Save Data Dialog Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 Library Overview

Purpose and Features

The Save Data Dialog library is a library that supports the realization of GUI display for the flow consisting of saving and loading of save data by applications. Through the use of the Save Data Dialog library, applications can easily implement guidance for users that complies with TRC (Technical Requirements Checklist).

The Save Data Dialog library is one of the Common Dialog library functions, and encapsulates the GUI display and user operation handling. The main utilization flow consists in first specifying and calling the display contents, monitoring the closing of the dialog through polling, and last retrieving the call results.

Main Functions

The following are the main functions provided by the Save Data Dialog library.

- Function for fixed display of save data
- Function for list display of save data
- Function to display user-specified messages
- Function to display system defined messages
- Function to display error codes
- Function to display progress bar

Embedding in Program

Include savedata_dialog.h in the source program. (Additionally, a number of header files are automatically included.)

The PRX module need not be loaded.

When building programs, link libSceCommonDialog_stub.a.

Sample Programs

The following file is provided as a sample program that uses the Save Data Dialog library for reference purposes.

sample_code/system/api_savedata/fixed_basic/

sample_code/system/api_savedata/list_basic/

This sample realizes the save data fixed display type and list display type save/load flow using the Save Data Dialog functions.

Reference Materials

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

- Common Dialog Overview

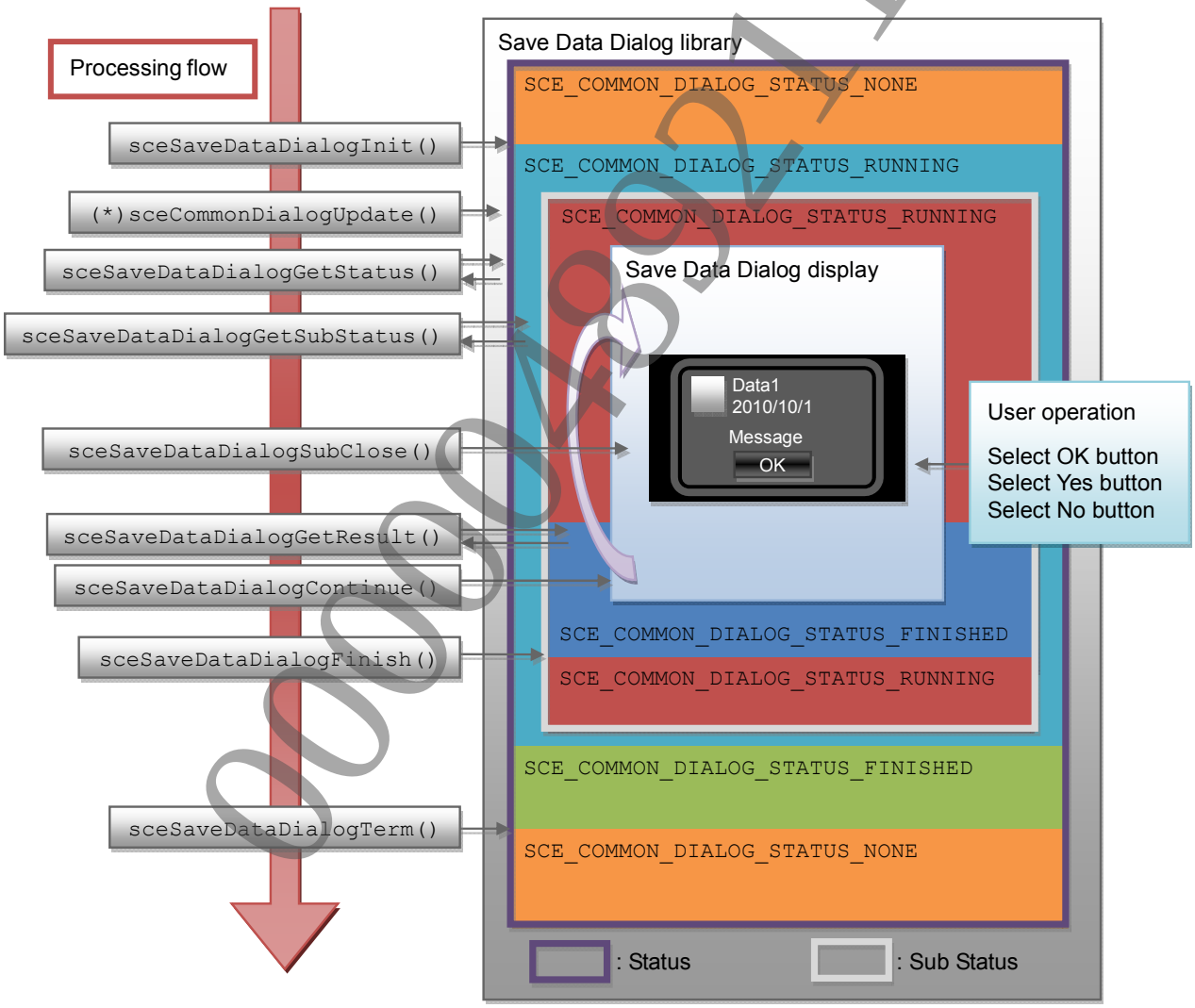
2 Usage Procedure

Basic Usage Procedure

The basic procedure to call the Save Data Dialog library is described below. The processing flow is outlined below.

