

- Network Overview

For the system software related to the NP Message library, refer to the following document.

- System Software Overview

2 Sending/Receiving Game Boot Messages

The game boot messages that can be sent/received by using the NP Message Dialog library or the NP Message library are classified into 2 types depending on the data attached:

Game Boot Messages (Messages with Invitation Data Attachments)

This message type is for inviting friends or other users to an online game using an NP matching room, etc. Information required for participation in the online game, such as a lobby or matching room ID, can be transmitted and received, attached to a text message.

Game Boot Messages (Messages with Custom Data Attachments)

This message type is for transmitting and receiving content and format data freely defined by the application, attached to a text message.

> **Note**
> All R3093 regulations in the TRC (Technical Requirements Checklist) relating to game boot messages can be satisfied by using only NP Message Dialog.
> If you wish to create a message list display screen or a message obtaining processing screen on the application side, you can do so by using NP Message Dialog and the NP Message library together.

Sending/Receiving Game Boot Messages with NP Message Dialog

Below is an explanation of how to send/receive game boot messages using only NP Message Dialog's APIs.

How to Send Messages

Call NP Message Dialog in "message transmission mode". Users can send game boot messages by tapping the **Send** button on the dialog that is displayed.

How to List Messages

By calling NP Message Dialog in "received message list display mode" at an arbitrary time, it is possible to display a list of the game boot messages received.

How to Obtain Message Data

If an application event (invitation event, custom data event) occurring when the user opens a message data attachment is received, the contents of the game boot message can be obtained by calling NP Message Dialog in "message data obtaining mode".

Receiving Game Boot Messages With an Original Application UI

Below is an outline of how to achieve the receiving processing of game boot message with an original application UI by combining NP Message Dialog and the NP Message library's APIs.

However, the NP Message library does not have a feature for sending messages. Use NP Message Dialog library for sending messages.

How to List Messages

You can display a list of the game boot messages received by calling NP Message library's API for obtaining message list information at an arbitrary time, and displaying the results of obtainment on the application's original UI.

How to Obtain Message Data

If an application event (invitation event, custom data event) occurring when the user opens a message data attachment is received, the contents of the game boot message can be obtained by calling the NP Message library's "Message Obtainment".

Receiving Game Boot Message by the Messages Application

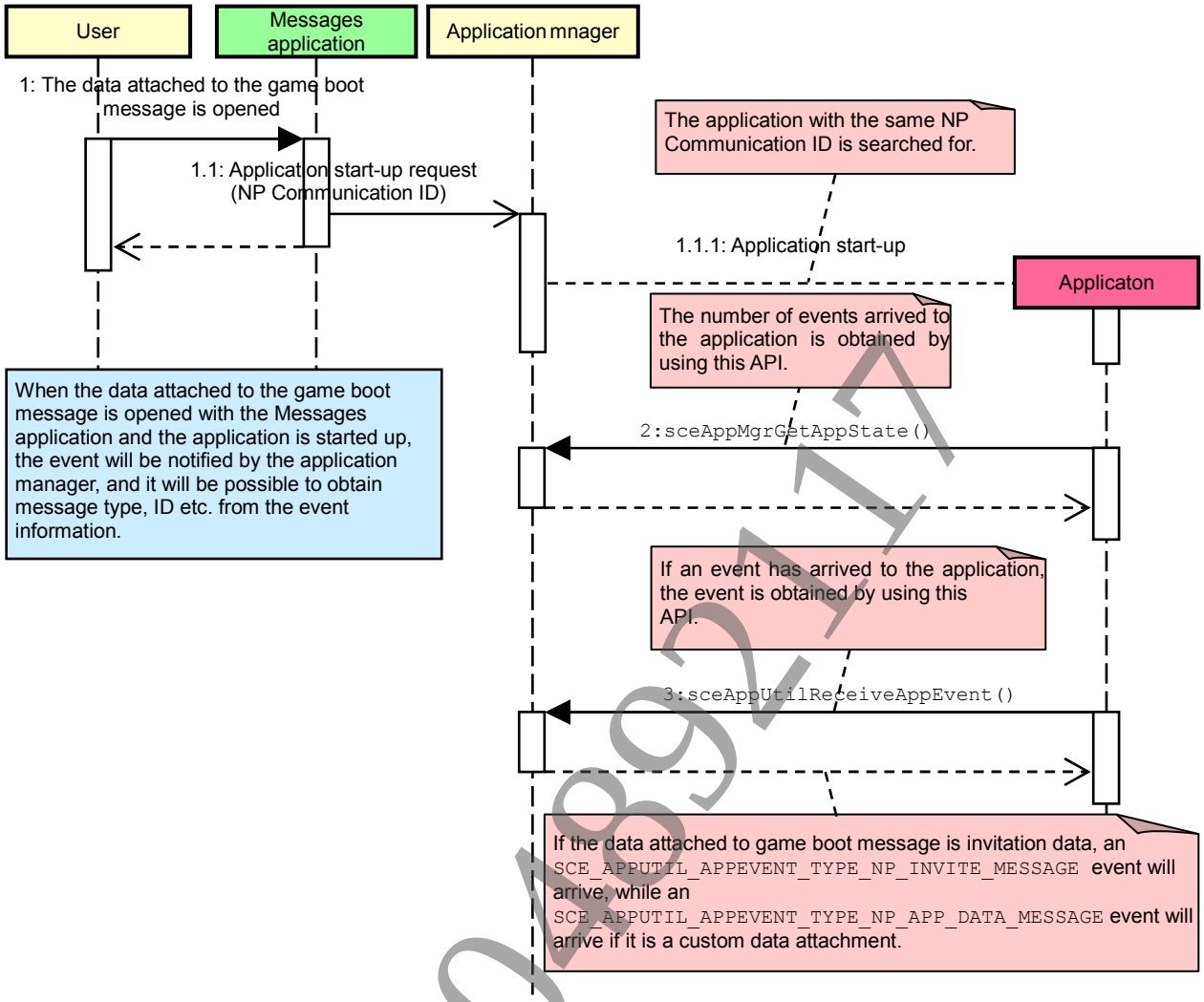
Messages application, one of the applications included in the system software, can handle messages relating to Sony Entertainment Network accounts. Of the received game boot messages, messages with invitation data attachments can be checked from **Invitations** and messages with custom data attachments can be checked from **Game Alerts**. When a user opens a game boot message with the Messages application, the corresponding application will be started up (if the application is already running, it will be resumed).

