

Application Utility Reference

© 2015 Sony Computer Entertainment Inc.
All Rights Reserved.
SCE Confidential

Table of Contents

Constants	5
Drive Names	6
Event Notification Type	7
Event Notification Parameter Size	8
Maximum Size of Event Notification Parameters from LiveArea™	9
Maximum Parameter Size for Event Notification from Activity	10
Maximum Parameter Size for Event Notification from Teleport Library	11
Title ID Size	12
Passcode Size	13
Maximum Size of a Mount Point Name	14
Save Data Slot Parameter Sizes	15
Save Data Slot Status	16
Unusable Save Data Slot ID	17
Maximum Data Path Size for Save Data	18
Data Save Mode for Save Data	19
Data Removal Mode for Save Data	20
Save Data Slot Search Target	21
Sort Key for Save Data Slot Search	22
Sort Order for a Save Data Slot Search	23
Work Buffer Size for a Save Data Slot Search	24
Maximum Number of Specifiable Save Data Data	25
Maximum Writable Size for Save Data	26
Maximum Filename Size for Save Data Files for PSP™	27
File ID Size for Protected Save Data Files for PSP™	28
Maximum Buffer Size for a Save Data File for PSP™	29
Maximum Number of Save Data for PSP™ to List	30
Additional Content Directory Name Size	31
Application Parameter ID	32
SKU Flag	33
Safe Memory Area Size	34
Store Browsing Type	35
Internet Browser Application Startup Type	36
Internet Browser Application Startup Command Type	37
Screenshot Trigger Type	38
Maximum Size of the Screenshot Content Path	39
Types	40
SceAppUtilBootAttribute	41
SceAppUtilAppEventType	42
SceAppUtilSaveDataSlotId	43
SceAppUtilSaveDataSlotStatus	44
SceAppUtilSaveDataDataSaveMode	45
SceAppUtilSaveDataDataRemoveMode	46
SceAppUtilPspSaveDataVersion	47
SceAppUtilPspSaveDataType	48

SCE CONFIDENTIAL

SceAppUtilPspSaveDataParamSfoSize	49
SceAppUtilAppParamId	50
SceAppUtilScreenShotTriggerType	51
Structures	52
SceAppUtilInitParam	53
SceAppUtilBootParam	54
SceAppUtilAppEventParam	55
SceAppUtilWorkBuffer	56
SceAppUtilNpInviteMessageParam	57
SceAppUtilNpAppDataMessageParam	58
SceAppUtilNearGiftParam	59
SceAppUtilLiveAreaParam	60
SceAppUtilScreenShotNotification	61
SceAppUtilNpActivityParam	62
SceAppUtilTeleportParam	63
SceAppUtilSessionInvitationParam	64
SceAppUtilGameCustomDataParam	65
SceAppUtilMountPoint	66
SceAppUtilTitleId	67
SceAppUtilPassCode	68
SceAppUtilSaveDataSlotParam	69
SceAppUtilSaveDataSlotEmptyParam	71
SceAppUtilSaveDataSlotSearchCond	72
SceAppUtilSlotSearchResult	74
SceAppUtilSaveDataSlot	75
SceAppUtilSaveDataDataSaveItem	76
SceAppUtilSaveDataDataRemoveItem	77
SceAppUtilSaveDataDataSlot	78
SceAppUtilPspSaveDataParamSfo	79
SceAppUtilPspSaveDataParams	80
SceAppUtilPspSaveDataDirName	82
SceAppUtilDrmAddcontId	83
SceAppUtilBgdlStatus	84
SceAppUtilStoreBrowseParam	85
SceAppUtilWebBrowserParam	86
SceAppUtilWebBrowserAddCookieParam	87
SceAppUtilWebBrowserResetCookieParam	88
Functions	89
sceAppUtilInit	90
sceAppUtilShutdown	92
sceAppUtilReceiveAppEvent	93
sceAppUtilAppEventParseNpInviteMessage	95
sceAppUtilAppEventParseNpAppDataMessage	96
sceAppUtilAppEventParseNearGift	98
sceAppUtilAppEventParseLiveArea	99
sceAppUtilAppEventParseScreenShotNotification	100
sceAppUtilAppEventParseNpActivity	102
sceAppUtilAppEventParseTeleport	104

©SCEI

sceAppUtilAppEventParseSessionInvitation.....	105
sceAppUtilAppEventParseGameCustomData.....	107
sceAppUtilSaveDataMount.....	108
sceAppUtilSaveDataUmount.....	110
sceAppUtilSaveDataSlotCreate.....	111
sceAppUtilSaveDataSlotDelete.....	113
sceAppUtilSaveDataSlotSearch.....	115
sceAppUtilSaveDataSlotSetParam.....	117
sceAppUtilSaveDataSlotGetParam.....	119
sceAppUtilSaveDataDataSave.....	121
sceAppUtilSaveDataDataRemove.....	125
sceAppUtilSaveDataGetQuota.....	128
sceAppUtilPspSaveDataGetDirNameList.....	130
sceAppUtilPspSaveDataLoad.....	132
sceAppUtilAddcontMount.....	134
sceAppUtilAddcontUmount.....	136
sceAppUtilDrmOpen.....	137
sceAppUtilDrmClose.....	139
sceAppUtilBgdlGetStatus.....	141
sceAppUtilPhotoMount.....	142
sceAppUtilPhotoUmount.....	143
sceAppUtilMusicMount.....	144
sceAppUtilMusicUmount.....	145
sceAppUtilExtVideoMount.....	146
sceAppUtilExtVideoUmount.....	148
sceAppUtilSystemParamGetInt.....	149
sceAppUtilSystemParamGetString.....	151
sceAppUtilAppParamGetInt.....	153
sceAppUtilSaveSafeMemory.....	154
sceAppUtilLoadSafeMemory.....	156
sceAppUtilStoreBrowse.....	158
sceAppUtilLaunchWebBrowser.....	160
sceAppUtilAddCookieWebBrowser.....	161
sceAppUtilResetCookieWebBrowser.....	162
Error Codes.....	163
Common Error Codes.....	164
Error Codes for Event Notification.....	165
Error Codes for Save Data.....	166
Error Codes for Additional Content.....	167
Error Codes for Content Data Mounting.....	168
Parameter Error Codes.....	169
Appendix: System Parameters.....	172
System Parameter ID.....	173
Language Settings.....	174
User Name Size for Display.....	175
SceSystemParamId.....	176
SceSystemParamLang.....	177

Constants

000004892117

Drive Names

Drive names for accessing various data directories

Definition

Value	(String)	Description
SCE_APPUTIL_DRIVE_NAME_APP	"app0:"	Name of main application data drive
SCE_APPUTIL_DRIVE_NAME_SAVEDATA	"savedata0:"	Name of save data drive
SCE_APPUTIL_DRIVE_NAME_SAVEDATA_ADD	"savedata1:"	Name of additional save data drive
SCE_APPUTIL_DRIVE_NAME_ADDCONT	"addcont0:"	Name of additional content drive
SCE_APPUTIL_DRIVE_NAME_ADDCONT_ADD	"addcont1:"	Name of additional drive for additional content
SCE_APPUTIL_DRIVE_NAME_PHOTO	"photo0:"	Name of photo data drive
SCE_APPUTIL_DRIVE_NAME_MUSIC	"music0:"	Name of music data drive
SCE_APPUTIL_DRIVE_NAME_VIDEO	"video0:"	Name of video data drive

Description

These are the string constant definitions for performing file access for various data.

"app0:" and "savedata0:" will be automatically mounted upon application boot, but other drives must be explicitly mounted with the respective mounting functions before access can be performed.

See Also

`sceAppUtilSaveDataMount()`, `sceAppUtilAddcontMount()`, `sceAppUtilPhotoMount()`,
`sceAppUtilMusicMount()`, `sceAppUtilExtVideoMount()`

SCE CONFIDENTIAL

Event Notification Type

Application event notification types

Definition

Value	(Number)	Description
SCE_APPUTIL_APPEVENT_TYPE_INVALID	0	Invalid event notification type
SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE	1	Invitation message event notification
SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE	2	Event notification of a message with game data attached
SCE_APPUTIL_APPEVENT_TYPE_NEAR_GIFT	4	Event notification of a "near" gift
SCE_APPUTIL_APPEVENT_TYPE_LIVEAREA	5	Event notification of application startup from LiveArea™
SCE_APPUTIL_APPEVENT_TYPE_SCREENSHOT_NOTIFICATION	8	Event notification of a screenshot capture
SCE_APPUTIL_APPEVENT_TYPE_NP_ACTIVITY	11	Event notification of application startup from an activity
SCE_APPUTIL_APPEVENT_TYPE_TELEPORT	12	Event notification of application startup from the Teleport library
SCE_APPUTIL_APPEVENT_TYPE_SESSION_INVITATION	13	Session invitation event notification
SCE_APPUTIL_APPEVENT_TYPE_GAME_CUSTOM_DATA	14	Game custom data event notification

Description

This is a constant definition representing the application event notification type. The type is obtained by specifying it to the *type* member of the *SceAppUtilAppEventParam* structure, which is an argument of *sceAppUtilReceiveAppEvent()*.

See Also

Event Notification Parameter Size, *SceAppUtilAppEventType*, *SceAppUtilAppEventParam*, *sceAppUtilReceiveAppEvent()*

SCE CONFIDENTIAL

Event Notification Parameter Size

Size of application event notification parameter

Definition

Value	(Number)	Description
SCE_APPUTIL_APPEVENT_PARAM_MAXSIZE	1024	The maximum size of event notification parameter

Description

This is a size definition of the members referenced by the `SceAppUtilAppEventParam` structure.

See Also

`SceAppUtilAppEventParam`

SCE CONFIDENTIAL

Maximum Size of Event Notification Parameters from LiveArea™

Maximum size of event notification parameter of application startup from LiveArea™

Definition

Value	(Number)	Description
SCE_APPUTIL_LIVEAREA_PARAM_MAX_SIZE	1019	Maximum size of event notification parameters of application startup from LiveArea™

Description

This is a size definition of the members referenced by the `SceAppUtilLiveAreaParam` structure.

See Also

`SceAppUtilLiveAreaParam`

SCE CONFIDENTIAL

Maximum Parameter Size for Event Notification from Activity

Maximum parameter size for event notification of application startup from an activity

Definition

Value	(Number)	Description
SCE_APPUTIL_NP_ACTIVITY_PARAM_MAXSIZE	897	Maximum parameter size for event notification of application startup from an activity

Description

This is a size definition of the members referenced by the `SceAppUtilNpActivityParam` structure.

See Also

`SceAppUtilNpActivityParam`

SCE CONFIDENTIAL

Maximum Parameter Size for Event Notification from Teleport Library

Maximum parameter size for event notification of application startup from the Teleport library

Definition

Value	(Number)	Description
SCE_APPUTIL_TELEPORT_PARAM_MAXSIZE	952	Maximum parameter size for event notification of application startup from the Teleport library

Description

This is a size definition of the members referenced by the `SceAppUtilTeleportParam` structure.

See Also

`SceAppUtilTeleportParam`

SCE CONFIDENTIAL

Title ID Size

Size of the title ID of the application

Definition

Value	(Number)	Description
SCE_APPUTIL_TITLE_ID_DATA_SIZE	10	Size of the title ID of the application

Description

This is a constant definition representing the application's title ID size.

Specify SCE_APPUTIL_TITLE_ID_DATA_SIZE alphanumeric characters including the NULL terminator in the title ID of the application.

See Also

SceAppUtilTitleId

SCE CONFIDENTIAL

Passcode Size

Passcode size

Definition

Value	(Number)	Description
SCE_APPUTIL_PASSCODE_DATA_SIZE	32	Passcode size

Description

This constant definition represents the size of the passcode, which is set per application package and required upon mounting save data or additional content. This size does not include the NULL terminator.

See Also

SceAppUtilPassCode

SCE CONFIDENTIAL

Maximum Size of a Mount Point Name

Maximum size of a mount point name

Definition

Value	(Number)	Description
SCE_APPUTIL_MOUNTPOINT_DATA_MAXSIZE	16	Maximum size of a mount point name

Description

This constant definition represents the maximum size of names for various mount points accessed by the application. This size includes the colon ":" (representing the drive letter) and the NULL terminator.

See Also

SceAppUtilMountPoint

Save Data Slot Parameter Sizes

Constants referenced by a save data slot APIs

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_SLOT_TITLE_MAXSIZE	64	Maximum size of title name
SCE_APPUTIL_SAVEDATA_SLOT_SUBTITLE_MAXSIZE	128	Maximum size of subtitle name
SCE_APPUTIL_SAVEDATA_SLOT_DETAIL_MAXSIZE	512	Maximum size of detailed information
SCE_APPUTIL_SAVEDATA_SLOT_ICON_PATH_MAXSIZE	64	Maximum size of thumbnail image path
SCE_APPUTIL_SAVEDATA_SLOT_MAX	256	Maximum number of slots

Description

These constants define the various sizes referenced when using a save data slot APIs..