(1) Set the parameters to the variables of the `SceSaveDataDialogParam` type.
(2) Call a function.
(3) Wait for the response from the dialog.
(4) Retrieve the call results.
(5) Repeat steps (2) to (4).
(6) End 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`.

Save Data Dialog Call Procedure

First, prepare the `SceSaveDataDialogParam` type variable and following initialization with `sceSaveDataDialogParamInit()`, be sure to set the operation mode (*`mode`*) and the parameters that are required accordingly.

(1) Calling the function

Call a Save Data Dialog function with `sceSaveDataDialogInit()`. Specify the `SceSaveDataDialogParam` type variable set beforehand as the argument.

(2) Waiting for the response from the dialog

According to the following rules, call `sceSaveDataDialogGetStatus()` and `sceSaveDataDialogGetSubStatus()` to poll the operation status of Save Data Dialog and the operation substatus at each frame.

- Retrieve the operation status with `sceSaveDataDialogGetStatus()`. Retrieve the operation substatus with `sceSaveDataDialogGetSubStatus()` while the operation status is `SCE_COMMON_DIALOG_STATUS_RUNNING`.
- While the operation substatus is `SCE_COMMON_DIALOG_STATUS_RUNNING`, either user response wait or dialog display is in progress, so wait until the status changes to `SCE_COMMON_DIALOG_STATUS_FINISHED`.

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.

(3) Retrieving the call results

When the operation substatus changes to `SCE_COMMON_DIALOG_STATUS_FINISHED`, the results can be retrieved with `sceSaveDataDialogGetResult()`. The results that can be retrieved include the *`mode`*, *`userdata`*, return code, and selected button(s) during the call, as well as the save data information freely specified during the call. The details regarding the return code, etc. are described in a later section in this document.

(4) Continuing and ending call

Only when the operation substatus is `SCE_COMMON_DIALOG_STATUS_FINISHED`, can `sceSaveDataDialogContinue()` and `sceSaveDataDialogFinish()` be called.

`sceSaveDataDialogContinue()` can then call the next Save Data Dialog function. The call method is the same as for `sceSaveDataDialogInit()`, with a `SceSaveDataDialogParam` type variable specified as the argument. However, limitations apply to some of the parameters (described later).

If continuation of the function is not required, call `sceSaveDataDialogFinish()` and end calling of Save Data Dialog. Normally, the operation status cannot change to `SCE_COMMON_DIALOG_STATUS_FINISHED` unless this function is called. (except when `sceSaveDataDialogAbort()` is called.)

(5) Terminating the processing

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

Aborting the processing

When quitting an application, etc., to abort the display of Message Dialog from the application side on an emergency basis, call `sceSaveDataDialogAbort()`. This ends the display faster than `sceSaveDataDialogSubClose()`, and the operation status changes directly to `SCE_COMMON_DIALOG_STATUS_FINISHED` without any operation substatus transitions. In this case too, the call results can be retrieved by calling `sceSaveDataDialogGetResult()`. `SCE_COMMON_DIALOG_RESULT_ABORTED` is returned as the return code.

Main APIs Used for Basic Processing

API	Description
`SceSaveDataDialogParam`	Parameter structure such as mode setting
`sceSaveDataDialogParamInit()`	Initializes parameter structure
`sceSaveDataDialogInit()`	Calls function
`sceSaveDataDialogGetStatus()`	Retrieves operation status
`sceSaveDataDialogGetSubStatus()`	Retrieves operation substatus
`sceSaveDataDialogGetResult()`	Retrieves call results
`sceSaveDataDialogContinue()`	Continues calling function
`sceSaveDataDialogFinish()`	Ends continuous calling of function
`sceSaveDataDialogTerm()`	Ends calling of function

3 Reference Information

Save Data Slots

The save data displayed in Save Data Dialog is handled in units called "save data slots". The application utility library is used to manage save data slots.

It is possible to create up to 256 slots, and each slot is assigned an identification number called save data slot ID. This save data slot ID is specified every time Save Data Dialog is called.

Save Data Fixed Display Mode

- One save data for the save data processing flow is displayed on the screen for the user to see.
- Display is performed in the same manner during other save data function calls. The difference with these functions is that only save data display is performed. Normally, the various types of message display function are called immediately following this, and this mode will not cause save data only to remain displayed.

Figure 2 Save Data Fixed Display Dialog

The following call result can be retrieved following save data display.

Call result : Success (0) / Failure (error code)
Button ID : None (`SCE_SAVEDATA_DIALOG_BUTTON_ID_INVALID`)

Save Data List Display Mode

- This mode displays on screen several save data for save data processing flow, for the user to choose from. It is also possible to specify the save data to be initially focused. Since this is a touch type display with no focus expression feature, the display position is automatically adjusted so that the specified save data is fit within the screen.