When the data attachment of a game boot message is opened and the application is started up, the following application events will be notified by the application manager, and it will be possible to obtain the ID of the messages selected by the user from the event information.

- Game boot messages (messages with invitation data attachments)
  `SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE`
- Game boot messages (messages with custom data attachments)
  `SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE`

For the event handling, refer to the "Starting Up an Application from the Messages Application" section in Chapter 3 and 4.

Figure 2 Receiving a Game Boot Message

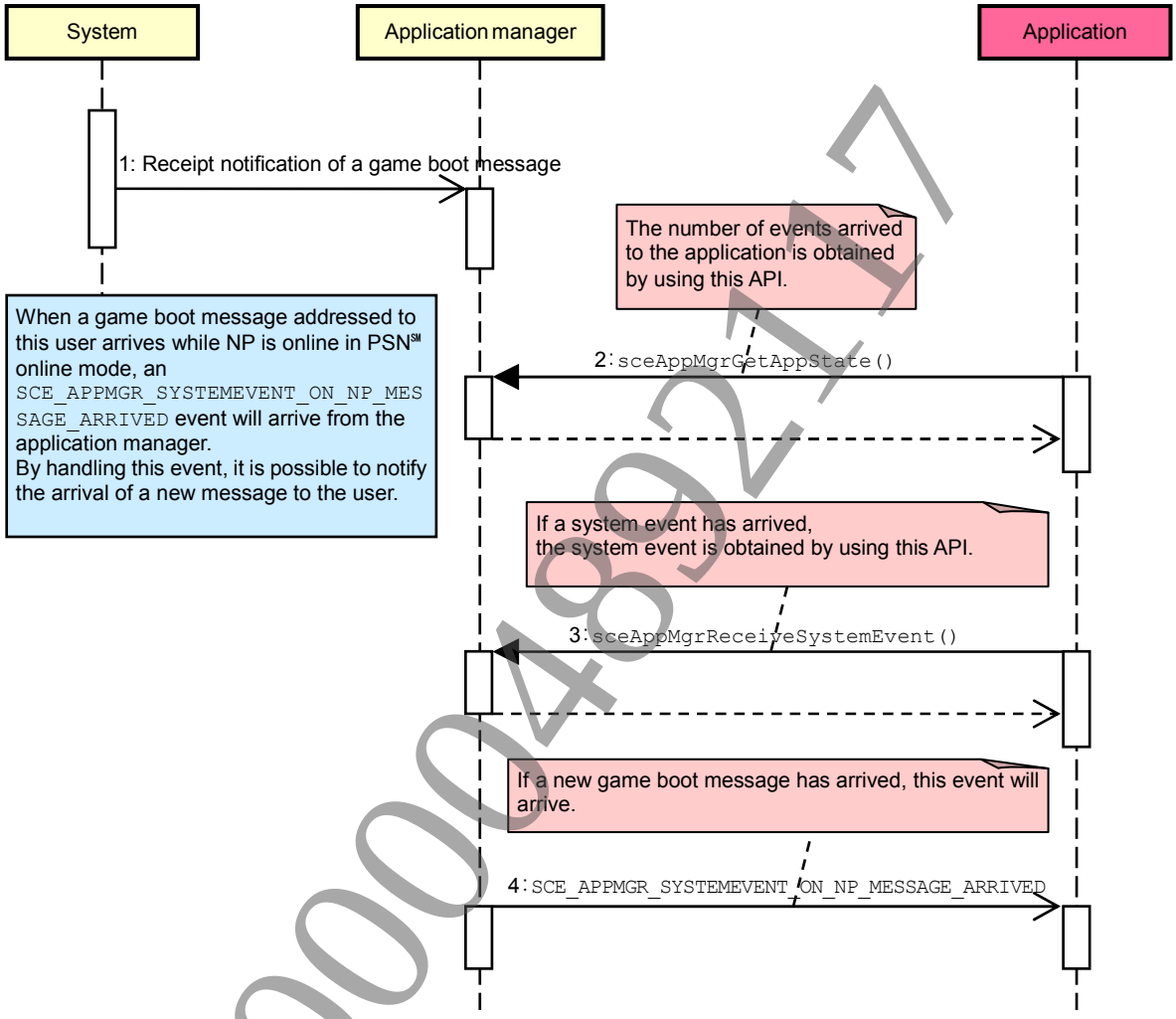


Note
The NP Communication ID is recorded in game boot messages, and matching with the application is performed with the NP Communication ID.
The Messages application, the NP Message Dialog library and the NP Message library handle the same game boot messages. Game boot messages that are sent with the NP Message Dialog library can be referenced with the Messages application. Furthermore, deleting the game boot messages of relevant games with the Messages application will also make it impossible to reference them from the NP Message Dialog library and the NP Message library.