The definitions referenced in the SceAppUtilSaveDataSlotParam structure are included as well.

See Also

SceAppUtilSaveDataSlotParam

SCE CONFIDENTIAL

Save Data Slot Status

Constants referenced when indicating a save data slot status

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_SLOT_STATUS_AVAILABLE	0	Normal status
SCE_APPUTIL_SAVEDATA_SLOT_STATUS_BROKEN	1	Broken status

Description

These constant definitions are referenced when indicating the transaction status of a save data slot.

See Also

`sceAppUtilSaveDataSlotGetParam()`

SCE CONFIDENTIAL

Unusable Save Data Slot ID

Invalid save data slot ID constant

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_SLOT_ID_INVALID	0xFFFFFFFF	Invalid save data slot ID

Description

This constant definition is used when specifying a save data slot ID that is ruled to be unusable.

See Also

`sceAppUtilSaveDataSlotGetParam()`

Maximum Data Path Size for Save Data

Maximum data path size for save data

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_DATA_PATH_MAXSIZE	200	Maximum data path size for save data

Description

This constant definition is the maximum data path size for save data. Specify a data path of up to SCE_APPUTIL_SAVEDATA_DATA_PATH_MAXSIZE characters, including the NULL terminator.

This constant definition represents the upper limit of the data path specified in the *dataPath* member of the *SceAppUtilSaveDataDataSaveItem* structure, which is specified in *data*, the second argument of *sceAppUtilSaveDataDataSave()*, a function for saving save data; and the upper limit of the data path specified in the *dataPath* member of the *SceAppUtilSaveDataDataRemoveItem* structure, which is specified in *data*, the second argument of *sceAppUtilSaveDataDataRemove()*.

See Also

SceAppUtilSaveDataDataSaveItem, *SceAppUtilSaveDataDataRemoveItem*,
sceAppUtilSaveDataDataSave(), *sceAppUtilSaveDataDataRemove()*

Data Save Mode for Save Data

Constants for specifying the data save mode for save data

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_FILE	0	File operation
SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_FILE_TRUNCATE	1	File operation (truncating the file by the specified offset and size)
SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_DIRECTORY	2	Directory operation

Description

These constant definitions are used when specifying the data save mode for save data.

Set the constant in the *mode* member of the `SceAppUtilSaveDataDataSaveItem` structure, which is given to *data*, the second argument of `sceAppUtilSaveDataDataSave`, a function for saving save data.

In the case that the size of the file after the write is less than that of the existing file when specifying `SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_FILE_TRUNCATE`, the file will be truncated to that size.

See Also

`SceAppUtilSaveDataDataSaveMode`, `SceAppUtilSaveDataDataSaveItem`,
`sceAppUtilSaveDataDataSave()`

Data Removal Mode for Save Data

Constants for specifying the data removal mode for save data

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_DEFAULT	0	Default operation
SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_KEEP_SLOT	1	Remove only the file/directory without removing the save data slots

Description

This constant definition is used when specifying data removal mode for save data.

It is specified in the *mode* member of the `SceAppUtilSaveDataDataRemoveItem` structure, which is given to *data*, the second argument of `sceAppUtilSaveDataDataRemove()`, a function for removing save data.

When `SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_DEFAULT` is specified, the file/directory is removed followed by the removal of save data slots.

When `SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_KEEP_SLOT` is specified, only the file/directory is removed and the save data slots are kept.

See Also

`SceAppUtilSaveDataDataRemoveMode`, `SceAppUtilSaveDataDataRemoveItem`, `sceAppUtilSaveDataDataRemove()`

Save Data Slot Search Target

Constants for specifying the save data slot search target

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EXIST_SLOT	0	Target existing slots
SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EMPTY_SLOT	1	Target empty slots

Description

This constant definition is used upon specifying the save data slot search target.

It is specified to the *type* member of the `SceAppUtilSaveDataSlotSearchCond` structure, which is passed to the second argument, *cond*, of the `sceAppUtilSaveDataSlotSearch()` function for searching save data slots.

`SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EXIST_SLOT` specification targets existing slots for the search.

`SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EMPTY_SLOT` specification targets empty slots for the search.

See Also

`SceAppUtilSaveDataSlotSearchCond`, `sceAppUtilSaveDataSlotSearch()`

Sort Key for Save Data Slot Search

Constants for specifying the sort key for a save data slot search

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SLOT_ID	0	Uses save data slot ID as the sort key
SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_USER_PARAM	1	Uses user parameter as the sort key
SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SIZE_KIB	2	Uses data size as the sort key
SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_MODIFIED_TIME	3	Uses the last update date as the sort key

Description

This constant definition is used to specify the sort key to be used in a save data slot search.

It is specified to the *key* member of the `SceAppUtilSaveDataSlotSearchCond` structure, which is passed to the second argument, *cond*, of the `sceAppUtilSaveDataSlotSearch()` function for searching save data slots.

When `SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SLOT_ID` is specified, the *id* member of the `SceAppUtilSaveDataSlot` save data slot structure is used as the sort key.

When `SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_USER_PARAM` is specified, the *userParam* member of the `SceAppUtilSaveDataSlotParam` save data slot parameter structure is used as the sort key.

When `SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SIZE_KIB` is specified, the *sizeKiB* member of the `SceAppUtilSaveDataSlotParam` save data slot parameter structure is used as the sort key.

When `SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_MODIFIED_TIME` is specified, the *modifiedTime* member of the `SceAppUtilSaveDataSlotParam` save data slot parameter structure is used as the sort key.

See Also

`SceAppUtilSaveDataSlotSearchCond`, `sceAppUtilSaveDataSlotSearch()`

Sort Order for a Save Data Slot Search

Constants for specifying the sort order for a save data slot search

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_ASCENT	0	Sorts in ascending order
SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_DESCENT	1	Sorts in descending order

Description

This constant definition is used to specify the sort order for save data slot search results.

It is specified to the *order* member of the `SceAppUtilSaveDataSlotSearchCond` structure passed to the second argument, *cond*, of the `sceAppUtilSaveDataSlotSearch()` function for searching save data slots.

When `SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_ASCENT` is specified, search results are stored in ascending order.

When `SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_DESCENT` is specified, search results are stored in descending order.

See Also

`SceAppUtilSaveDataSlotSearchCond`, `sceAppUtilSaveDataSlotSearch()`

Work Buffer Size for a Save Data Slot Search

Constants for specifying the work buffer size for a save data slot search

Definition

Value	(Number)	Description
SCE_APPUTIL_WORKBUF_SEARCH_SLOT_DEFAULT_ELEMENT_SIZE	16	Default work buffer size
SCE_APPUTIL_WORKBUF_SEARCH_SLOT_BY_SIZE_KIB_ELEMENT_SIZE	20	Work buffer size when using the data size as the sort key
SCE_APPUTIL_WORKBUF_SEARCH_SLOT_BY_MODIFIEDTIME_ELEMENT_SIZE	32	Work buffer size when using the last update date as the sort key

Description

This constant definition is used to specify the work buffer size for a save data slot search.

Specify the required work buffer to the `SceAppUtilWorkBuffer` structure to pass to the first argument, *workBuf*, of the `sceAppUtilSaveDataSlotSearch()` function upon searching for save data slots. The size of this work buffer differs by the sort key used in the search. A size that is larger than the product of the work buffer size (corresponding to the search sort key) multiplied by the number of searches must be specified or `sceAppUtilSaveDataSlotSearch()` will result in a parameter error.

The number of searches corresponds to the value specified to the *range* member of the second argument, *cond*, of `sceAppUtilSaveDataSlotSearch()`.

See Also

`SceAppUtilWorkBuffer`, `SceAppUtilSaveDataSlotSearchCond`,
`sceAppUtilSaveDataSlotSearch()`

Maximum Number of Specifiable Save Data Data

Maximum number of specifiable save data data

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_DATA_MAXNUM	5	Maximum number of specifiable save data data

Description

This is the maximum number of data that can be specified simultaneously with a single call of `sceAppUtilSaveDataDataSave()`, a function for saving save data, or a single call of `sceAppUtilSaveDataDataRemove()`, a function for removing save data. A value from 1 to `SCE_APPUTIL_SAVEDATA_DATA_MAXNUM` must be specified in *dataNum*, the third argument of `sceAppUtilSaveDataDataSave()` and *dataNum*, the third argument of `sceAppUtilSaveDataDataRemove()`.

See Also

`sceAppUtilSaveDataDataSave()`, `sceAppUtilSaveDataDataRemove()`

SCE CONFIDENTIAL

Maximum Writable Size for Save Data

Maximum size that can be written by a function that saves save data

Definition

Value	(Number)	Description
SCE_APPUTIL_SAVEDATA_DATA_SAVE_MAX SIZE	1*1024*1024	Maximum size that can be written by a function that saves save data (unit/total)

Description

This is the value of the upper limit of save data that can be written by a single call of `sceAppUtilSaveDataDataSave()`, a function that saves save data. When `SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_FILE` or `SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_FILE_TRUNCATE` is specified to the *mode* member of the `SceAppUtilSaveDataDataSaveItem` structure, the second argument of `sceAppUtilSaveDataDataSave()`, the size of the save data to be written is calculated based on the *bufSize* and *offset* members of the structure.

Units of save data to be saved and the total size for the number of files must be `SCE_APPUTIL_SAVEDATA_DATA_SAVE_MAXSIZE` or less.

SCE CONFIDENTIAL

Maximum Filename Size for Save Data Files for PSP™

Maximum filename size for save data files for PSP™ (PlayStation®Portable)

Definition

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_FILENAME_SIZE	13	Maximum filename size for save data files for PSP™

Description

This constant definition represents the upper limit for the filename to be saved to save data files for PSP™.

See Also

`SceAppUtilPspSaveDataParams`, `sceAppUtilPspSaveDataLoad()`

SCE CONFIDENTIAL

File ID Size for Protected Save Data Files for PSP™

File ID size for protected save data files for PSP™

Definition

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_SECUREFILEID_SIZE	16	File ID size for protected save data files for PSP™

Description

This constant definition represents the file ID size for protected data files required when reading out a protected data file that is stored in save data for PSP™.

See Also

SceAppUtilPspSaveDataParams, sceAppUtilPspSaveDataLoad()

SCE CONFIDENTIAL

Maximum Buffer Size for a Save Data File for PSP™

Maximum file buffer size to specify when processing save data for PSP™

Definition

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_DATA_BUF_SIZE_MAX	2*1024*1024	Maximum buffer size for a save data file for PSP™

Description

This constant definition represents the upper limit of the buffer size to be specified upon handling a save data file for PSP™. A file exceeding this size cannot be handled.

See Also

SceAppUtilPspSaveDataParams, sceAppUtilPspSaveDataLoad()

SCE CONFIDENTIAL

Maximum Number of Save Data for PSP™ to List

Maximum number of save data to handle for PSP™

Definition

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_DIRNAME_LIST_MAXNUM	256	Maximum number of save data for PSP™ that can be handled

Description

This constant definition represents the upper limit for the number of save data for PSP™ to specify in a list or to receive results for. Each processing will use this value to determine the maximum number of save data for PSP™ to handle.

See Also

SceAppUtilPspSaveDataParams, sceAppUtilPspSaveDataLoad(),
sceAppUtilPspSaveDataGetDirNameList()

SCE CONFIDENTIAL

Additional Content Directory Name Size

Size of the additional contents directory name

Definition

Value	(Number)	Description
SCE_APPUTIL_NP_DRM_ADDCONT_ID_SIZE	17	Additional content directory name size

Description

This is a constant definition indicating the size of the additional content directory name.

The additional content directory name is represented as a half-width alphanumeric character string of SCE_APPUTIL_NP_DRM_ADDCONT_ID_SIZE characters, including the NULL terminator.

SCE CONFIDENTIAL

Application Parameter ID

Application parameter ID constant

Definition

Value	(Number)	Description
SCE_APPUTIL_APPPARAM_ID_SKU_FLAG	0	SKU flag

Description

This is a constant definition used when obtaining application parameters. It is specified in the first argument of `sceAppUtilAppParamGetInt()`, and obtains the values of relevant application parameters.

SKU Flag

Constant representing SKU flag types

Definition

Value	(Number)	Description
SCE_APPUTIL_APPPARAM_SKU_FLAG_NONE	0	Off
SCE_APPUTIL_APPPARAM_SKU_FLAG_TRIAL	1	Trial
SCE_APPUTIL_APPPARAM_SKU_FLAG_FULL	3	Full

Description

These are constant definitions used when obtaining SKU flag types currently set in the application's rights information or the Settings application's **★Debug Settings -> PSN™ -> Upgradable App Debug**. By specifying SCE_APPUTIL_APPPARAM_ID_SKU_FLAG as the first argument of sceAppUtilAppParamGetInt(), one of the above-described values is stored in the second argument and returns.

Notes

For details on the SKU flag, refer to the "PSN™ Commerce Service Overview" document and the "PSN™ - Upgradable App Debug" section of the "★Debug Settings Functions" chapter in the "System Software Overview" document.