Figure 3 Save Data List Display Dialog

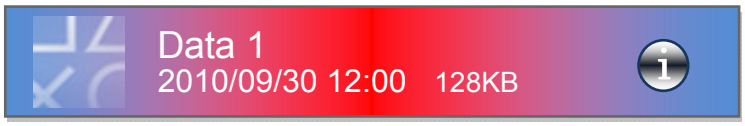
- The layout of the save data information displayed in the list can be selected from the following three types.

Example Title/Subtitle/Date (3-line display)
Title/Date/Subtitle (3-line display)
Title/Date (2-line display)

Figure 4 Display Layout 1 (Title/Subtitle/Date)

Figure 5 Display Layout 2 (Title/Date/Subtitle)

Figure 6 Display Layout 3 (Title/Date)



When using the "Title/Date (2-line display)" layout, the amount of Subtitle information is reduced, so set the title character string with special caution to make the respective data easily identifiable.

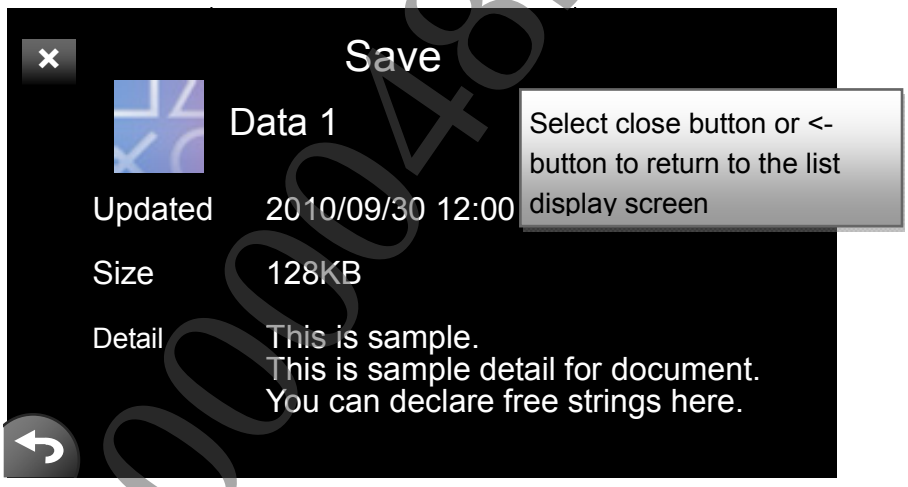
- Selecting the "i" icon causes the screen to change to the detailed information screen for the corresponding save data. Selecting the **<-** button on the detailed information screen causes the screen to change back to the list display screen. As the transitions to/from the detailed information screen are automatically done internally in Common Dialog, the application need not be aware of them. The detailed information screen shows the following information corresponding to the save data, which can be referenced by the user when overwriting or deleting the save data.
  - Title
  - Thumbnail
  - Updated date and time
  - Size (When != 0)
  - Detailed information

* The Title, Thumbnail, Updated date and time, and Size contents are the same as displayed on the list display screen. On the list display screen, information that is too long to fit in the display field is displayed through horizontal scrolling, but on the detailed information screen, it is displayed with auto-wraparound. Also, Subtitle displayed on the list display screen is displayed first as the "Detail" item, followed by the detailed character strings. (Even when using the "Title/Date (2-line display)" layout, the Subtitle information is displayed on the detailed information screen.)

```
Example:
    SubTitle : This is sample.
    Detail : This is sample detail for document.You can declare free strings here.

    → Displayed : This is sample.
                  Detail for document.You can declare free strings here.
```

Figure 7　Save Data Detailed Information Display Screen



- When the user selects or chooses to cancel save data, the following call results can be retrieved.

Call result　:　Success (0) / Cancel (`SCE_COMMON_DIALOG_RESULT_USER_CANCELED`) / Failure (error code)

Button ID　:　None (`SCE_SAVEDATA_DIALOG_BUTTON_ID_INVALID`)