Handling of New Message Events

When a game boot message addressed to a user arrives while in NP online state in the PSN℠ online mode, an SCE_APPMGR_SYSTEMEVENT_ON_NP_MESSAGE_ARRIVED event will arrive from the application manager. By handling this event, it will be possible to implement features such as notifying the arrival of new messages to the user and displaying a dialog listing received messages.

Figure 3 Event Handling



Note
New message events are received only when they are sent while both the sending side and the receiving side are in the NP online state in the PSN℠ online mode.
This event means that a game boot message addressed to the user has arrived (in some cases, the game boot message will be addressed to another application).

Following transmission, several tens of seconds may elapse before that event arrives to the receiving application.
If a dialog is open in the received message list display mode for game boot messages, the latest invitation messages will be included automatically; therefore, this event can be ignored.

3 Using NP Message Dialog

Operation Modes of NP Message Dialog

The NP Message Dialog library has the following operation modes.

Operation mode	Description
SCE_NP_MESSAGE_DIALOG_MODE_MESSAGE_SEND	Message transmission mode
SCE_NP_MESSAGE_DIALOG_MODE_MESSAGE_LIST	Received message list display mode
SCE_NP_MESSAGE_DIALOG_MODE_MESSAGE_DETAIL	Received message details display mode
SCE_NP_MESSAGE_DIALOG_MODE_MESSAGE_GET	Message data obtaining mode

Message Transmission Mode

This mode displays the dialogs for message transmission. The application can specify the NP ID (SceNpId) list designating the recipient, the body, attached data, index icon, and message expiration date.

The user can make additional inputs in a range consisting of a given number of characters in the body specified by the application, in the displayed dialog. If no recipient is specified by the application, the user can specify the recipient.

Note
When sending messages with store link attachments, the message cannot be sent if the application or user only specifies the sender of the message alone as the recipient of the message. Messages can be sent when another recipient besides the transmission origin user is input.
This limitation does not apply for messages with invitation data attachments and messages with custom data attachments.

The transmission processing is actually executed when the user presses the **Send** button. The transmitted message is saved on the server of PSN℠. The processing result can be obtained by having the application call sceNpMessageDialogGetResult() after the dialog has been closed by the user.

Result Code	Description
SCE_NP_MESSAGE_DIALOG_USER_ACTION_CANCEL	The transmission process was canceled by the user.
SCE_NP_MESSAGE_DIALOG_USER_ACTION_MESSAGE_SEND_OK	The transmission was successful.
SCE_NP_MESSAGE_DIALOG_USER_ACTION_MESSAGE_SEND_NG	The message was not transmitted because the transmission failed.

Transmitted messages can be checked by the application on the receiver's side calling NP Message Dialog in the "received message list display mode". Transmitted messages can also be checked by starting the Messages application incorporated in the system software.

Received Message List Display Mode

This is the mode for displaying the list of received messages. When this mode is started up, it may take some time until the dialog is displayed because of the processing to receive the messages stored on the server of PSN℠.

The list displays the sender, body, index icon, etc. of messages of the message type specified by the application.

The user can select any of the displayed messages and open the data attached to that message.

After the user closes the dialog, the application can obtain the processing results by calling `sceNpMessageDialogGetResult()`.

Result Code	Description
`SCE_NP_MESSAGE_DIALOG_USER_ACTION_CANCEL`	List display was canceled by the user.
`SCE_NP_MESSAGE_DIALOG_USER_ACTION_OPEN_ATTACHED_DATA`	Data attached to the message was opened by the user. The `SceNpMessageEntry` structure, which has the information of the opened message, is included in the `SceNpMessageDialogResult` structure.

Received Message Details Display Mode

This mode displays the details of one message. Events are issued by the application manager based on the application's start-up request from the Messages application. When the application specifies the message ID included in that event, the sender of the message, the body of the message, etc. are displayed.

The user can open the data attached to the message.

After the user closes the dialog, the application can obtain the processing results by calling `sceNpMessageDialogGetResult()`.

Result Code	Description
`SCE_NP_MESSAGE_DIALOG_USER_ACTION_CANCEL`	List display was canceled by the user.
`SCE_NP_MESSAGE_DIALOG_USER_ACTION_OPEN_ATTACHED_DATA`	Data attached to the message was opened by the user. The `SceNpMessageEntry` structure, which has the information of the opened message, is included in the `SceNpMessageDialogResult` structure.

Message Data Obtaining Mode

This is the mode for obtaining message data. An event is issued by the application manager based on the application start-up request from the Messages application. When the application specifies the message ID included in that event, it will be possible to obtain that message's body and attached data.