See Also

Application Parameter ID, SceAppUtilAppParamId, sceAppUtilAppParamGetInt()

SCE CONFIDENTIAL

Safe Memory Area Size

Size of the safe memory area

Definition

Value	(Number)	Description
SCE_APPUTIL_SAFEMEMORY_MEMORY_SIZE	64*1024	Size of the safe memory area

Description

This is a constant definition representing the safe memory area size.

The sum of the arguments *bufSize* and *offset*, specified in `sceAppUtilSaveSafeMemory()` and `sceAppUtilLoadSafeMemory()` must be SCE_APPUTIL_SAFEMEMORY_MEMORY_SIZE or less.

Store Browsing Type

Constants for specifying the store browsing type

Definition

Value	(Number)	Description
SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT2	3	Displays the product detail screen upon startup
SCE_APPUTIL_STORE_BROWSE_TYPE_CATEGORY2	4	Displays the category list screen upon startup
SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT_CODE2	5	Calls the promotion code obtaining feature

Description

These constant definitions are used when specifying the operation of the Title Store application. The operation type is set to the *type* member of the *param*, which is an argument of `sceAppUtilStoreBrowse()`.

See Also

`SceAppUtilStoreBrowseParam, sceAppUtilStoreBrowse()`

Internet Browser Application Startup Type

Constants for specifying the Internet Browser application startup type

Definition

Value	(Number)	Description
SCE_APPUTIL_WEBBROWSER_LAUNCH_APP_NORMAL	0	Starts up the normal Internet Browser application
SCE_APPUTIL_WEBBROWSER_LAUNCH_APP_MODAL	1	Starts up the modal Internet Browser application

Description

These constant definitions are used when specifying the Internet Browser application startup type. These are set to the *launchMode* member of the *param*, which is an argument of `sceAppUtilLaunchWebBrowser()`.

See Also

Internet Browser Application Startup Command Type, `SceAppUtilWebBrowserParam`, `sceAppUtilLaunchWebBrowser()`

Internet Browser Application Startup Command Type

Constants specifying the startup command type of the Internet Browser application

Definition

Value	(Number)	Description
SCE_APPUTIL_WEBBROWSER_LAUNCH_CMD_OPENURL	0	Opens the page specified by the URL
SCE_APPUTIL_WEBBROWSER_LAUNCH_CMD_SEARCH	4	Searches using the Internet search engine

Description

These are constant definitions for specifying the feature to execute when the Internet Browser application is started up. These are set to the *launchMode* member of the *param*, which is an argument of `sceAppUtilLaunchWebBrowser()`.

See Also

Internet Browser Application Startup Type, `SceAppUtilWebBrowserParam`, `sceAppUtilLaunchWebBrowser()`

SCE CONFIDENTIAL

Screenshot Trigger Type

Constants representing the screenshot trigger type for a screenshot capture notification event

Definition

Value	(Number)	Description
SCE_APPUTIL_SCREENSHOT_TRIGGER_TYPE_USER	0	User-operated trigger
SCE_APPUTIL_SCREENSHOT_TRIGGER_TYPE_IN_GAME	1	Game-initiated trigger

Description

This constant definition represents a screenshot trigger in a screenshot capture notification event. It is set to the *triggerType* member of the `SceAppUtilScreenShotNotification` structure.

See Also

`SceAppUtilScreenShotNotification`,
`sceAppUtilAppEventParseScreenShotNotification()`

SCE CONFIDENTIAL

Maximum Size of the Screenshot Content Path

Maximum size of the screenshot content path

Definition

Value	(Number)	Description
SCE_APPUTIL_SCREENSHOT_CONTENT_PATH_MAXSIZE	1024	Maximum size of the screenshot content path

Description

This is the maximum size of the screenshot content path.

It is the size of the *contentPath* member of the `SceAppUtilScreenShotNotification` structure.
This size includes the NULL terminator of the character string.

See Also

`SceAppUtilScreenShotNotification`,
`sceAppUtilAppEventParseScreenShotNotification()`

Types

000004892117

SCE CONFIDENTIAL

SceAppUtilBootAttribute

Boot-up attribute type

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilBootAttribute;
```

Description

This is a boot-up attribute type used when obtaining boot-up attributes. It is used as type of the *attr* member of the *SceAppUtilBootParam* structure. It is possible to obtain boot-up attributes at the time of application boot-up by specifying the *SceAppUtilBootParam* structure in the second argument of *sceAppUtilInit()*.

Notes

The value returned to the *attr* member of the *SceAppUtilBootParam* structure is always 0.

See Also

SceAppUtilBootParam, *sceAppUtilInit()*

SCE CONFIDENTIAL

SceAppUtilAppEventType

Type of event notification type

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilAppEventType;
```

Description

This is a type of event notification type used when obtaining event parameters. It is used as type of the *type* member of the *SceAppUtilAppEventParam* structure. It is possible to obtain the application event notification type and its notification parameters by specifying the *SceAppUtilAppEventParam* structure in an argument of *sceAppUtilReceiveAppEvent()*.

Also, by calling *sceAppUtilAppEventParseXxx()* with the obtained *SceAppUtilAppEventParam* structure specified as the first argument, the obtained notification parameters are properly parsed, and then the result will be stored in each structure specified in the second argument of the *sceAppUtilAppEventParseXxx()*.

See Also

Event Notification Type, Event Notification Parameter Size, *SceAppUtilAppEventParam*, *sceAppUtilReceiveAppEvent()*

SCE CONFIDENTIAL

SceAppUtilSaveDataSlotId

Save data slot ID type

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilSaveDataSlotId;
```

Description

This is the save data slot ID type specified when using save data slot functions. 0 to SCE_APPUTIL_SAVEDATA_SLOT_MAX-1 (integer value) can be specified for the save data slot IDs.

See Also

SceAppUtilSaveDataSlot, SceAppUtilSaveDataDataSlot,
sceAppUtilSaveDataSlotCreate(), sceAppUtilSaveDataSlotDelete(),
sceAppUtilSaveDataSlotSetParam(), sceAppUtilSaveDataSlotGetParam()

SCE CONFIDENTIAL

SceAppUtilSaveDataSlotStatus

Save data slot status type

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilSaveDataSlotStatus;
```

Description

This is the save data slot status type indicating the transaction status of a save data slot.

It is used as the type of the *status* member of the `SceAppUtilSaveDataSlotParam` structure. By comparing the constant for save data slot status indicated with `SCE_APPUTIL_SAVEDATA_SLOT_STATUS_XXX` with the save data slot parameter obtained with `sceAppUtilSaveDataSlotGetParam()`, it is possible to know the status of the relevant save data slot.

See Also

Save Data Slot Status, `SceAppUtilSaveDataSlotParam`, `SceAppUtilSaveDataSlot`, `sceAppUtilSaveDataSlotGetParam()`

SCE CONFIDENTIAL

SceAppUtilSaveDataDataSaveMode

Data save mode type for save data

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilSaveDataDataSaveMode;
```

Description

This type is used to specify the data save mode for save data.

It is used as the type of the *mode* member of the *SceAppUtilSaveDataDataSaveItem* structure.

The operation of *sceAppUtilSaveDataDataSave()*, a function for saving save data, changes based on the specified data save mode for save data indicated with *SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_XXX*.

See Also

Data Save Mode for Save Data, *SceAppUtilSaveDataDataSaveItem*,
sceAppUtilSaveDataDataSave()

SCE CONFIDENTIAL

SceAppUtilSaveDataDataRemoveMode

Data removal mode type for save data

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilSaveDataDataRemoveMode;
```

Description

This type is used to specify the data removal mode for save data.

It is used as the type of the *mode* member of the *SceAppUtilSaveDataDataRemoveItem* structure.

The operation of *sceAppUtilSaveDataDataRemove()*, a function for removing save data, changes based on the specified data removal mode for save data indicated with

SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_XXX.

See Also

Data Removal Mode for Save Data, *SceAppUtilSaveDataDataRemoveItem*,
sceAppUtilSaveDataDataRemove()

SCE CONFIDENTIAL

SceAppUtilPspSaveDataVersion

Format version of save data for PSP™

Definition

```
#include <apputil/apputil_psp.h>
typedef enum {
    SCE_APPUTIL_PSP_SAVEDATA_VERSION_0 = 0,
    SCE_APPUTIL_PSP_SAVEDATA_VERSION_1,
    SCE_APPUTIL_PSP_SAVEDATA_VERSION_2
} SceAppUtilPspSaveDataVersion;
```

Enumeration Values

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_VERSION_0	0	Format of PSP™ Runtime Library release 1.0.3/1.5.0/1.5.2
SCE_APPUTIL_PSP_SAVEDATA_VERSION_1	1	Format of PSP™ Runtime Library release 2.0.0/2.5.0/2.6.0
SCE_APPUTIL_PSP_SAVEDATA_VERSION_2	2	Format of PSP™ Runtime Library release 2.7.1 and later

Description

This type is used to specify the version of the save data for PSP™ format.

There are three format versions to a save data for PSP™ according to the timing at which the title was released.

See Also

SceAppUtilPspSaveDataParams, sceAppUtilPspSaveDataLoad()

SCE CONFIDENTIAL

SceAppUtilPspSaveDataType

Save data for PSP™ file type

Definition

```
#include <apputil/apputil_psp.h>
typedef enum {
    SCE_APPUTIL_PSP_SAVEDATA_TYPE_SECUREFILE,
    SCE_APPUTIL_PSP_SAVEDATA_TYPE_NORMALFILE
} SceAppUtilPspSaveDataType;
```

Enumeration Values

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_TYPE_SECUREFILE	0	Protected data file
SCE_APPUTIL_PSP_SAVEDATA_TYPE_NORMALFILE	1	Normal data file

Description

This type is used to specify the save data for PSP™ file type. A protected data file is encrypted and formatted so that any tampering can be detected. A normal file is saved as plain text and any tampering cannot be detected.

See Also

SceAppUtilPspSaveDataParams, sceAppUtilPspSaveDataLoad()

SCE CONFIDENTIAL

SceAppUtilPspSaveDataParamSfoSize

Size of each parameter to set to PARAM.SFO

Definition

```
#include <apputil/apputil_psp.h>
typedef enum {
    SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE = 32,
    SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_TITLE_SIZE = 128,
    SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_SD_TITLE_SIZE = 128,
    SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DETAIL_SIZE = 1024
} SceAppUtilPspSaveDataParamSfoSize;
```

Enumeration Values

Value	(Number)	Description
SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE	32	Size of the directory name
SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_TITLE_SIZE	128	Size of the title
SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_SD_TITLE_SIZE	128	Size of the save data title
SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DETAIL_SIZE	1024	Size of detailed information

Description

This type defines the sizes of each parameter of the save data for PSP™ system file (PARAM.SFO).

SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE is the size of the *saveDataDirectory* member variable in the *SceAppUtilPspSaveDataParamSfo* structure. This size includes the NULL-terminator.

This size is also used as the size of the *data* member variable in the *SceAppUtilPspSaveDataDirName* structure.

SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_TITLE_SIZE is the size of the *title* member variable in the *SceAppUtilPspSaveDataParamSfo* structure. This size includes the NULL-terminator.

SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_SD_TITLE_SIZE is the size of the *saveDataTitle* member variable in the *SceAppUtilPspSaveDataParamSfo* structure. This size includes the NULL-terminator.

SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DETAIL_SIZE is the size of the *detail* member variable in the *SceAppUtilPspSaveDataParamSfo* structure. This size includes the NULL-terminator.