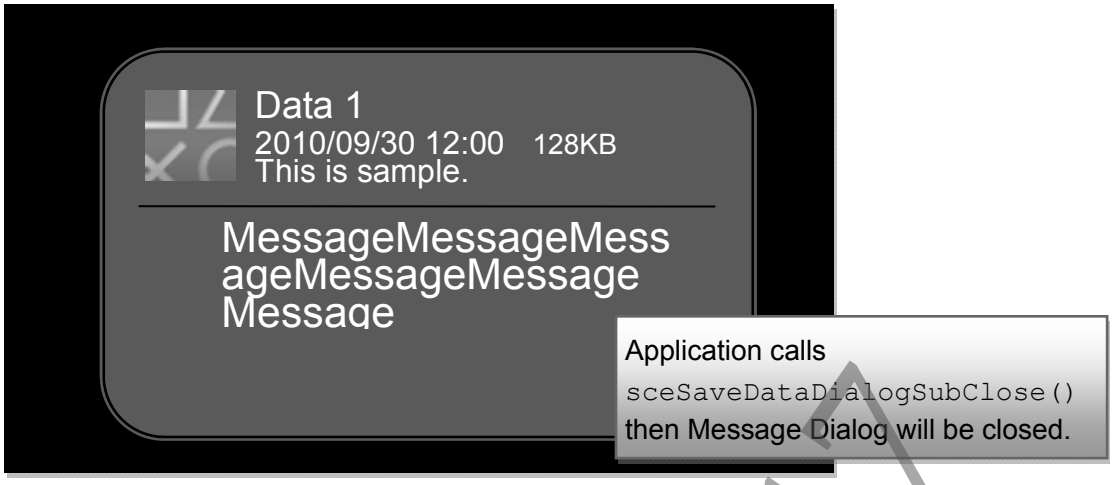
User Specified Message Display Mode

- The target save data for save data fixed display can be specified.
  (only for call using `sceSaveDataDialogInit()`)
- Applications can freely specify character strings to be displayed.
- Applications can also freely specify buttons to be displayed.
  Example **OK** button
  Yes/**No** button
  No buttons
- The purpose for which this is to be used is freely selectable by the application.

Figure 8 Save Data Dialog with OK Button

Figure 9 Save Data Dialog with Yes and No Buttons

Figure 10 Save Data Dialog without Button



- The following call result can be retrieved when the user selects a button to close Message Dialog.
  Call result : Success (0)
  Button ID : **OK** button (`SCE_SAVEDATA_DIALOG_BUTTON_ID_OK`)
  Yes button (`SCE_SAVEDATA_DIALOG_BUTTON_ID_YES`)
  No button (`SCE_SAVEDATA_DIALOG_BUTTON_ID_NO`)

System Defined Message Display Mode

- The save data for save data fixed display can be specified.
  (only when calling with `sceSaveDataDialogInit()`)
- The messages previously prepared for the system are displayed.
- General purpose wordings and wordings for TRC compliance are provided as messages.
  Example "Save data?"
  "Save completed"
- The display language setting of the main unit is automatically reflected.
- Buttons cannot be specified. They are determined uniquely by the message type.

Figure 11 System Defined Message Display Save Data Dialog with Yes and No Buttons

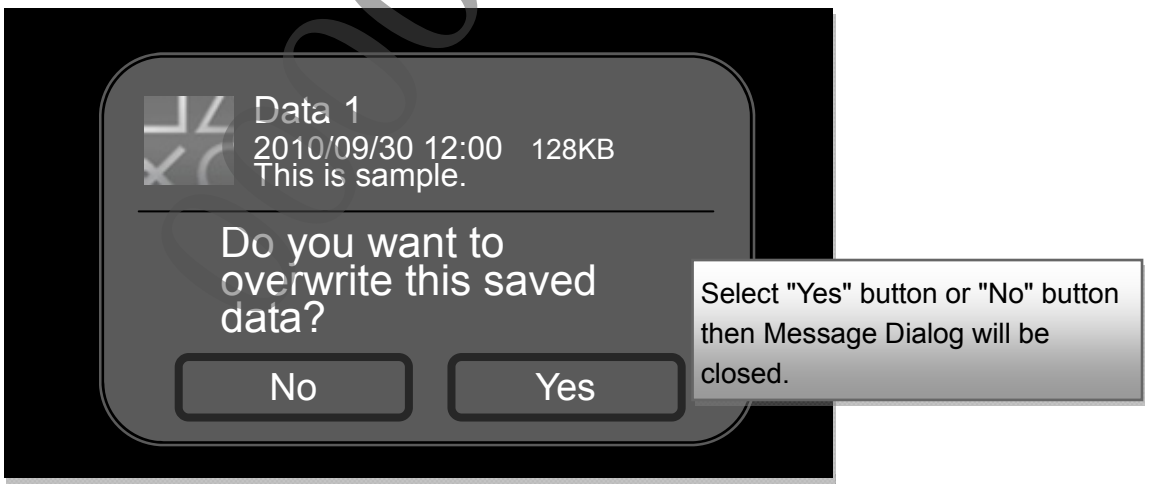


Figure 12 System Defined Message Save Data Dialog with OK Button

- The following call result can be retrieved when the user selects a button and Message Dialog closes.
  Call Result : Success (0)
  Button ID : **OK** button (`SCE_SAVEDATA_DIALOG_BUTTON_ID_OK`)
  Yes button (`SCE_SAVEDATA_DIALOG_BUTTON_ID_YES`)
  No button (`SCE_SAVEDATA_DIALOG_BUTTON_ID_NO`)

Error Code Display Mode

- The save data for save data fixed display can be specified.
  (only for call using `sceSaveDataDialogInit()`)
- The short error code for end users is displayed. For details on the short error code, refer to the "Error Overview" document.
- Call Save Data Dialog with the specification of a hexadecimal error code returned by the SDK library. Some of the error codes are automatically replaced with wordings that explain the error details.
- Display character strings cannot be specified.
- Buttons cannot be specified. They are determined uniquely by the display contents.