After the dialog is closed, the application can obtain the processing results by calling `sceNpMessageDialogGetResult()`.

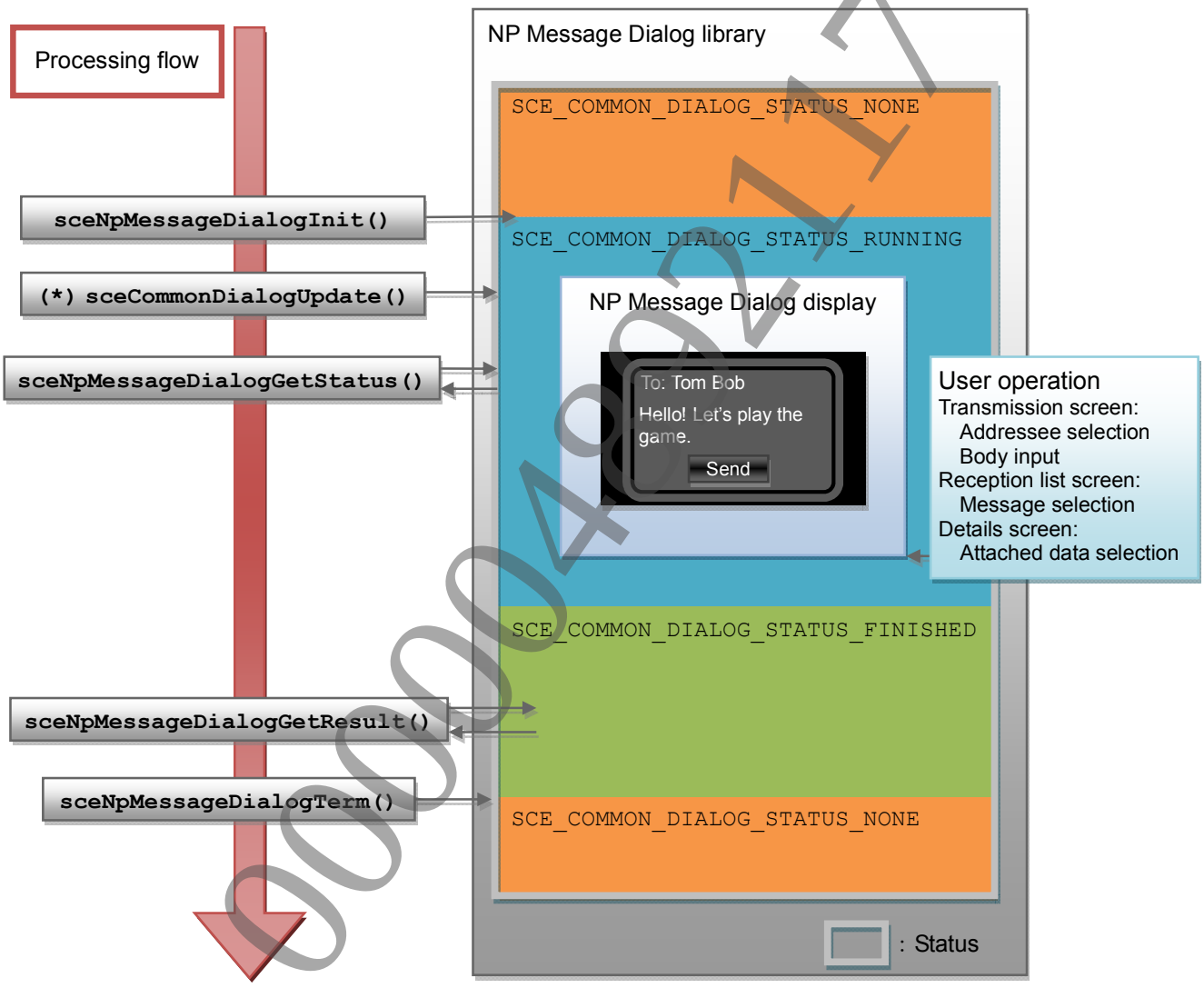
Result Code	Description
`SCE_NP_MESSAGE_DIALOG_USER_ACTION_MESSAGE_GET_OK`	Message obtainment successful
`SCE_NP_MESSAGE_DIALOG_USER_ACTION_MESSAGE_GET_NG`	Message obtainment failed

Basic Usage Procedure

The basic procedure to use the NP Message Dialog library is described below. The processing flow is outlined below.

(1) Set the parameters to the variables of the `SceNpMessageDialogParam` type.
(2) Initialize the library (call NP Message Dialog)
(3) Wait until NP Message Dialog is closed by polling the operation status.
(4) Obtain the call results.
(5) End processing.

Figure 4 Basic Processing Procedure



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

Libraries to Be Loaded and Initialized Beforehand

In order to use the NP Message Dialog library, the dependent libraries must be loaded and initialized by calling the following APIs beforehand.

- `sceSysmoduleLoadModule(SCE_SYSMODULE_NET);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_HTTPS);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_NP);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_NP_BASIC);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_NP_UTILITY);`
- `sceNetInit();`
- `sceNetCtlInit();`
- `sceSslInit();`
- `sceHttpInit();`
- `sceNpInit();`
- `sceNpLookupInit();`

NP Message Dialog Call Procedure

(1) Setting the parameters

First, prepare the `SceNpMessageDialogParam` type variable and following initialization with `sceNpMessageDialogParamInit()`, set the operation mode (*`mode`*), message type (*`type`*), and the parameters that are required accordingly.