See Also

SceAppUtilPspSaveDataParams, *SceAppUtilPspSaveDataDirName*,
sceAppUtilPspSaveDataGetDirNameList(), *sceAppUtilPspSaveDataLoad()*

SCE CONFIDENTIAL

SceAppUtilAppParamId

Application parameter ID type

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilAppParamId;
```

Description

This is an application parameter ID type used when obtaining application parameters.

It is used as the type of *paramId*, the first argument of `sceAppUtilAppParamGetInt()`, a function for obtaining application parameters. In the *paramId* argument, specify the application parameter ID indicated with `SCE_APPUTIL_APPPARAM_ID_XXX`.

See Also

Application Parameter ID, `sceAppUtilAppParamGetInt()`

SceAppUtilScreenShotTriggerType

Type of the screenshot trigger type

Definition

```
#include <apputil.h>
typedef SceUInt32 SceAppUtilScreenShotTriggerType;
```

Description

This type represents the screenshot trigger type used in a screenshot capture notification event.

It is used as the *triggerType* type member of the *SceAppUtilScreenShotNotification* structure. The event parameters of a screenshot capture notification can be obtained by specifying the *SceAppUtilScreenShotNotification* structure as an argument of *sceAppUtilAppEventParseScreenShotNotification()*.

See Also

SceAppUtilScreenShotNotification,
sceAppUtilAppEventParseScreenShotNotification()

Structures

000004892117

SCE CONFIDENTIAL

SceAppUtilInitParam

Structure for storing initial library settings

Definition

```
#include <apputil.h>
typedef struct SceAppUtilInitParam {
    SceSize workBufSize;
    SceChar8 reserved[60];
} SceAppUtilInitParam;
```

Members

<i>workBufSize</i>	Memory size used in the library
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure represents parameters set for the first argument of the initialization function `sceAppUtilInit()`, which is required for using the application utility library. Specify 0 to *workBufSize*. If a value other than 0 is specified, a parameter error will return.

See Also

`sceAppUtilInit()`

SCE CONFIDENTIAL

SceAppUtilBootParam

Structure for boot-up parameter obtaining

Definition

```
#include <apputil.h>
typedef struct SceAppUtilBootParam {
    SceAppUtilBootAttribute attr;
    SceUInt32 appVersion;
    SceChar8 reserved[32];
} SceAppUtilBootParam;
```

Members

<i>attr</i>	Boot-up attribute
<i>appVersion</i>	Version of the application
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure is specified in the second argument of the initialization function `sceAppUtilInit()` when using the application utility library. The application's boot-up attributes and version information are stored in this structure after returning from calling `sceAppUtilInit()`.

Notes

The values returned to the *attr* and *appVersion* members of the `SceAppUtilBootParam` structure are always 0.

See Also

`SceAppUtilBootAttribute`, `sceAppUtilInit()`

SCE CONFIDENTIAL

SceAppUtilAppEventParam

Structure for event parameter obtaining

Definition

```
#include <apputil.h>
typedef struct SceAppUtilAppEventParam {
    SceAppUtilAppEventType type;
    SceUChar8 data[SCE_APPUTIL_APPEVENT_PARAM_MAXSIZE];
} SceAppUtilAppEventParam;
```

Members

type Event notification type (either one of SCE_APPUTIL_APPEVENT_TYPE_XXX)
data Event notification parameter

Description

This is a structure specified in an argument of the `sceAppUtilReceiveAppEvent()` function that is used to obtain the event notification type and event notification parameter. The application's event notification type and its notification parameter are stored in this structure after returning from calling `sceAppUtilReceiveAppEvent()`.

Also, by calling `sceAppUtilAppEventParseXxx()` function with the obtained `SceAppUtilAppEventParam` structure specified as the first argument, the obtained notification parameter is properly parsed, and then the result will be stored in each structure specified in the second argument of the function.

See Also

Event Notification Type, Event Notification Parameter Size, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseNpInviteMessage()`, `sceAppUtilAppEventParseNpAppDataMessage()`, `sceAppUtilAppEventParseNearGift()`, `sceAppUtilAppEventParseSessionInvitation()`, `sceAppUtilAppEventParseGameCustomData()`

SCE CONFIDENTIAL

SceAppUtilWorkBuffer

Structure for the work buffer area

Definition

```
#include <apputil.h>
typedef struct SceAppUtilWorkBuffer {
    void *buf;
    SceUInt32 bufSize;
    SceChar8 reserved[32];
} SceAppUtilWorkBuffer;
```

Members

<i>buf</i>	Beginning pointer of the buffer area
<i>bufSize</i>	Buffer size
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure specifies the work buffer used in various application utility APIs.

To the *buf* member variable, specify the beginning pointer of the allocated work buffer area. To the *bufSize* member variable, specify the buffer size of the allocated work buffer area.

See Also

Work Buffer Size for a Save Data Slot Search, `sceAppUtilSaveDataSlotSearch()`

SCE CONFIDENTIAL

SceAppUtilNpInviteMessageParam

Structure for invitation message obtaining

Definition

```
#include <apputil.h>
typedef struct SceAppUtilNpInviteMessageParam {
    SceNpCommunicationId commId;
    SceUChar8 uid[SCE_NP_MESSAGE_MAX_MESSAGE_ID_SIZE+1];
} SceAppUtilNpInviteMessageParam;
```

Members

<i>commId</i>	Communication ID
<i>uid</i>	Message ID (NULL terminated, US-ASCII)

Description

This is a structure specified in the second argument of the `sceAppUtilAppEventParseNpInviteMessage()` function that is used to obtain the invitation message parameter. By calling the function in the case that `SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, the event notification parameter stored in the *data* member of `SceAppUtilAppEventParam` structure is parsed, and then the result will be stored in the `SceAppUtilNpInviteMessageParam` structure specified in the second argument. A parameter error will be returned if `sceAppUtilAppEventParseNpInviteMessage()` is called in the state where another value than `SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE` is specified in the *type* member of the `SceAppUtilAppEventParam` structure.

Notes

For the `SceNpCommunicationId` structure, refer to the "NP Library Reference" document and for the `SCE_NP_MESSAGE_MAX_MESSAGE_ID_SIZE` constant, refer to the "NP Message Reference" document.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseNpInviteMessage()`

SCE CONFIDENTIAL

SceAppUtilNpAppDataMessageParam

Structure for obtaining message with game data attached

Definition

```
#include <apputil.h>
typedef struct SceAppUtilNpAppDataMessageParam {
    SceNpCommunicationId commId;
    SceUChar8 uid[SCE_NP_MESSAGE_MAX_MESSAGE_ID_SIZE+1];
} SceAppUtilNpAppDataMessageParam;
```

Members

<i>commId</i>	Communication ID
<i>uid</i>	Message ID (NULL terminated, US-ASCII)

Description

This is a structure specified in the second argument of the `sceAppUtilAppEventParseNpAppDataMessage()` function that is used to obtain the parameter of a message with game data attached. By calling the function in the case that `SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, the event notification parameter stored in the *data* member of `SceAppUtilAppEventParam` structure is parsed, and then the result will be stored in the `SceAppUtilNpAppDataMessageParam` structure specified in the second argument. A parameter error will be returned if `sceAppUtilAppEventParseNpAppDataMessage()` is called in the state where another value than `SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE` is specified in the *type* member of the `SceAppUtilAppEventParam` structure.

Notes

For the `SceNpCommunicationId` structure, refer to the "NP Library Reference" document and for the `SCE_NP_MESSAGE_MAX_MESSAGE_ID_SIZE` constant, refer to the "NP Message Reference" document.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseNpAppDataMessage()`

SCE CONFIDENTIAL

SceAppUtilNearGiftParam

Structure for "near" gift obtaining

Definition

```
#include <apputil.h>
typedef struct SceAppUtilNearGiftParam {
    SceNpCommunicationId commId;
    SceNearGiftId giftId;
    SceNpId npId;
    SceUInt32 version;
    SceUChar8 giftData[SCE_NEAR_GIFT_DATA_PARAM_MAX_SIZE];
} SceAppUtilNearGiftParam;
```

Members

<i>commId</i>	Communication ID
<i>giftId</i>	Gift ID
<i>npId</i>	NP ID
<i>version</i>	Version of gift
<i>giftData</i>	Starting part of the gift

Description

This is a structure specified in the second argument of the `sceAppUtilAppEventParseNearGift()` function that is used to obtain the "near" gift parameter. By calling the function in the case that `SCE_APPUTIL_APPEVENT_TYPE_NEAR_GIFT` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, the event notification parameter stored in the *data* member of `SceAppUtilAppEventParam` structure is parsed, and then the result will be stored in the `SceAppUtilNearGiftParam` structure specified in the second argument. A parameter error will be returned if `sceAppUtilAppEventParseNearGift()` is called in the state where another value than `SCE_APPUTIL_APPEVENT_TYPE_NEAR_GIFT` is specified in the *type* member of the `SceAppUtilAppEventParam` structure.

Notes

For the `SceNpCommunicationId` and `SceNpId` structures, refer to the "NP Library Reference" document. For the `SCE_NEAR_GIFT_DATA_PARAM_MAX_SIZE` constant, refer to the "near Utility Reference" document.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseNearGift()`

SCE CONFIDENTIAL

SceAppUtilLiveAreaParam

Structure for obtaining parameter of application startup from LiveArea™

Definition

```
#include <apputil.h>
typedef struct SceAppUtilLiveAreaParam {
    SceUChar8 param[SCE_APPUTIL_LIVEAREA_PARAM_MAX_SIZE];
    SceChar8 reserved[32];
} SceAppUtilLiveAreaParam;
```

Members

<i>param</i>	Event parameter (NULL terminated, US-ASCII)
<i>reserved</i>	Reserved area

Description

This is a structure specified in the second argument of the `sceAppUtilAppEventParseLiveArea()` function that is used to obtain parameter of application startup from LiveArea™. By calling the function in the case that `SCE_APPUTIL_APPEVENT_TYPE_LIVEAREA` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, the event notification parameter stored in the *data* member of `SceAppUtilAppEventParam` structure is parsed, and then the result will be stored in the `SceAppUtilLiveAreaParam` structure specified in the second argument. A parameter error will be returned if `sceAppUtilAppEventParseLiveArea()` is called in the state where another value than `SCE_APPUTIL_APPEVENT_TYPE_LIVEAREA` is specified in the *type* member of the `SceAppUtilAppEventParam` structure.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseLiveArea()`

SCE CONFIDENTIAL

SceAppUtilScreenShotNotification

Structure for obtaining screenshot capture notification event parameters

Definition

```
#include <apputil.h>
typedef struct SceAppUtilScreenShotNotification{
    SceAppUtilScreenShotTriggerType triggerType;
    SceUChar8 contentPath[SCE_APPUTIL_SCREENSHOT_CONTENT_PATH_MAXSIZE];
    SceChar8 reserved[32];
} SceAppUtilScreenShotNotification;
```

Members

triggerType Screenshot trigger type
contentPath Content path (NULL terminated)
reserved Reserved area

Description

This structure is specified to the second argument of the `sceAppUtilAppEventParseScreenShotNotification()` function, which obtains screenshot capture notification event parameters.

When `SCE_APPUTIL_APPEVENT_TYPE_SCREENSHOT_NOTIFICATION` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, call the `sceAppUtilAppEventParseScreenShotNotification()` function, parse the event notification parameters stored in the *data* member of the `SceAppUtilAppEventParam` structure and store the results in this structure (specified as the second argument).

When `sceAppUtilAppEventParseScreenShotNotification()` is called when a value other than `SCE_APPUTIL_APPEVENT_TYPE_SCREENSHOT_NOTIFICATION` is specified to the *type* member of the `SceAppUtilAppEventParam` structure, the function returns a parameter error.

To access content in the *contentPath*, call the photo mount feature, `sceAppUtilPhotoMount()`, and carry out mount processing of the photo directory.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseScreenShotNotification()`, `sceAppUtilPhotoMount()`, `sceAppUtilPhotoUmount()`

SCE CONFIDENTIAL

SceAppUtilNpActivityParam

Structure for obtaining application startup parameters from an activity

Definition

```
#include <apputil.h>
typedef struct SceAppUtilNpActivityParam {
    SceUChar8 param[SCE_APPUTIL_NP_ACTIVITY_PARAM_MAXSIZE];
    SceChar8 reserved[32];
} SceAppUtilNpActivityParam;
```

Members

param Event parameter (NULL terminated)
reserved Reserved area

Description

This structure is specified for the second argument of the function `sceAppUtilAppEventParseNpActivity()` that obtains application startup parameters from an activity.

When `SCE_APPUTIL_APPEVENT_TYPE_NP_ACTIVITY` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, call the `sceAppUtilAppEventParseNpActivity()` function, parse the event notification parameters stored in the *data* member of the `SceAppUtilAppEventParam` structure and store the results in this structure (specified as the second argument). When `sceAppUtilAppEventParseNpActivity()` is called when a value other than `SCE_APPUTIL_APPEVENT_TYPE_NP_ACTIVITY` is specified to the *type* member of the `SceAppUtilAppEventParam` structure, the function returns a parameter error.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseNpActivity()`

SCE CONFIDENTIAL

SceAppUtilTeleportParam

Structure for obtaining application startup parameters from the Teleport library

Definition

```
#include <apputil.h>
typedef struct SceAppUtilTeleportParam{
    SceChar8 param[SCE_APPUTIL_TELEPORT_PARAM_MAXSIZE];
    SceChar8 reserved[32];
} SceAppUtilTeleportParam;
```

Members

param Event parameter (NULL terminated)
reserved Reserved area

Description

This structure is specified for the second argument of the function `sceAppUtilAppEventParseTeleport()` that obtains application startup parameters from the Teleport library.

When `SCE_APPUTIL_APPEVENT_TYPE_TELEPORT` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, call the `sceAppUtilAppEventParseTeleport()` function, parse the event notification parameters stored in the *data* member of the `SceAppUtilAppEventParam` structure and store the results in this structure (specified as the second argument). When `sceAppUtilAppEventParseTeleport()` is called when a value other than `SCE_APPUTIL_APPEVENT_TYPE_TELEPORT` is specified to the *type* member of the `SceAppUtilAppEventParam` structure, the function returns a parameter error.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseTeleport()`

SCE CONFIDENTIAL

SceAppUtilSessionInvitationParam

Structure for obtaining session invitation

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSessionInvitationParam {
    SceNpSessionId sessionId;
    SceNpInvitationId invitationId;
} SceAppUtilSessionInvitationParam;
```

Members

sessionId Session ID (NULL terminated, US-ASCII)
invitationId Invitation ID (NULL terminated, US-ASCII)

Description

This structure is specified for the second argument of the function `sceAppUtilAppEventParseSessionInvitation()` that obtains session invitation parameters. When `SCE_APPUTIL_APPEVENT_TYPE_SESSION_INVITATION` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, call the `sceAppUtilAppEventParseSessionInvitation()` function, parse the event notification parameters stored in the *data* member of the `SceAppUtilAppEventParam` structure, and store the results in this structure (specified as the second argument). When `sceAppUtilAppEventParseSessionInvitation()` is called when a value other than `SCE_APPUTIL_APPEVENT_TYPE_SESSION_INVITATION` is specified to the *type* member of the `SceAppUtilAppEventParam` structure, the function returns a parameter error.