Figure 13 Error Code Display Save Data Dialog

Common Call Results

If Save Data Dialog is closed by other than user operation, the following call results are returned in common to all the operation modes.

- The application closed Message Dialog (closed by `sceSaveDataDialogSubClose()`)
  Call Result : Success (0)
  Button ID : None (`SCE_SAVEDATA_DIALOG_BUTTON_ID_INVALID`)

- The application aborted Message Dialog call (closed by `sceSaveDataDialogAbort()`)
  Call Result : Aborted (`SCE_COMMON_DIALOG_RESULT_ABORTED`)
  Button ID : None (`SCE_SAVEDATA_DIALOG_BUTTON_ID_INVALID`)

Display Wording Types

The value specified for the *`dispType`* parameter of the `SceSaveDataDialogParam` structure passed to `sceSaveDataDialogInit()` is used to determine the type of wordings to be used during Save Data Dialog calls. As an example, the system defined messages change as follows.

When `SCE_SAVEDATA_DIALOG_TYPE_SAVE` is specified

`SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_CONFIRM` : "Do you want to save the data?"
`SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_PROGRESS` : "Saving...
Do not power off the system or close the application."
`SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_FINISHED` : "Saving complete."

When `SCE_SAVEDATA_DIALOG_TYPE_LOAD` is specified

`SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_CONFIRM` : "Do you want to load this saved data?"
`SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_PROGRESS` : "Loading..."
`SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_FINISHED` : "Loading complete."

Call Parameter Limitations

When a `SceSaveDataDialogParam` type variable is passed to `sceSaveDataDialogContinue()`, some of the parameters inherit the settings passed to `sceSaveDataDialogInit()`. Concretely, the following parameters are ignored.

- *`sdkVersion`*
- *`flag`* (not supported in this SDK)

Moreover, the following parameter can be overwritten, but if omitted, the value specified with `sceSaveDataDialogInit()` is inherited.

- *`dispType`*

Available Thumbnails

For the save data thumbnails displayed in Save Data Dialog, `SceAppUtilSaveDataSlotParam.`*`iconPath`*, set in the save data slot by the application utility library, is referenced. If the save data slot does not exist, `SceAppUtilSaveDataSlotEmptyParam.`*`iconPath`* or `SceAppUtilSaveDataSlotEmptyParam.`*`iconBuf`* specified when calling Save Data Dialog is referenced.

Image files which can be specified as thumbnails must meet the following specifications:

- Must be in PNG format
- Resolution must be under 960 x 544
- File size must be under 960*544*4= 2088960 [byte]

If the above specifications are not met, a default icon stored in the system will be used as thumbnail.

Also, thumbnail images will be resized so as to fit in a 160*90 square while maintaining the same aspect ratio.

File path specification to *`iconPath`* is performed as follows:

`iconPath` : `"host0:icon0.png"`	Specify the path to the file on the host (can only be used in the development environment)
`iconPath` : `"app0:thumbnail/icon0.png"`	Specify the file in the game data path
`iconPath` : `"savedata0:icon0.png"`	Specify the file in the save data

For a file path whose *`iconPath`* begins with “app0:”, `SceSaveDataDialogSlotConfigParam` can be used to replace the file path with arbitrary subdirectory under the “app0:”.

Example) app0:oldDir/icon0.png -> app0:newDir/icon0.png

This function is primarily used for save data transfer functions. For save data transfer functions, refer to the "Save Data User's Guide" document.

Retrieval of Save Data Information

With regard to variables of the `SceSaveDataDialogResult` type passed to `sceSaveDataDialogGetResult()`, the detailed information of the save data can be received by the structure specified to the `SceSaveDataDialogResult.`*`slotInfo`* parameter.

`SceSaveDataDialogSlotInfo.`*`isExist`*	:	Information on whether save data exists or not.
`SceSaveDataDialogSlotInfo.`*`slotParam`*	:	Save data slot parameter

If `SceSaveDataDialogResult.`*`slotInfo`* and `SceSaveDataDialogSlotInfo.`*`slotParam`* are not required, receiving their information can be skipped by setting NULL.