In the message transmission mode, provide the `SceNpMessageDialogMessageSendParam` type variable and fill them with 0s before setting the recipient's NP ID, the body, data to be attached, etc.

In the case of the received message list display mode, provide the `SceNpMessageDialogMessageListParam` type variable and fill them with 0s.

In the received message details display mode, provide the `SceNpMessageDialogMessageDetailParam` type variable and fill them with 0s before setting the message IDs to be displayed.

In the case of message data obtaining mode, prepare an `SceNpMessageDialogMessageGetParam` type variable and fill it with 0, then set the message ID whose data you wish to obtain and the buffers for storing body and attached data, etc.

(2) Initializing the library

Call `sceNpMessageDialogInit()` to perform initialization. Specify the `SceNpMessageDialogParam` type variable set beforehand as the argument.

(3) Obtaining the operation status

Call `sceNpMessageDialogGetStatus()` at every frame to poll the operation status of NP Message Dialog. `SCE_COMMON_DIALOG_STATUS_RUNNING` is returned as the operation status while NP Message Dialog is displayed; wait until `SCE_COMMON_DIALOG_STATUS_FINISHED` is returned.

> **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.

(4) Obtaining the call results

When NP Message Dialog closes and the operation status changes to `SCE_COMMON_DIALOG_STATUS_FINISHED` as a result, the results can be obtained with `sceNpMessageDialogGetResult()`. The information included in the obtainable results consists of the operation mode (*`mode`*) and message type (*`type`*) specified during the calling procedure and whether the user OK'ed or canceled; in the case of message transmission, whether the transmission was successful; and if the attached data was opened, the corresponding message ID.

(5) Terminating the processing

Once the call results have been obtained, call `sceNpMessageDialogTerm()` to terminate the processing. The resources allocated during initialization will then be released.

Aborting the Processing

To abort the display of NP Message Dialog from the application side on an emergency basis, for example, when an application is quit, call `sceNpMessageDialogAbort()`. Display will quickly be terminated, and operation status will change to `SCE_COMMON_DIALOG_STATUS_FINISHED`. In this case, too, the call result can be obtained with `sceNpMessageDialogGetResult()`. `SCE_COMMON_DIALOG_RESULT_ABORTED` is returned as obtained result.

Main APIs and Structures Used for Basic Processing

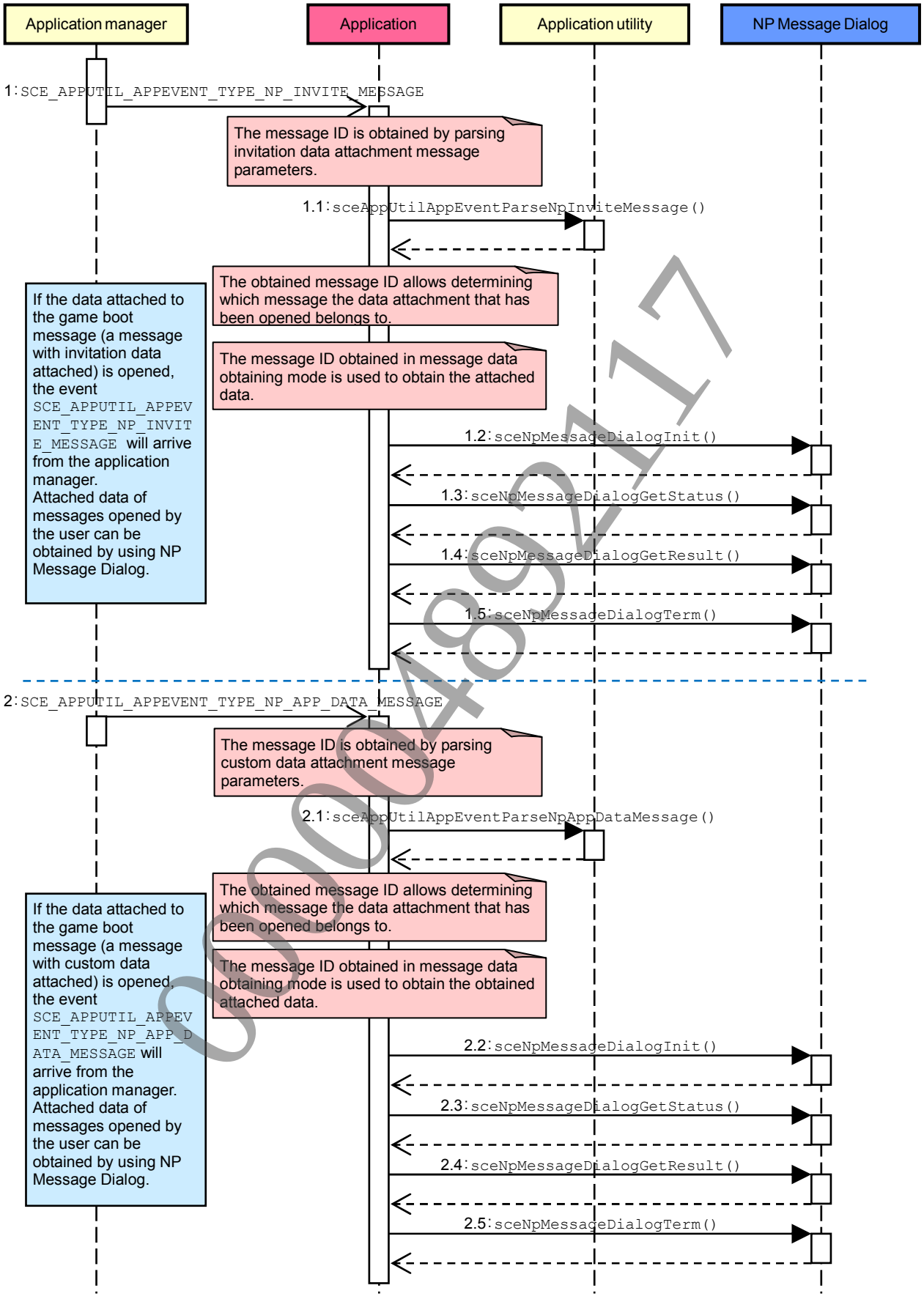
API	Description
`SceNpMessageDialogParam`	Parameter structure including mode setting
`sceNpMessageDialogParamInit()`	Initializes parameter structure
`sceNpMessageDialogInit()`	Initializes the library
`sceNpMessageDialogGetStatus()`	Obtains operation status
`sceNpMessageDialogGetResult()`	Obtains call results
`sceNpMessageDialogTerm()`	Terminates the library
`sceNpMessageDialogAbort()`	Aborts calling of feature