Notes

Regarding the session ID and invitation ID, refer to the "Session/Invitation Overview" document.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseSessionInvitation()`

SCE CONFIDENTIAL

SceAppUtilGameCustomDataParam

Structure for obtaining game custom data

Definition

```
#include <apputil.h>
typedef struct SceAppUtilGameCustomDataParam {
    SceNpGameCustomDataId gameCustomDataId;
} SceAppUtilGameCustomDataParam;
```

Members

gameCustomDataId Game custom data ID (maximum value: 9223372036854775807)

Description

This structure is specified for the second argument of the function `sceAppUtilAppEventParseGameCustomData()` that obtains game custom data parameters. When `SCE_APPUTIL_APPEVENT_TYPE_GAME_CUSTOM_DATA` is stored in the *type* member of the `SceAppUtilAppEventParam` structure, call the `sceAppUtilAppEventParseGameCustomData()` function, parse the event notification parameters stored in the *data* member of the `SceAppUtilGameCustomDataParam` structure, and store the results in this structure (specified as the second argument). When `sceAppUtilAppEventParseGameCustomData()` is called when a value other than `SCE_APPUTIL_APPEVENT_TYPE_GAME_CUSTOM_DATA` is specified to the *type* member of the `SceAppUtilAppEventParam` structure, the function returns a parameter error.

Notes

Regarding the game custom data ID, refer to the "Game Custom Data Overview" document.

See Also

Event Notification Type, `SceAppUtilAppEventType`, `sceAppUtilReceiveAppEvent()`, `sceAppUtilAppEventParseGameCustomData()`

SCE CONFIDENTIAL

SceAppUtilMountPoint

Mount point structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilMountPoint {
    SceChar8 data[SCE_APPUTIL_MOUNTPOINT_DATA_MAXSIZE];
} SceAppUtilMountPoint;
```

Members

data Mount point name

Description

This structure represents the mount point. It is used to specify the processing-target mount point to a function that supports the specification of a mount point.

See Also

Maximum Size of a Mount Point Name, `sceAppUtilSaveDataSlotCreate()`,
`sceAppUtilSaveDataSlotDelete()`, `sceAppUtilSaveDataSlotSetParam()`,
`sceAppUtilSaveDataSlotGetParam()`, `sceAppUtilSaveDataSlotSearch()`,
`sceAppUtilSaveDataDataSave()`, `sceAppUtilSaveDataDataRemove()`,
`sceAppUtilSaveDataGetQuota()`, `sceAppUtilDrmOpen()`, `sceAppUtilDrmClose()`

SCE CONFIDENTIAL

SceAppUtilTitleId

Title ID structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilTitleId {
    SceChar8 data[SCE_APPUTIL_TITLE_ID_DATA_SIZE];
    SceChar8 padding[6];
} SceAppUtilTitleId;
```

Members

<i>data</i>	Title ID
<i>padding</i>	Reserved area

Description

This structure represents the title ID. It is used when specifying a processing-target application.

Notes

Initialize all reserved area with 0s.

See Also

Title ID Size, `sceAppUtilSaveDataMount()`, `sceAppUtilAddcontMount()`

SCE CONFIDENTIAL

SceAppUtilPassCode

Passcode structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilPassCode {
    SceChar8 data[SCE_APPUTIL_PASSCODE_DATA_SIZE];
} SceAppUtilPassCode;
```

Members

data Passcode

Description

This structure represents a passcode. It is used in processing that requires a passcode.

See Also

Passcode Size, `sceAppUtilSaveDataMount()`, `sceAppUtilAddcontMount()`

SCE CONFIDENTIAL

SceAppUtilSaveDataSlotParam

Save data slot parameter structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataSlotParam {
    SceAppUtilSaveDataSlotStatus status;
    SceChar8 title[SCE_APPUTIL_SAVEDATA_SLOT_TITLE_MAXSIZE];
    SceChar8 subTitle[SCE_APPUTIL_SAVEDATA_SLOT_SUBTITLE_MAXSIZE];
    SceChar8 detail[SCE_APPUTIL_SAVEDATA_SLOT_DETAIL_MAXSIZE];
    SceChar8 iconPath[SCE_APPUTIL_SAVEDATA_SLOT_ICON_PATH_MAXSIZE];
    SceInt32 userParam;
    SceUInt32 sizeKiB;
    SceDateTime modifiedTime;
    SceChar8 reserved[48];
} SceAppUtilSaveDataSlotParam;
```

Members

<i>status</i>	Save data slot status
<i>title</i>	Title name (NULL terminated, UTF-8)
<i>subTitle</i>	Subtitle name (NULL terminated, UTF-8)
<i>detail</i>	Detail information (NULL terminated, UTF-8)
<i>iconPath</i>	Thumbnail icon path (NULL terminated)
<i>userParam</i>	User parameter
<i>sizeKiB</i>	Data size (unit: KiB)
<i>modifiedTime</i>	Last modification date
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure represents one specific save data slot parameters.

It is used when creating save data slots, setting or getting save data slot parameters, etc. It is also used when updating save data slot parameters in synchronization with save data saving/removing.

The save data transaction status (normal status/broken status) is recorded to the member variable *status*.

Set character strings that can be represented in UTF-8 character code as the character strings specified in member variables *title*, *subTitle* and *detail*. If they contain characters that cannot be represented in UTF-8 character code, or if a linefeed is included in a *title* or *subTitle*, a parameter error will occur when creating/setting save data slot parameters.

For the character strings specified in the member variable *iconPath*, [a-z], [A-Z], [0-9], '-' (hyphen), '_' (underscore), '.' (period), '/' (slash), and ':' (colon) can be used. Because the member variable

Applications can insert an arbitrary value into the member variable *userParam*. This value will not be shown to users. This value can also be used to perform slot searches.

When a value is set for the member variable *sizeKiB*, the set value will be displayed as the save data size when displaying the slot with Save Data Dialog. When 0 is set, size display will not be performed. If there are not particular requests to perform size display, set 0.

For the member variable *modifiedTime*, the time (local time) when the save data slot was updated will be automatically set by the library. Applications do not need to set this value themselves. When displaying the set value with Save Data Dialog, it will be displayed in the format that corresponds to the system software time display settings.

See Also

Save Data Slot Parameter Sizes, Save Data Slot Status, `SceAppUtilSaveDataSlotStatus`,
`SceAppUtilSaveDataDataSlot`, `sceAppUtilSaveDataSlotCreate()`,
`sceAppUtilSaveDataSlotSetParam()`, `sceAppUtilSaveDataSlotGetParam()`,
`sceAppUtilSaveDataDataSave()`, `sceAppUtilSaveDataDataRemove()`

000004892117

SceAppUtilSaveDataSlotEmptyParam

Empty save data slot parameter specifying structure (for Save Data Dialog)

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataSlotEmptyParam {
    SceChar8 *title;
    SceChar8 *iconPath;
    void *iconBuf;
    SceSize iconBufSize;
    SceChar8 reserved[32];
} SceAppUtilSaveDataSlotEmptyParam;
```

Members

<i>title</i>	Title character string displayed in empty save data slot (NULL terminated, UTF-8)
<i>iconPath</i>	Specification of thumbnail image file displayed in an empty save data slot
<i>iconBuf</i>	Buffer of thumbnail image displayed in an empty save data slot
<i>iconBufSize</i>	Buffer size of thumbnail image displayed in an empty save data slot
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure represents empty save data slot parameter for Save Data Dialog.

Specify the title name string displayed in the empty save data slot for the member variable *title*.

Specify the file path, buffer, and buffer size of thumbnail images displayed in the empty save data slot for member variables *iconPath*, *iconBuf*, and *iconBufSize*, respectively. If the thumbnail image is specified with a file path, the *iconPath* value must be set, while the *iconBuf* and *iconBufSize* values must be set when specifying via buffer.

Notes

For details on parameters that can be specified for Save Data Dialog, refer to the "Save Data Dialog Overview" and "Save Data Dialog Reference" documents.

SceAppUtilSaveDataSlotSearchCond

Structure representing save data slot search conditions (for Save Data Dialog)

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataSlotSearchCond {
    SceAppUtilSaveDataSlotSearchType type;
    SceAppUtilSaveDataSlotId from;
    SceUInt32 range;
    SceAppUtilSaveDataSlotSortKey key;
    SceAppUtilSaveDataSlotSortType order;
    SceChar8 reserved[32];
} SceAppUtilSaveDataSlotSearchCond;
```

Members

type Save data slot search target
from Save data slot ID to start search from
range Search range
key Save data slot search key
order Save data slot search sort order
reserved Reserved area (fill with 0s)

Description

When searching save data slots, this structure is used to represent the search conditions.

For the *type* member variable, specify the save data slot search target of the `SceAppUtilSaveDataSlotSearchType` type. Enter one of the following values.

Value	(Number)	Description
<code>SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EXIST_SLOT</code>	0	Targets existing slots
<code>SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EMPTY_SLOT</code>	1	Targets empty slots

For the *from* member variable, specify the save data slot ID of the `SceAppUtilSaveDataSlotId` type.

For the *range* member variable, specify the number of slots (search range) using the `SceUInt32` type. Beginning with the *from* member variable, the save data slots covering the range specified by the *range* member variable will be the search target.

For the *key* member variable, specify the save data slot search sort key of the `SceAppUtilSaveDataSlotSortKey` type. Enter one of the following values.

Value	(Number)	Description
<code>SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SLOT_ID</code>	0	Uses save data slot ID as the sort key
<code>SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_USER_PARAM</code>	1	Uses the user parameter as the sort key
<code>SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SIZE_KIB</code>	2	Uses data size as the sort key
<code>SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_MODIFIED_TIME</code>	3	Uses the last update date as the sort key

When specifying `SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EMPTY_SLOT` to the *type* member variable, only the `SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SLOT_ID` value can be specified to *key*; any other value will cause a parameter error.

SCE CONFIDENTIAL

For the *order* member variable, specify the save data slot search sort order of the `SceAppUtilSaveDataSlotSortType` type. Enter one of the following values.

Value	(Number)	Description
<code>SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_ASCENT</code>	0	Ascending order
<code>SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_DESCENT</code>	1	Descending order

See Also

`sceAppUtilSaveDataSlotSearch()`

SCE CONFIDENTIAL

SceAppUtilSlotSearchResult

Structure for storing the save data slot search results (for Save Data Dialog)

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSlotSearchResult {
    SceUInt32 hitNum;
    SceAppUtilSaveDataSlot *slotList;
    SceChar8 reserved[32];
} SceAppUtilSlotSearchResult;
```

Members

hitNum Search result (number of slots)
slotList Save data slot structure (for Save Data Dialog)
reserved Reserved area (fill with 0s)

Description

This structure represents the results of a save data slot search.

For the *hitNum* member variable, the number of save data slots obtained as a result of the search will be stored.

For the *slotList* member variable, the beginning pointer of the array of save data slots obtained as a result of the search will be stored. The storing area of these save data slots is the same as the work buffer area of the *SceAppUtilWorkBuffer* structure specified as the first argument of *sceAppUtilSaveDataSlotSearch()*.

See Also

SceAppUtilSaveDataSlot, *SceAppUtilWorkBuffer*, *sceAppUtilSaveDataSlotSearch()*

SceAppUtilSaveDataSlot

Save data slot parameter structure (for Save Data Dialog)

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataSlot {
    SceAppUtilSaveDataSlotId id;
    SceAppUtilSaveDataSlotStatus status;
    SceInt32 userParam;
    SceAppUtilSaveDataSlotEmptyParam *emptyParam;
} SceAppUtilSaveDataSlot;
```

Members

<i>id</i>	Save data slot ID
<i>status</i>	Save data slot status
<i>userParam</i>	User parameter
<i>emptyParam</i>	Empty save data slot parameter specification structure

Description

This structure represents a specific save data slot from among those in Save Data Dialog. Specify a save data slot ID of the `SceAppUtilSaveDataSlotId` type for the member variable *id*. For the member variable *status*, specify a save data slot status of the `SceAppUtilSaveDataSlotStatus` type. For *userParam* and *emptyParam*, specify a user parameter and an empty save data slot parameter for Save Data Dialog, respectively.