Mount Point

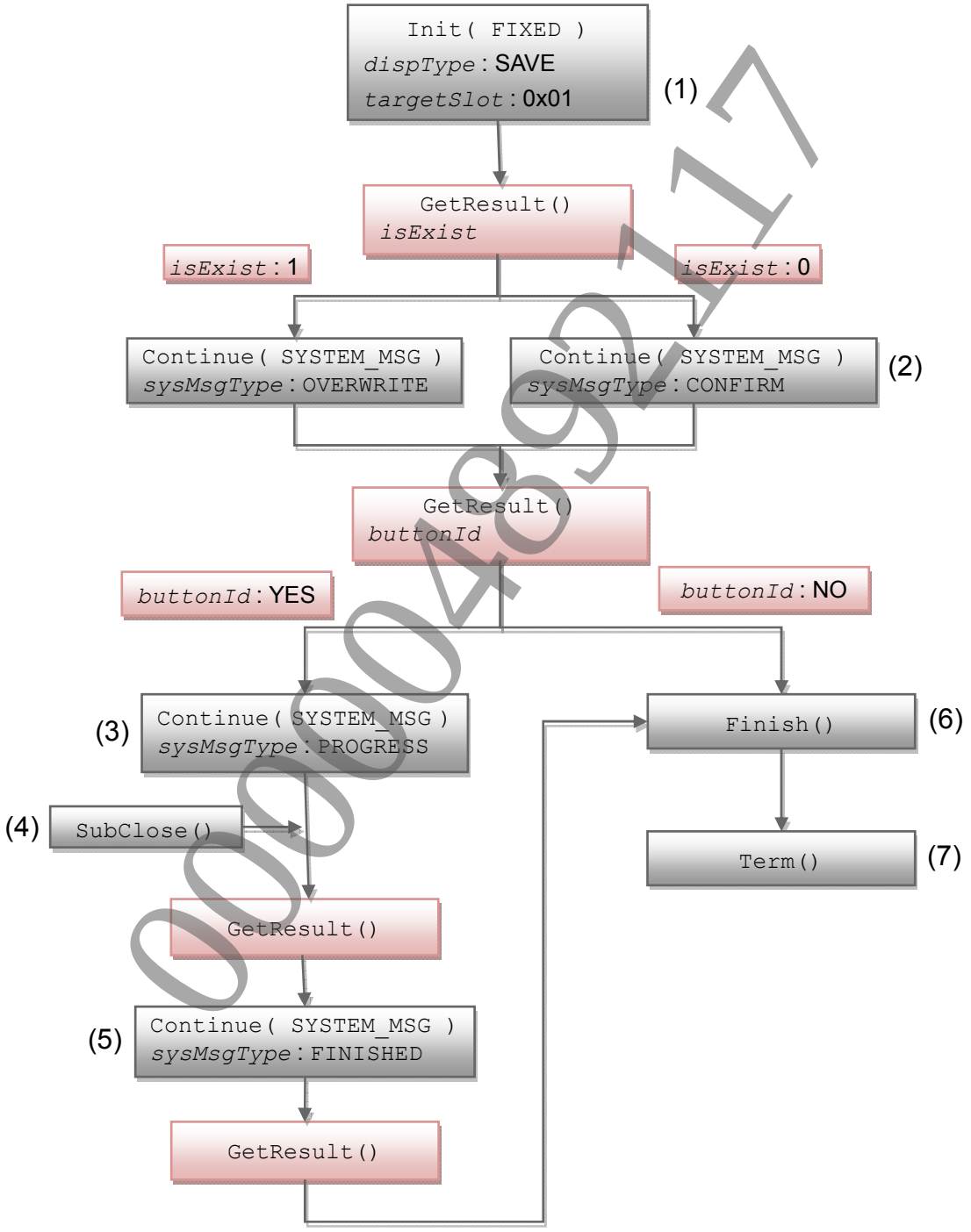
By default, the Save Data Dialog functions display the save data slots saved in “savedata0:”. However, the save data slots saved in “savedata1:” can be displayed by explicitly specifying “savedata1:” as the *`mountPoint`* in `SceSaveDataDialogSlotConfigParam`. This function is used for save data transfer functions. For save data transfer functions, refer to the "Save Data User's Guide" document.

Save Data Flow Execution Procedure

Save Data Dialog provides the GUI components required for the display of save data operations in primitive units. The procedure required for realizing the GUI flow for a typical save data operation is introduced below.

For simplicity, the prefixes of function names and constants (`sceSaveDataDialog*`, `SCE_SAVEDATA_DIALOG_*`) have been omitted in Figure 14 and Figure 15.

Figure 14 Fixed Save Flow Execution Procedure



(1) Calling Save Data Dialog function (operation mode: Save data fixed display)

Call `sceSaveDataDialogInit()` in the `SCE_SAVEDATA_DIALOG_MODE_FIXED` mode (save data fixed display). At this time, specify `SCE_SAVEDATA_DIALOG_TYPE_SAVE` for the display wording type (`SceSaveDataDialogParam.`*`dispType`*), and the save data slot ID for the next processing for the ID of the display target save data slot (`SceSaveDataDialogParam.`*`fixedParam->targetSlot`*). Whether or not this save data slot does actually exist can be received to `SceSaveDataDialogResult.`*`slotInfo->isExist`* using `sceSaveDataDialogGetResult()`.

(2) Displaying save confirmation dialog

To provide confirmation of save to the user, call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display). At this time, if save data does exist, specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_OVERWRITE` ("Overwrite data?") to the message type (`SceSaveDataDialogParam.`*`sysMsgParam->sysMsgType`*), and if save data does not exist, specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_CONFIRM` ("Save data?").