Starting Up an Application from the Messages Application

When the attached data of a game boot message is opened with the Messages application, the application will be started up. In this case, after start-up `SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE` or `SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE` event will arrive from the application manager. Therefore, parse the message parameters using `sceAppUtilAppEventParseNpInviteMessage()` or `sceAppUtilAppEventParseNpAppDataMessage()`, and obtain the message ID.

It is possible to obtain attached data by opening Common Dialog in message data obtaining mode using the obtained message ID.

Figure 5 Starting Up Applications and Obtaining Attached Data Using NP Message Dialog for Reception



4 Using the NP Message Library

Libraries to Be Loaded and Initialized Beforehand

In order to use the NP Message library, the following APIs must be called in advance to load (and initialize) each NET, HTTP, NP, and NP_Utility module. For details, refer to each applicable library document.

- `sceSysmoduleLoadModule(SCE_SYSMODULE_NET);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_HTTPS);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_NP);`
- `sceSysmoduleLoadModule(SCE_SYSMODULE_NP_UTILITY);`

The following initialization functions must be called in advance.

- `sceNetInit();`
- `sceNetCtlInit();`
- `sceSslInit();`
- `sceHttpInit();`
- `sceNpInit();`
- `sceNpLookupInit();`

Initialization

(1) Loading the PRX

Call `sceSysmoduleLoadModule()` with `SCE_SYSMODULE_NP_MESSAGE` specified as the module ID and load the NP PRX module.

(2) Initializing the library

Next, initialize the NP Message library using `sceNpMessageInit()`.

```
ret = sceNpMessageInit (
              SCE_NP_MESSAGE_TYPE_INVITATION_ATTACHMENT
              );
if (ret < 0) {
        // Error handling
}
```

When `SCE_NP_MESSAGE_TYPE_INVITATION_ATTACHMENT` is set to the first argument, *`messageType`*, the library will be able to handle game boot messages (messages with invitation data attachments).

When `SCE_NP_MESSAGE_TYPE_APP_DATA_ATTACHMENT` is set, the library will be able to handle game boot messages (messages with custom data attachments).

If initialization was performed using the `sceNpMessageInit()` function, only those messages whose attached data has not been used and which are within the expiration date are included in the message list acquisition results. To acquire also messages with attached data that has already been used or messages whose expiration date has expired, use `sceNpMessageInitWithParam()`.

Communication Processing

(1) Synchronizing the server with messages

When you want to download new messages that just arrived on the server of PSN℠ to PlayStation®Vita, call `sceNpMessageSyncMessage()` and synchronize messages accumulated on the server of PSN℠ with messages stored on PlayStation®Vita.

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

Note
Call `sceNpMessageSyncMessage()` based on user operations, such as the timing of the beginning of screen display and on the timing in which on-screen **Update** buttons are pressed. Avoid performing automatic, frequent calling instead of user operation, as it would place a significant burden on the server.

Obtaining a List of Messages

(1) Obtaining number of messages

Obtain the number of NP messages stored on PlayStation®Vita and the number of unread messages.

```
SceInt32 totalCount,
SceInt32 newCount

ret = sceNpMessageGetMessageEntryCount (&totalCount, &newCount);
if (ret < 0) {
        // Error handling
}
```

The messages that will be included in the above number are the messages with the message type specified upon executing `sceNpMessageInit()`, out of all the messages addressed to the application.