Notes

For details on Save Data Dialog, refer to the "Save Data Dialog Overview" and "Save Data Dialog Reference" documents.

See Also

`SceAppUtilSaveDataSlotId`, `SceAppUtilSaveDataSlotStatus`, `SceAppUtilSaveDataSlotEmptyParam`

SCE CONFIDENTIAL

SceAppUtilSaveDataDataSaveItem

Structure for specifying the save-target save data

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataDataSaveItem {
    const SceChar8 *dataPath;
    const void *buf;
    SceSize bufSize;
    SceChar8 padding[4];
    SceOff offset;
    SceAppUtilSaveDataDataSaveMode mode;
    SceChar8 reserved[36];
} SceAppUtilSaveDataDataSaveItem;
```

Members

<i>dataPath</i>	Data path of the target save data
<i>buf</i>	Buffer storing the contents of the save data to be saved when saving file
<i>bufSize</i>	Size of the save data to be saved when saving file
<i>padding</i>	Padding area (fill with 0s)
<i>offset</i>	Writing offset position from the start of the file for the target save data when saving file
<i>mode</i>	Specify the data save mode of the save data to be saved
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure represents the information of one specific save data (in file/directory units) to be saved.

Specify the target save data data path for the member variable *dataPath*. A directory of up to four levels deep can be specified for the path in *dataPath* (for example "directory1/directory2/directroy3/data4.dat"). Specify the file path and omit mount point names such as "savedata0:", etc. Also, the length of the file path must be no more than SCE_APPUTIL_SAVEDATA_DATA_PATH_MAXSIZE, including the NULL terminator. [a-z], [A-Z], [0-9], '-' (hyphen), '_' (underscore), '.' (period), and '/' (slash) can be used. Relative reference using ".." is not possible. Also, the length of each directory name and file name specified in the data path must be up to 64 characters, including the NULL terminator.

For the member variable *buf*, specify the address of the buffer storing the contents of the save data to be saved. For the member variable *bufSize*, specify the size of the buffer.

For the member variable *offset*, specify the writing offset position from the start of file when saving the save data contents specified in *buf*. For the member variable *mode*, specify the save data data save mode. If SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_DIRECTORY is specified, NULL must be set in the member variable *buf*, and 0 must be set in the member variables *bufSize* and *offset*.

See Also

Maximum Data Path Size for Save Data, Data Save Mode for Save Data,
SceAppUtilSaveDataDataSaveMode, sceAppUtilSaveDataDataSave()

SceAppUtilSaveDataDataRemoveItem

Structure for specifying the removal-target save data

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataDataRemoveItem {
    const SceChar8 *dataPath;
    SceAppUtilSaveDataDataRemoveMode mode;
    SceChar8 reserved[36];
} SceAppUtilSaveDataDataRemoveItem;
```

Members

dataPath Path of the removal-target save data
mode Specify the data removal mode of the save data to be removed
reserved Reserved area (fill with 0s)

Description

This structure represents the information of one specific save data (in file/directory units) to be removed.

In the member variable *dataPath*, specify the path of the removal-target save data. Limitations on the path of the removal-target save data are equivalent to those for the *dataPath* member of the *SceAppUtilSaveDataDataSaveItem* structure. Specify the data remove mode of the save data in the member variable *mode*.

See Also

Maximum Data Path Size for Save Data, Data Removal Mode for Save Data,
SceAppUtilSaveDataDataRemoveMode, *sceAppUtilSaveDataDataRemove()*

SceAppUtilSaveDataDataSlot

Save data data slot structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilSaveDataDataSlot {
    SceAppUtilSaveDataSlotId id;
    SceAppUtilSaveDataSlotParam *slotParam;
    SceChar8 reserved[32];
} SceAppUtilSaveDataDataSlot;
```

Members

<i>id</i>	Save data slot ID
<i>slotParam</i>	Save data slot parameter structure
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure stores a save data slot ID along with parameters of the save data slot.

Specify a save data slot ID of the `SceAppUtilSaveDataSlotId` type for the member variable *id*.

For the member variable *slotParam*, specify a pointer to the save data slot parameter structure of the `SceAppUtilSaveDataSlotParam` type. This structure is used when save data slot parameters are updated in synchronization with saving/removing of save data.

See Also

`SceAppUtilSaveDataSlotId`, `SceAppUtilSaveDataSlotParam`,
`sceAppUtilSaveDataDataSave()`, `sceAppUtilSaveDataDataRemove()`

SCE CONFIDENTIAL

SceAppUtilPspSaveDataParamSfo

Structure of the save data system file for PSP™ (PARAM.SFO)

Definition

```
#include <apputil/apputil_psp.h>
typedef struct SceAppUtilPspSaveDataParamSfo {
    SceChar8 saveDataDirectory
        [SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE];
    SceChar8 title
        [SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_TITLE_SIZE];
    SceChar8 saveDataTitle
        [SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_SD_TITLE_SIZE];
    SceChar8 detail
        [SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DETAIL_SIZE];
    SceUChar8 parentalLev;
    SceChar8 reserved[3];
} SceAppUtilPspSaveDataParamSfo;
```

Members

<i>saveDataDirectory</i>	Directory name (NULL-terminated, ASCII)
<i>title</i>	Title (NULL-terminated, UTF-8)
<i>saveDataTitle</i>	Save data title (NULL-terminated, UTF-8)
<i>detail</i>	Detailed information (NULL-terminated, UTF-8)
<i>parentalLev</i>	Parental lock level
<i>reserved</i>	Reserved area (fill with 0s)

Description

This structure stores information of the save data system file for PSP™ (PARAM.SFO).
 The directory name of the save data directory for PSP™ is stored in *saveDataDirectory*.
 The title name of the game is stored in *title*.
 The title name of the save data is stored in *saveDataTitle*.
 Detailed information of the save data is specified in *detail*.
 The parental lock level of the save data is stored in *parentalLev*.

See Also

SceAppUtilPspSaveDataParams, sceAppUtilPspSaveDataLoad()

SCE CONFIDENTIAL

SceAppUtilPspSaveDataParams

Structure to store processing content for save data for PSP™

Definition

```
#include <apputil/apputil_psp.h>
typedef struct SceAppUtilPspSaveDataParams {
    SceChar8 dirName[SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE];
    SceChar8 fileName[SCE_APPUTIL_PSP_SAVEDATA_FILENAME_SIZE];
    SceChar8 reserved[3];
    SceAppUtilPspSaveDataType fileType;
    SceAppUtilPspSaveDataVersion dataVersion;
    const SceChar8 *secureFileId;
    SceAppUtilPspSaveDataParamSfo *paramSfo;
    void *dataBuf;
    SceSize dataBufSize;
    SceChar8 reserved2[4];
} SceAppUtilPspSaveDataParams;
```

Members

<i>dirName</i>	Directory name (NULL-terminated, ASCII)
<i>fileName</i>	Filename (NULL-terminated, ASCII)
<i>reserved</i>	Reserved area (fill with 0s)
<i>fileType</i>	File type
<i>dataVersion</i>	Save data format version
<i>secureFileId</i>	Protected file ID
<i>paramSfo</i>	Buffer specifying the system file (PARAM.SFO) content
<i>dataBuf</i>	Buffer specifying data
<i>dataBufSize</i>	Size of <i>dataBuf</i>
<i>reserved2</i>	Reserved area (fill with 0s)

Description

This structure specifies the processing content for save data for PSP™.

For *dirName*, specify the directory name of the processing target save data. Although the game product code and user ID were separately specified in the save data utility of PSP™, these are combined in this library to be specified as the directory name.

Example: ABCD00001 + DATA01 -> ABCD00001DATA01

The first four characters must be [A-Z] followed by five digits [0-9]. The remaining section cannot include the following characters.

- A value less than 0x20
- A value greater than 0x7E
- '\', '/', ':', '*', '?', '"', '<', '>', '|'

For *fileName*, specify the processing target save data file. [A-Z], [0-9], [.), [.), and [-] characters can be used and the following conditions must be met.

- "filename within eight characters" + "." + "extension within three characters"
- The filename does not start with "SCE_", "_", or "."

For *fileType*, specify the file type of the processing target save data as a *SceAppUtilPspSaveDataType* type.

For *dataVersion*, specify the version of the processing target save data format as a `SceAppUtilPspSaveDataVersion` type.

For *secureFileId*, specify the structure storing the protected file ID. This specification is required when `SCE_APPUTIL_PSP_SAVEDATA_TYPE_SECUREFILE` is specified to the *fileType* member variable. When `SCE_APPUTIL_PSP_SAVEDATA_TYPE_NORMALFILE` is specified to *fileType*, specify NULL for *secureFileId*. *secureFileId* differs according to each game for which the save data was created; 0 cannot be specified for all contents.

For *paramSfo*, specify the destination to store contents of PARAM.SFO when specifying or receiving the processing target save data's system file (PARAM.SFO). When NULL is specified here, processing related to PARAM.SFO will be skipped.

For *dataBuf*, specify the destination buffer for storing a file when specifying or receiving the file contents of the file specified to the *fileName* member variable. When NULL is specified here, processing related to the file will be skipped; however, note that NULL cannot be specified to both *paramSfo* and *dataBuf*.

For *dataBufSize*, specify the buffer size of *dataBuf*.

The maximum size that can be specified for *dataBufSize* is `SCE_APPUTIL_PSP_SAVEDATA_DATA_BUF_SIZE_MAX`.

See Also

`SceAppUtilPspSaveDataVersion`, `SceAppUtilPspSaveDataType`,
`SceAppUtilPspSaveDataParamSfo`, `sceAppUtilPspSaveDataLoad()`

SCE CONFIDENTIAL

SceAppUtilPspSaveDataDirName

Structure storing directory name of save data for PSP™

Definition

```
#include <apputil/apputil_psp.h>
typedef struct SceAppUtilPspSaveDataDirName {
    SceChar8 data[SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE];
} SceAppUtilPspSaveDataDirName;
```

Members

data directory name of save data for PSP™ (NULL-terminated, ASCII)

Description

This structure stores a directory name of save data for PSP™. It is used to specify save data for PSP™ in all Save Data Dialogs.

To *data*, specify the directory name of the target save data. For some features, this is used to specify the prefix of the save data's directory name.

See Also

SceAppUtilPspSaveDataParamSfoSize, SceAppUtilPspSaveDataParams,
sceAppUtilPspSaveDataGetDirNameList(), sceAppUtilPspSaveDataLoad()

SCE CONFIDENTIAL

SceAppUtilDrmAddcontId

Additional contents structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilDrmAddcontId {
    SceChar8 data[ SCE_APPUTIL_NP_DRM_ADDCONT_ID_SIZE ];
    SceChar8 padding[3];
} SceAppUtilDrmAddcontId;
```

Members

data Additional contents directory name
padding Padding area (fill with 0s)

Description

This structure represents one specific additional content.

In the member variable *data*, specify the directory name (label part of the contents ID of the additional contents package). This character string must be composed of [A-Z] and/or [0-9] characters and the NULL terminator of the SCE_APPUTIL_NP_DRM_ADDCONT_ID_SIZE size.

See Also

Additional Content Directory Name Size, `sceAppUtilDrmOpen()`, `sceAppUtilDrmClose()`

SCE CONFIDENTIAL

SceAppUtilBgdlStatus

Background download list status structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilBgdlStatus {
    SceAppUtilBgdlStatusType type;
    SceUInt32 addcontNumReady;
    SceUInt32 addcontNumNotReady;
    SceUInt32 licenseReady;
    SceChar8 reserved[28];
} SceAppUtilBgdlStatus;
```

Members

<i>type</i>	Status type
<i>addcontNumReady</i>	Number of additional contents for which download has been completed
<i>addcontNumNotReady</i>	Number of additional contents for which download has not been completed
<i>licenseReady</i>	Whether or not a downloaded upgrade license to the full version exists
<i>reserved</i>	Reserved area (fill with 0s)

Description

This is a structure that indicates information related to items that can be used by the application, among all the items that are listed in the background download list.

Specify the type of information to be obtained in *type*. *type* takes the following value.

Value	(Number)	Description
SCE_APPUTIL_BGDL_STATUS_TYPE_ADDCONT	0	Information related to the number of additional contents only
SCE_APPUTIL_BGDL_STATUS_TYPE_ADDCONT_AND_LICENSE	1	Information related to the number of additional contents and an upgrade license to the full version

The additional contents accumulated in the background download list are counted as *addcontNumNotReady* until downloading is completed. Upon download completion, they are counted as *addcontNumReady*. Additional contents become installable as soon as they are counted as *addcontNumReady*.

licenseReady is set to "!=0" when a downloaded upgrade license to the full version exists in the background download list. The state where "downloading is not complete" will never be entered in the case of an upgrade license to the full version.

For the methods to install additional contents from within an application and upgrade to the full version, refer to the "Application Development Process Overview" document.

See Also

`sceAppUtilStoreBrowse()`, `sceStoreCheckoutDialogInit()`

SCE CONFIDENTIAL

SceAppUtilStoreBrowseParam

Store browsing parameter structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilStoreBrowseParam {
    SceUInt32 type;
    const SceChar8 *id;
} SceAppUtilStoreBrowseParam;
```

Members

type Store browsing type
id ID of the product or category to open upon startup or a product code to redeem

Description

This is a system parameter type specifying the operation of the Title Store application, which is started up with `sceAppUtilStoreBrowse()`.