(3) Receiving save confirmation result and displaying dialog accordingly

The save confirmation result is received by `SceSaveDataDialogResult.`*`buttonId`* using `sceSaveDataDialogGetResult()`. If it was `SCE_SAVEDATA_DIALOG_BUTTON_ID_OK` (**OK** button selected), then call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display). Specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_PROGRESS` ("Saving... Please wait.") to the message type (`SceSaveDataDialogParam.`*`sysMsgParam->sysMsgType`*.)

If the user selected not to save (`SCE_SAVEDATA_DIALOG_BUTTON_ID_NO`), call `sceSaveDataDialogFinish()` to finish the fixed save flow.

(4) Closing the saving dialog upon completion of the save processing

Once the save processing is completed, call `sceSaveDataDialogSubClose()` and close the dialog called by specifying `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_PROGRESS` ("Saving... Please wait.") to the message type from the program.

(5) Displaying the save completion dialog

Specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_FINISHED` ("Save completed." display) to the message type (`SceSaveDataDialogParam.`*`sysMsgParam->sysMsgType`*) and call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display). As a result, a dialog indicating save completion is displayed. After this dialog is closed, call `sceSaveDataDialogFinish()` and finish the fixed save flow.

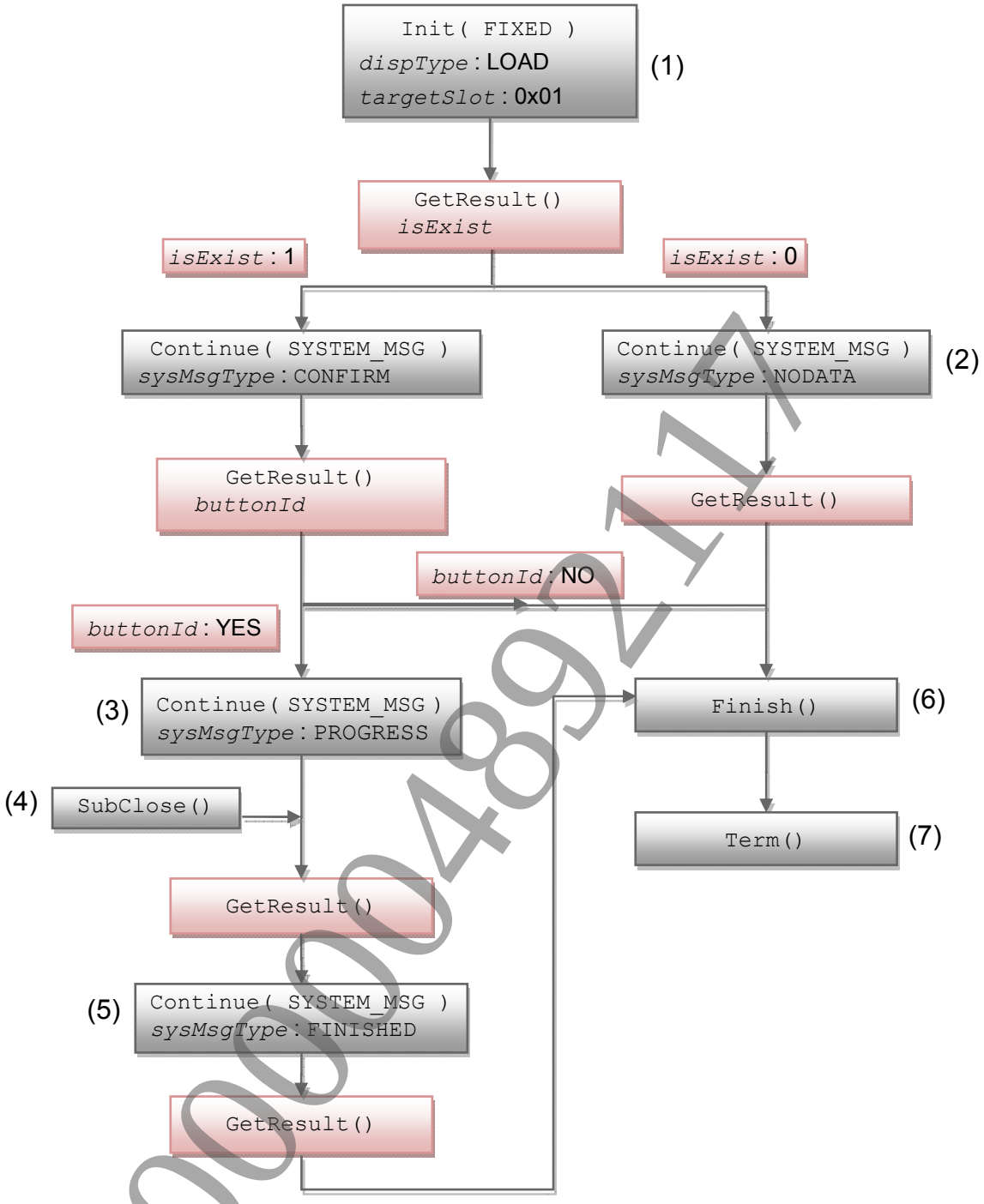
(6) Ending continuous calling of Save Data Dialog

When `sceSaveDataDialogFinish()` is called, continuous calling of Save Data Dialog ends, and the operation status changes to `SCE_COMMON_DIALOG_STATUS_FINISHED`.