(2) Obtaining a list of messages

`sceNpMessageGetMessageEntries()` can be called to efficiently obtain a list information of NP messages held by PlayStation®Vita. As arguments, specify the offset of the messages you want to obtain a list of, and the limit value as the maximum number of NP messages you want to obtain. The most recently received message corresponds to offset 0. For offset value, set the number of messages after which list information is to be obtained. The number of NP messages actually obtained will be stored in *storedCount*. Information of each message on the list will be stored in *entries*. *entries* will store message IDs, the sent dates of messages, the NP IDs of message senders, and previews (beginning sections) of actual messages. These can be used for the application to create a display of NP messages.

```
SceInt32 offset;
SceInt32 limit;
SceNpMessageEntry *entries;
SceInt32 *storedCount;

ret = sceNpMessageGetMessageEntries(
        offset,      // Offset
        limit,       // Limit
        *entries,    // List of messages
        *storedCount// Number of messages actually obtained
        );
if (ret < 0) {
        // Error handling
}
```

Obtaining a Message

(1) Obtaining the body of a message

Specify the `SceNpMessageId` of the desired message and obtain the actual message body.

The message will be stored in the UTF-8 format.

```
// Process to obtain a message body of up to 256 bytes in the buffer
SceChar8 buffer[256];
SceSize storedSize;
ret = sceNpMessageGetMessage(messageId, buffer, sizeof(buffer), & storedSize);
if (ret < 0) {
        // Error handling
}
// The actual obtained data size will be stored in storedSize
```

(2) Obtaining attached data

Specify the `SceNpMessageId` of the desired message and obtain data attached to the message.

```
// Process to store attached data of up to 1024 bytes in the dataBuffer
SceChar8 dataBuffer[1024];
SceSize storedSize;
ret = sceNpMessageGetMessage(messageId, dataBuffer, sizeof(buffer),
                                                    & storedSize);
if (ret < 0) {
        // Error handling
}
// The actual obtained data size will be stored in storedSize
```

(3) Set attached data as used

Set attached data that has been used once as used data. Once set to used data, it will no longer be a target for extraction by NP Message Dialog, the NP Message library and the Messages application.

```
ret = sceNpMessageSetAttachedDataUsedFlag(messageId);
if (ret < 0) {
        // Error handling
}
```

Termination Processing

Terminate the NP Message library

Terminate the NP Message library by calling `sceNpMessageTerm()`.

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

Other Termination Processing

The libraries loaded and initialized in advance should also be terminated when they become unnecessary.

Aborting the Processing

Aborting the NP Message Library

Abort the NP Message library with `sceNpMessageAbort()`.

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

Note
Abort processing and call the NP Message library's termination processing internally.
If continuing processing, start again from initialization processing.

Starting Up an Application from the Messages Application

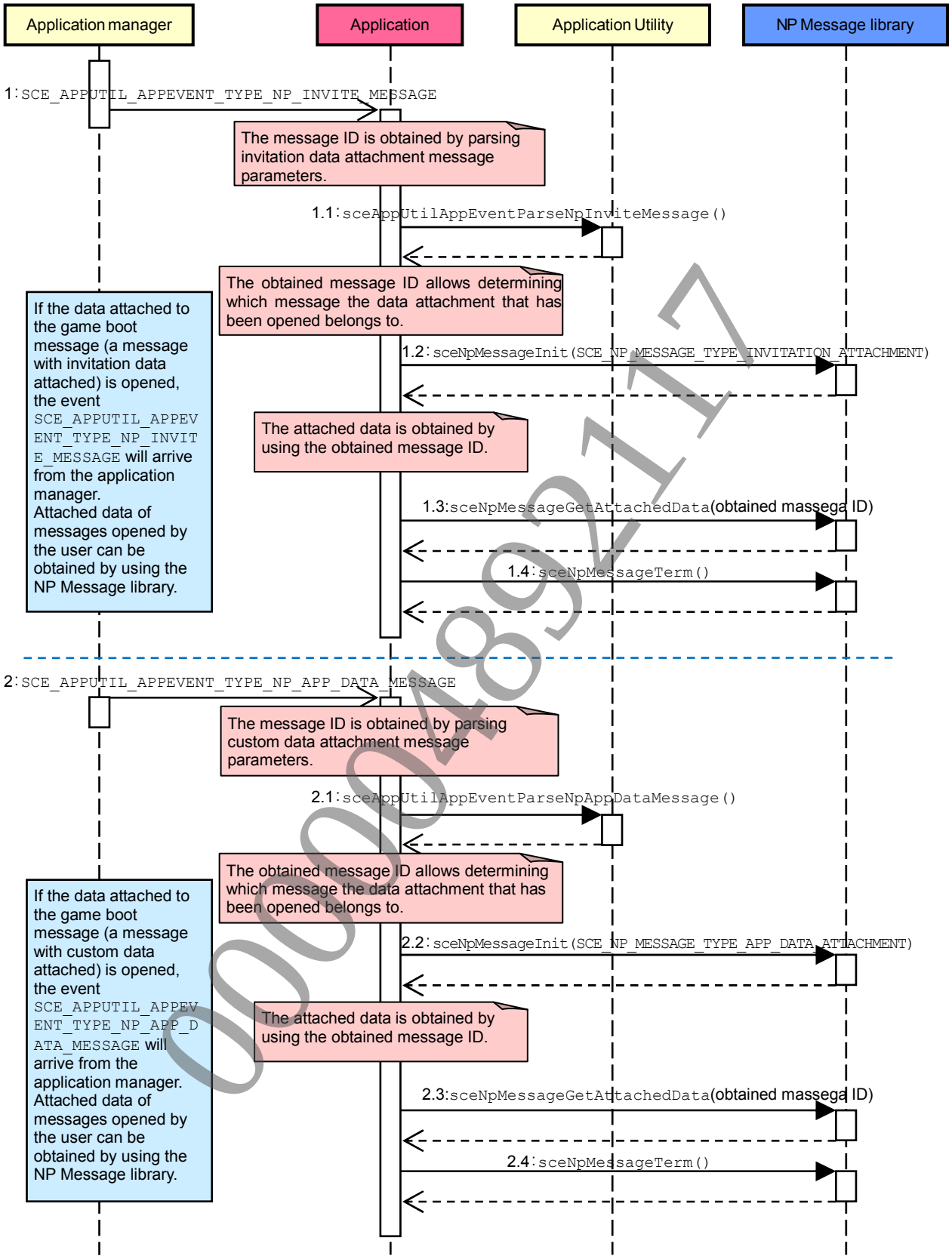
When the attached data of a game boot message is opened with the Messages application, the application will be started up. In this case, after start-up `SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE` or `SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE` event will arrive from the application manager. Therefore, parse the message parameters using `sceAppUtilAppEventParseNpInviteMessage()` or `sceAppUtilAppEventParseNpAppDataMessage()`, and obtain the message ID.