If browsing and purchasing the products displayed in the Title Store of PlayStation®Store, specify `SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT2` or `SCE_APPUTIL_STORE_BROWSE_TYPE_CATEGORY2` to the member variable *type*.

If `SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT2` is specified in *type*, specify the product ID in the member variable *id*; if `SCE_APPUTIL_STORE_BROWSE_TYPE_CATEGORY2` is specified in *type*, specify the category ID in *id*.

If redeeming a promotion code, specify `SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT_CODE2` in the member variable *type*.

In the member variable *id*, specify the promotion code to be redeemed as a 14-character character string divided with "-" at every four characters (e.g.: "XXXX-YYYY-ZZZZ"). If NULL or an empty character string ("") is specified in *id*, display the promotion code input UI and redeem the input promotion code.

The following macro definitions are obsolete store browsing types. They have been left for compatibility purposes, but do not use them from SDK 1.600 or later.

- `SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT`
- `SCE_APPUTIL_STORE_BROWSE_TYPE_CATEGORY`
- `SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT_CODE`

See Also

Store Browsing Type, `sceAppUtilStoreBrowse()`

SCE CONFIDENTIAL

SceAppUtilWebBrowserParam

Internet Browser parameter structure

Definition

```
#include <apputil.h>
typedef struct SceAppUtilWebBrowserParam {
    const SceChar8 *wbstr;
    SceUInt32  wbstrLength;
    SceUInt32  launchMode;
    SceUInt32  reserved0;
} SceAppUtilWebBrowserParam;
```

Members

<i>wbstr</i>	Character string passed to the command at startup
<i>wbstrLength</i>	Length of <i>wbstr</i> (bytes)
<i>launchMode</i>	Operation mode of the Internet Browser application at startup
<i>reserved0</i>	Reserved area

Description

This is a system parameter type specifying the operation of the Internet Browser application, which is started up with `sceAppUtilLaunchWebBrowser()`.

For the startup of the Internet Browser application, it is necessary to specify the startup type (`SCE_APPUTIL_WEBBROWSER_LAUNCH_APP_*`) and startup command type (`SCE_APPUTIL_WEBBROWSER_LAUNCH_CMD_*`) in *launchMode*.

The format of the character string passed to *wbstr* varies based on the startup command type. Pass an appropriate character string in accordance with each command.

See Also

Internet Browser Application Startup Type, Internet Browser Application Startup Command Type, `sceAppUtilLaunchWebBrowser()`

SCE CONFIDENTIAL

SceAppUtilWebBrowserAddCookieParam

Structure storing data parameters of a cookie to add to the Internet Browser

Definition

```
#include <apputil.h>
typedef struct SceAppUtilWebBrowserAddCookieParam {
    const SceChar8 *wbstr;
    const SceChar8 *wbstrName;
    const SceChar8 *wbstrValue;
    const SceChar8 *wbstrExpires;
    const SceChar8 *wbstrPath;
    const SceChar8 *wbstrDomain;
    SceBool isSecure;
    SceChar8 reserved;
} SceAppUtilWebBrowserAddCookieParam;
```

Members

<i>wbstr</i>	Cookie source character string (must include "scheme://domain/path")
<i>wbstrName</i>	NAME character string
<i>wbstrValue</i>	VALUE character string
<i>wbstrExpires</i>	Cookie expiration date
<i>wbstrPath</i>	Path by which cookie will be valid
<i>wbstrDomain</i>	Domain in which cookie will be valid
<i>isSecure</i>	Whether or not the cookie can only be used in HTTPS
<i>reserved</i>	Reserved area

Description

This is the system parameter type for specifying cookie data to be read by the Internet Browser using `sceAppUtilAddCookieWebBrowser()`.

For the source of *wbstr*, make sure to specify the URL of an absolute path that includes "scheme://domain/path".

For *wbstrDomain*, do not specify a value comprising just the public suffix, such as ".com" or ".co.jp".

See Also

`sceAppUtilAddCookieWebBrowser()`, `sceAppUtilResetCookieWebBrowser()`

SCE CONFIDENTIAL

SceAppUtilWebBrowserResetCookieParam

Structure storing initialization parameters of cookie data to add to the Internet Browser

Definition

```
#include <apputil.h>
typedef struct SceAppUtilWebBrowserResetCookieParam {
    SceChar8 reserved;
} SceAppUtilWebBrowserResetCookieParam;
```

Members

reserved Reserved area

Description

This is a system parameter type to specify to `sceAppUtilResetCookieWebBrowser()`.

See Also

`sceAppUtilResetCookieWebBrowser()`, `sceAppUtilAddCookieWebBrowser()`

Functions

000004892117

sceAppUtilInit

Initialize the library

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilInit(
    const SceAppUtilInitParam *initParam,
    SceAppUtilBootParam *bootParam
);
```

Calling Conditions

Not multithread safe.

Arguments

initParam Structure for setting the initializing parameter
bootParam Structure for storing boot-up parameters

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_BUSY	0x80100603	This API was called more than once
Other errors	Negative value	Fatal error

Description

This function initializes the whole application utility library according to the parameters specified in the argument *initParam*. Specify 0 to the *workBufSize* member of the first argument *initParam*. When a value other than 0 is specified, a parameter error will be returned. Application boot-up attributes and version information are stored in the second argument *bootParam*.

Make sure to call this function only once at the time of application startup.

The SCE_APPUTIL_ERROR_NOT_INITIALIZED error will occur if another function provided by the application utility is used before the initialization processing is executed.

Examples

```
SceAppUtilInitParam initParam;
SceAppUtilBootParam bootParam;

/* 0 clear */
memset( &initParam, 0, sizeof(SceAppUtilInitParam) );
memset( &bootParam, 0, sizeof(SceAppUtilBootParam) );

/* Initialize the application utility library */
ret = sceAppUtilInit( &initParam, &bootParam );
```

SCE CONFIDENTIAL

Notes

The values returned to the *attr* and *appVersion* members of the *SceAppUtilBootParam* structure are always 0.

See Also

SceAppUtilBootApplication, *SceAppUtilInitParam*, *SceAppUtilBootParam*,
sceAppUtilShutdown()

000004892117

SCE CONFIDENTIAL

sceAppUtilShutdown

Terminate the library

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilShutdown( void );
```

Calling Conditions

Not multithread safe.

Arguments

None

Return Values

Returns SCE_OK(0) for normal termination.

Returns the following error code (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized

Description

This function terminates the whole application utility library. This function is used in combination with `sceAppUtilInit()`. However, it is not necessary to call this function if `sceAppUtilInit()` has been called once at the time of application startup.

Examples

```
/* Terminate the application utility library */
ret = sceAppUtilShutdown();
```

See Also

`sceAppUtilInit()`

SCE CONFIDENTIAL

sceAppUtilReceiveAppEvent

Receive an application event

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilReceiveAppEvent(
    SceAppUtilAppEventParam *eventParam
);
```

Calling Conditions

Not multithread safe.

Arguments

eventParam Structure for event parameter obtaining

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function receives an application event.

It is possible to receive the application event parameters obtained by using the application status obtaining function `sceAppMgrGetAppState()` provided by the application manager together with this function, and then store the results into the argument *eventParam* structure.

Examples

```
SceAppMgrAppState appState;
memset(&appState, 0, sizeof(SceAppMgrAppState));

/* Obtain the application status */
ret = sceAppMgrGetAppState(&appState);

/* When obtainment of the status completes successfully and an application event
is received */
if( (ret==SCE_OK) && (appState.appEventNum > 0))
{
    SceAppUtilAppEventParam eventParam;
    memset(&eventParam, 0, sizeof(SceAppUtilAppEventParam));

    /* Receive the content of the application event */
    ret = sceAppUtilReceiveAppEvent( &eventParam );
}
```

Notes

For details on the features provided by the application manager, refer to the "Application Manager Overview" and "Application Manager Reference" documents.

SCE CONFIDENTIAL

See Also

Event Notification Type, Event Notification Parameter Size, SceAppUtilAppEventType,
SceAppUtilAppEventParam, sceAppUtilAppEventParseNpInviteMessage(),
sceAppUtilAppEventParseNpAppDataMessage(), sceAppUtilAppEventParseNearGift(),
sceAppUtilAppEventParseSessionInvitation(),
sceAppUtilAppEventParseGameCustomData()

000004892117

SCE CONFIDENTIAL

sceAppUtilAppEventParseNpInviteMessage

Parse invitation message event parameter

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseNpInviteMessage (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilNpInviteMessageParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Event parameter structure
<i>param</i>	Structure for obtaining the invitation message parameter

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses the invitation message event parameters.

When the application event type received through `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE`, the event parameters are parsed and the results are stored in the second argument *param* by calling this function. A parameter error will be returned if a different application event type is specified.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_NP_INVITE_MESSAGE) )
{
    SceAppUtilNpInviteMessageParam param;
    memset(&param, 0, sizeof(SceAppUtilNpInviteMessageParam));

    /* Parse invitation message parameters and then store the results */
    ret = sceAppUtilAppEventParseNpInviteMessage( &appEventParam, &param );
}
```

See Also

Event Notification Type, Event Notification Parameter Size, `SceAppUtilAppEventType`, `SceAppUtilAppEventParam`, `SceAppUtilNpInviteMessageParam`, `sceAppUtilReceiveAppEvent()`

©SCEI

sceAppUtilAppEventParseNpAppDataMessage

Parse event parameters of a message with game data attached

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseNpAppDataMessage (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilNpAppDataMessageParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Event parameter structure
<i>param</i>	Structure for obtaining the parameters of a message with game data attached

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses the event parameters of a message with game data attached.

When the application event type received through `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE`, the event parameters are parsed and the results are stored in the second argument *param* by calling this function. A parameter error will be returned if a different application event type is specified.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_NP_APP_DATA_MESSAGE) )
{
    SceAppUtilNpAppDataMessageParam param;
    memset(&param, 0, sizeof(SceAppUtilNpAppDataMessageParam));

    /* Parse the event parameters of a message with game data attached and then
    store the results */
    ret = sceAppUtilAppEventParseNpAppDataMessage( &appEventParam, &param );
}
```


SCE CONFIDENTIAL

See Also

Event Notification Type, Event Notification Parameter Size, SceAppUtilAppEventType,
SceAppUtilAppEventParam, SceAppUtilNpAppDataMessageParam,
sceAppUtilReceiveAppEvent ()

000004892117

SCE CONFIDENTIAL

sceAppUtilAppEventParseNearGift

Parse "near" gift event parameters

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseNearGift (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilNearGiftParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Event parameter structure
<i>param</i>	Structure for obtaining the "near" gift event parameters

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses the "near" gift event parameters.

When the application event type received through `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_NEAR_GIFT`, the event parameters are parsed and the results are stored in the second argument *param* by calling this function. A parameter error will be returned if a different application event type is specified.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type ==
    SCE_APPUTIL_APPEVENT_TYPE_NEAR_GIFT) )
{
    SceAppUtilNearGiftParam param;
    memset(&param, 0, sizeof(SceAppUtilNearGiftParam));

    /* Parse the "near" gift event parameters and then store the results */
    ret = sceAppUtilAppEventParseNearGift( &appEventParam, &param );
}
```

See Also

Event Notification Type, Event Notification Parameter Size, `SceAppUtilAppEventType`, `SceAppUtilAppEventParam`, `SceAppUtilNearGiftParam`, `sceAppUtilReceiveAppEvent()`

©SCEI

sceAppUtilAppEventParseLiveArea

Parse event parameters of application startup from LiveArea™

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseLiveArea (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilLiveAreaParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Event parameter structure
<i>param</i>	Structure for obtaining event parameters of application startup from LiveArea™

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses the event parameter of application startup from LiveArea™.