(7) Ending calling of Save Data Dialog

Lastly, call `sceSaveDataDialogTerm()` and complete calling of Save Data Dialog.

Figure 15 Fixed Load Flow Execution Procedure



(1) Calling Save Data Dialog Function (operation mode: Save data fixed display)

Call `sceSaveDataDialogInit()` in the `SCE_SAVEDATA_DIALOG_MODE_FIXED` mode (save data fixed display). At this time, specify `SCE_SAVEDATA_DIALOG_TYPE_LOAD` (load wording display) to the display wording type (`SceSaveDataDialogParam.`*`dispType`*), and the save data slot ID to be processed next to the ID of the display target save data slot (`SceSaveDataDialogParam.`*`fixedParam->targetSlot`*). Whether or not this save data slot does actually exist can be received to `SceSaveDataDialogResult.slot`*`Info->isExist`* using `sceSaveDataDialogGetResult()`.

(2) Displaying load confirmation dialog (if there is no save data, flow is ended)

If save data exists, specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_CONFIRM` ("Load data?") to the message type (`SceSaveDataDialogParam.`*`sysMsgParam`*`->`*`sysMsgType`*) for confirming load with the user, and call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display).

If save data does not exist, in order to inform the user of this fact, specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_NODATA` ("There is no saved data.") to the message type (`SceSaveDataDialogParam.`*`sysMsgParam`*`->`*`sysMsgType`*), and call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display). After this dialog is closed, call `sceSaveDataDialogFinish()` to finish fixed load flow.

(3) Displaying loading dialog (if user does not select load, flow is ended)

The load confirmation result is received by `SceSaveDataDialogResult.`*`buttonId`* using `sceSaveDataDialogGetResult()`. If it was `SCE_SAVEDATA_DIALOG_BUTTON_ID_OK` (**OK** button selected), next call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display). Specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_PROGRESS` ("Loading. Please wait.") to the message type (`SceSaveDataDialogParam.`*`sysMsgParam`*`->`*`sysMsgType`*).

If the user selected not to save (`SCE_SAVEDATA_DIALOG_BUTTON_ID_NO`), call `sceSaveDataDialogFinish()` to finish the fixed load flow.

(4) Closing the load dialog upon completion of the load processing

Once the save processing is completed, call `sceSaveDataDialogSubClose()` and close the dialog called by specifying `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_PROGRESS` ("Loading... Please wait.") from the program.

(5) Displaying the dialog announcing load completion

Specify `SCE_SAVEDATA_DIALOG_SYSMSG_TYPE_FINISHED` ("Load completed") to the message type (`SceSaveDataDialogParam.`*`sysMsgParam`*`->`*`sysMsgType`*) and call `sceSaveDataDialogContinue()` in the `SCE_SAVEDATA_DIALOG_MODE_SYSTEM_MSG` mode (system defined message display). As a result, a dialog indicating load completion is displayed. After this dialog is closed, call `sceSaveDataDialogFinish()` and finish the fixed load flow.

(6) Ending continuous calling of Save Data Dialog

When `sceSaveDataDialogFinish()` is called, continuous calling of Save Data Dialog ends, and the operation status changes to `SCE_COMMON_DIALOG_STATUS_FINISHED`.

(7) Ending calling of Save Data Dialog

Lastly, call `sceSaveDataDialogTerm()` and complete calling of Save Data Dialog.

4 Display of Save Data for PSP™ (PlayStation®Portable)

Overview

Save data for PSP™ can be displayed using the Save Data Dialog library. A list of save data can be displayed for a user so they can make a selection, and various Message Dialog types can be displayed.

> **Note**
> As with display of save data for PlayStation®Vita, the Save Data Dialog library only performs display of save data for PSP™. For actual data loading, refer to the "Application Utility Overview", "Application Utility Reference", and "Save Data User's Guide" documents.

Usage

The Save Data Dialog library has a subset of APIs and structures for handling save data for PSP™. Save data for PlayStation®Vita is handled with slot specification while save data for PSP™ differs since it is handled with directory name specification, but the API interface and calling method is almost exactly the same as the standard functionality for displaying save data for PlayStation®Vita.

Save Data References

The save location of save data for PSP™ that is referenced by the display feature for save data for PSP™ is in the system area of memory cards. In order to place save data for PSP™ here, use the Content Manager application to transfer save data for PSP™ saved in a PC or PlayStation®3. This method is the same one used by end users.

Sample Program

Refer to the following sample program for save data for PSP™.

sample_code/system/api_savedata/psp_basic/

This sample uses the Save Data Dialog display features for save data for PSP™ and implements the save data fixed display type and list display type load flows.

5 Precautions

Limitations

Common Dialog limitations apply.