It is possible to obtain attached data with `sceNpMessageGetAttachedData()` using the obtained message ID.

The following sample program using this event is provided for reference purposes.

- sample_code/network/api_np/np_message/event_handling/

Figure 6 Starting Up Applications and Obtaining Attached Data Using NP Message Library for Reception



5 When Beginning Development

NP Communication ID Obtainment

The NP Message library requires NP Communication IDs in order to determine the titles that are started up with game boot messages.

An NP Communication ID can be requested for, and issued, per application via the PlayStation®Vita Developer Network (https://psvita.scedev.net/)

NP Communication ID Setting

When sending a game boot message, TITLE_ID and NP_COMMUNICATION_ID information in param.sfo will also be sent; therefore, it is necessary to set an NP Communication ID to the param file (param.sfo).

Enable "Game boot message" under Param File Editor in Publishing Tools, and specify the NP Communication ID in the "NP Communication ID" setting field. For details, refer to the "Param File Editor User's Guide" document contained in the Publishing Tools.

In order to check whether the NP Communication ID has been set correctly to the param file, turn **ON** the **★Debug Settings -> PSN℠ -> NP Debug** setting in the Development Kit's or Testing Kit's Settings application to display the NP Communication ID specified by the application.

When the following APIs are called, the NP Communication ID specified by the application will be displayed in a system message.

Library	Function
NP Message Dialog library	`sceNpMessageDialogInit()`
NP Message library	`sceNpMessageInit()`
NP Message library	`sceNpMessageInitWithParam()`

Note
If NP Communication ID in the param.sfo is not set during development, etc., the NP Communication ID specified by the NP library's initialization function `sceNpInit()` will be used instead.
Note, however, that if **Release Check Mode** of the Development Kit is **Release Mode**, or if the Testing Kit is used for development, an error will occur in the following APIs and master completion will not be possible if no NP Communication ID has been specified.
`sceNpMessageDialogInit()`, `sceNpMessageInit()`, `sceNpMessageInitWithParam()`

How to Start up an Application from a Game Boot Message

By using the Messages application or the NP Message Dialog library to open the data attachment of a game boot message, it is possible to start up the specified target application (if it is not already running) and to participate in specific features of the application.

Since the start-up target application will be searched for in the order below, it will be necessary to have one of the following ready:

(1) An application that is already running (including applications running with the Debugger)
(2) An application already installed on the PlayStation®Vita as a pkg file
(3) An application available for startup from ★APP_HOME on the home screen

6 Notes

Limitations

- NP Message Dialog library and NP message library cannot be used simultaneously. Make sure that one of the libraries is in the terminated state before using the other.
- Functions of the NP Message library are not multithread safe. In addition, all processing are synchronized. Thus, application rendering will be affected when executing functions of the NP Message library on the main thread. Use sub threads to execute functions.
- When sending a message using the NP Message Dialog library, always specify data to be attached (extended data). When not specified, an error dialog will be displayed upon sending the message and transmission will fail.

Size Limitations

The following size limitations apply to the messages that can be sent with NP Message Dialog library.

Item	Size Limitation
Recipients	Up to 16 persons
Message body	Up to 512 bytes including terminating NULL character (UTF-8)
Attached data size	Up to 1 MiB
Index icon	Pixel size: Up to 128 x 128 pixels File size: Up to 64 KiB Format: PNG or JPEG

Recipient Limitations

When sending messages with store link attachments, the message cannot be sent if the application or user specifies the sender alone as the recipient of the message. Messages can be sent when another recipient besides the original message sender is input.

This limitation is not applicable for messages with invitation data attachments and messages with custom data attachments.

Conditions for Use

To use NP Message Dialog library in the message transmission mode, Network Check Dialog must be started in PSN℠ online mode in advance and it is required to transition to the NP online state.

For details, refer to the "Network Overview" document.