When the application event type received through `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_LIVEAREA`, the event parameters are parsed and the results are stored in the second argument *param* by calling this function. A parameter error will be returned if a different application event type is specified.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_LIVEAREA) )
{
    SceAppUtilLiveAreaParam param;
    memset(&param, 0, sizeof(SceAppUtilLiveAreaParam));

    /* Parse the parameters of application startup from LiveArea(TM) and then
    store the results */
    ret = sceAppUtilAppEventParseLiveArea( &appEventParam, &param );
}
```

See Also

Event Notification Type, Event Notification Parameter Size, `SceAppUtilAppEventType`, `SceAppUtilAppEventParam`, `SceAppUtilLiveAreaParam`, `sceAppUtilReceiveAppEvent()`

SCE CONFIDENTIAL

sceAppUtilAppEventParseScreenShotNotification

Parse event parameters of the screenshot capture notification event

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseScreenShotNotification (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilScreenShotNotification *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Structure for obtaining event parameters
<i>param</i>	Structure for obtaining event parameters of the screenshot capture notification

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses event parameters of the screenshot capture notification.

When the type of application event received by `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_SCREENSHOT_NOTIFICATION`, those event parameters can be parsed by calling this function; the results will be stored in the second argument, *param*. If the type of application event is not the above, this function returns a parameter error.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type ==
SCE_APPUTIL_APPEVENT_TYPE_SCREENSHOT_NOTIFICATION) )
{
    SceAppUtilScreenShotNotification param;
    memset(&param, 0, sizeof(SceAppUtilScreenShotNotification));

    /* Parse event parameters of the screenshot capture notification and store
    results */
    ret = sceAppUtilAppEventParseScreenShotNotification ( &appEventParam,
&param );
}
```

SCE CONFIDENTIAL

See Also

Event Notification Type, Event Notification Parameter Size, SceAppUtilAppEventType,
SceAppUtilAppEventParam, SceAppUtilScreenShotNotification,
sceAppUtilReceiveAppEvent ()

000004892117

SCE CONFIDENTIAL

sceAppUtilAppEventParseNpActivity

Parse application startup event parameters from an activity

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseNpActivity (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilNpActivityParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Event parameters structure
<i>param</i>	Structure for obtaining application startup event parameters from an activity

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses application startup event parameters from an activity.

When the type of application event received by `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_NP_ACTIVITY`, those event parameters can be parsed by calling this function; the results will be stored in the second argument, *param*. If the type of application event is not the above, this function returns a parameter error.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_NP_ACTIVITY) )
{
    SceAppUtilNpActivityParam param;
    memset(&param, 0, sizeof(SceAppUtilNpActivityParam));

    /* Parse application startup event parameters from an activity and store
    results */
    ret = sceAppUtilAppEventParseNpActivity( &appEventParam, &param );
}
```

SCE CONFIDENTIAL

See Also

Event Notification Type, Event Notification Parameter Size, SceAppUtilAppEventType,
SceAppUtilAppEventParam, SceAppUtilNpActivityParam,
sceAppUtilReceiveAppEvent ()

000004892117

SCE CONFIDENTIAL

sceAppUtilAppEventParseTeleport

Parse application startup event parameters from the Teleport library

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseTeleport (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilTeleportParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

<i>eventParam</i>	Event parameters structure
<i>param</i>	Structure for obtaining application startup event parameters from the Teleport library

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses application startup event parameters from the Teleport library.

When the type of application event received by `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_TELEPORT`, those event parameters can be parsed by calling this function; the results will be stored in the second argument, *param*. If the type of application event is not the above, this function returns a parameter error.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_TELEPORT) )
{
    SceAppUtilTeleportParam param;
    memset(&param, 0, sizeof(SceAppUtilTeleportParam));

    /* Parse application startup event parameters from the Teleport library and
    store results */
    ret = sceAppUtilAppEventParseTeleport( &appEventParam, &param );
}
```

See Also

Event Notification Type, Event Notification Parameter Size, `SceAppUtilAppEventType`, `SceAppUtilAppEventParam`, `SceAppUtilTeleportParam`, `sceAppUtilReceiveAppEvent()`

©SCEI

SCE CONFIDENTIAL

sceAppUtilAppEventParseSessionInvitation

Parse session invitation event parameters

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseSessionInvitation (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilSessionInvitationParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

eventParam Event parameters structure
param Structure for obtaining session invitation

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses session invitation event parameters.

When the type of application event received by `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_SESSION_INVITATION`, those event parameters can be parsed by calling this function; the results will be stored in the second argument, *param*. If the type of application event is not the above, this function returns a parameter error.

If an invitation ID is not specified for the parameters to parse, the *invitationId* member of the second argument *param* cannot be set.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_SESSION_INVITATION) )
{
    SceAppUtilSessionInvitationParam param;
    memset(&param, 0, sizeof(SceAppUtilSessionInvitationParam));

    /* Parse session invitation event parameters and store results */
    ret = sceAppUtilAppEventParseSessionInvitation( &appEventParam, &param );
}
```

SCE CONFIDENTIAL

See Also

Event Notification Type, Event Notification Parameter Size, SceAppUtilAppEventType,
SceAppUtilAppEventParam, SceAppUtilSessionInvitationParam,
sceAppUtilReceiveAppEvent ()

000004892117

SCE CONFIDENTIAL

sceAppUtilAppEventParseGameCustomData

Parse game custom data event parameters

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppEventParseGameCustomData (
    const SceAppUtilAppEventParam *eventParam,
    SceAppUtilGameCustomDataParam *param
);
```

Calling Conditions

Not multithread safe.

Arguments

eventParam Event parameters structure
param Structure for obtaining game custom data

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function parses game custom data event parameters.

When the type of application event received by `sceAppUtilReceiveAppEvent()` is `SCE_APPUTIL_APPEVENT_TYPE_GAME_CUSTOM_DATA`, those event parameters can be parsed by calling this function; the results will be stored in the second argument, *param*. If the type of application event is not the above, this function returns a parameter error.

Examples

```
SceAppUtilAppEventParam appEventParam;
memset(&appEventParam, 0, sizeof(SceAppUtilAppEventParam));

/* Receive the content of the application event */
ret = sceAppUtilReceiveAppEvent( &appEventParam );
if( (ret==SCE_OK) &&
    (appEventParam.type == SCE_APPUTIL_APPEVENT_TYPE_GAME_CUSTOM_DATA) )
{
    SceAppUtilGameCustomDataParam param;
    memset(&param, 0, sizeof(SceAppUtilGameCustomDataParam));

    /* Parse game custom data event parameters and store results */
    ret = sceAppUtilAppEventParseGameCustomData( &appEventParam, &param );
}
```

See Also

Event Notification Type, Event Notification Parameter Size, `SceAppUtilAppEventType`, `SceAppUtilAppEventParam`, `SceAppUtilGameCustomDataParam`, `sceAppUtilReceiveAppEvent()`

©SCEI

SCE CONFIDENTIAL

sceAppUtilSaveDataMount

Mount save data directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataMount(
    const SceAppUtilTitleId *titleId,
    const SceAppUtilPassCode *passCode
);
```

Calling Conditions

Multithread safe.

Arguments

titleId Title ID of the save data
passCode Passcode set to the save data

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Save data directory is already mounted
SCE_APPUTIL_ERROR_PASSCODE_MISMATCH	0x80100606	Passcode does not match
SCE_ERROR_ERRNO_ENOENT	0x80010002	Save data directory does not exist
Other errors	Negative value	Fatal error

Description

This function mounts the save data directory specified by *titleId*.

For *passCode*, specify the passcode set to the package of the application that created the applicable save data.

After the call of this function succeeds, the save data file will become read-accessible via the "savedata1:" drive.

The title ID of the save data specified to *titleId* must be set in advance to INSTALL_DIR_SAVEDATA_ADD1 - 7 of the application's param.sfo. When another title ID is specified, this function returns a parameter error.

Examples

```
/* Specify the save data to mount */
SceAppUtilTitleId titleId;
memset(&titleId, 0, sizeof(titleId));
strncpy((char*)titleId.data, "ABCD00001", sizeof(titleId)-1 );

/* Specify passcode of the save data (note the NULL terminator is excluded) */
SceAppUtilPassCode passCode;
memset(&passCode, 0, sizeof(passCode));
strncpy((char*)passCode.data, "AAAABBBBCCCCDDDEEEEEFFFFGGGGHHHH",
sizeof(passCode) );

/* Mount the save data directory */
ret = sceAppUtilSaveDataMount(&titleId, &passCode);
```

See Also

SceAppUtilTitleId, SceAppUtilPassCode, sceAppUtilSaveDataUmount()

SCE CONFIDENTIAL

sceAppUtilSaveDataUmount

Unmount the save data directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataUmount( void );
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns SCE_OK(0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Save data directory is being accessed
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
Other errors	Negative value	Fatal error

Description

This function unmounts the save data directory. After the call of this function succeeds, the drive called "savedata1:" will no longer exist.

Examples

```
/* Unmount the save data directory */
ret = sceAppUtilSaveDataUmount();
```

See Also

sceAppUtilSaveDataMount()

sceAppUtilSaveDataSlotCreate

Create the save data slot

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataSlotCreate (
    SceAppUtilSaveDataSlotId slotId,
    const SceAppUtilSaveDataSlotParam *param,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

slotId Save data slot ID (0 - SCE_APPUTIL_SAVEDATA_SLOT_MAX-1)
Param Save data slot parameter structure
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZE D	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
SCE_APPUTIL_ERROR_SAVEDATA_SLOT_ EXISTS	0x80100640	Slot data already exists for the specified save data slot ID
Other errors	Negative value	Fatal error

Description

This function creates a save data slot with the ID specified by *slotId*. The save data slot to be created will have the content specified by *param* set to it. Specify NULL to *mountPoint*.

An error will return if the specified ID for a save data slot already exists.

Moreover, If the specified save data slot ID is invalid (save data slot ID has been specified outside of the 0 to SCE_APPUTIL_SAVEDATA_SLOT_MAX-1 range), if *param* is NULL, or if the values of *param.title*, *param.subTitle*, *param.detail*, and/or *param.iconPath* are invalid, this function returns an error.

When NULL is specified to *mountPoint*, this function will operate in the same manner as when "savedata0:" is specified.

Examples

```
SceAppUtilSaveDataSlotId slotId;
SceAppUtilSaveDataSlotParam slotParam;
memset(&slotParam, 0, sizeof(SceAppUtilSaveDataSlotParam));

/* Set the save data slot parameter saved in the save data slot indicated with
the save data slot ID number 0 in slotParam. */
slotId = 0;
strncpy((char*)&slotParam.title, "TitleName", 9);
strncpy((char*)&slotParam.subTitle, "SubTitleName", 12);
strncpy((char*)&slotParam.detail, "DetailInfo", 10);
strncpy((char*)&slotParam.iconPath, "savedata0:sce_sys/icon0.png", 27);
slotParam.userParam = 255;
slotParam.sizeKiB = 10;

/* Create the save data slot */
ret = sceAppUtilSaveDataSlotCreate(slotId, &slotParam, NULL);
```

Notes

Specify NULL or "savedata0:" for the argument *mountPoint*. If a different mount point is specified, a parameter error is returned.

See Also

Maximum Size of a Mount Point Name, Save Data Slot Parameter Sizes,
SceAppUtilSaveDataSlotId, SceAppUtilSaveDataSlotParam, SceAppUtilMountPoint,
sceAppUtilSaveDataSlotDelete()

sceAppUtilSaveDataSlotDelete

Delete the save data slot

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataSlotDelete (
    SceAppUtilSaveDataSlotId slotId,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

slotId Save data slot ID (0 - SCE_APPUTIL_SAVEDATA_SLOT_MAX-1)
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
SCE_APPUTIL_ERROR_SAVEDATA_SLOT_NOT_FOUND	0x80100641	Slot data does not exist for the specified save data slot ID
Other errors	Negative value	Fatal error

Description

This function deletes the save data slot specified by its ID in *slotId*.

An error will return if there is no save data slot corresponding to the specified ID. Moreover, if the specified save data slot ID is invalid (save data slot ID has been specified outside of the 0 to SCE_APPUTIL_SAVEDATA_SLOT_MAX-1 range), this function returns an error.

When NULL is specified to *mountPoint*, this function will operate in the same manner as when "savedata0:" is specified.

Examples

```
SceAppUtilSaveDataSlotId slotId;

/* Set number 0 for the save data slot ID to be deleted. */
slotId = 0;

/* Delete the save data slot */
ret = sceAppUtilSaveDataSlotDelete(slotId, NULL);
```

SCE CONFIDENTIAL

Notes

Specify NULL or "savedata0:" for the argument *mountPoint*. If a different mount point is specified, this function returns a parameter error.

See Also

Maximum Size of a Mount Point Name, SceAppUtilSaveDataSlotId, SceAppUtilMountPoint, sceAppUtilSaveDataSlotCreate()

000004892117

SCE CONFIDENTIAL

sceAppUtilSaveDataSlotSearch

Search save data slots

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataSlotSearch(
    SceAppUtilWorkBuffer *workBuf,
    const SceAppUtilSaveDataSlotSearchCond *cond,
    SceAppUtilSlotSearchResult *result,
    Const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

workBuf Structure indicating the work buffer area
cond Structure representing save data slot search conditions
result Structure for storing save data slot search results
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
Other errors	Negative value	Fatal error

Description

This function searches save data slots according to the search conditions specified in *cond*. This function enables faster obtainment of a list of applicable save data slots instead of calling `sceAppUtilSaveDataSlotGetParam()` for all slots.

The number of save data slots satisfying the search conditions will be stored in the *hitNum* member of *result* argument. The pointer to the beginning of the save data slot array will be stored in the *slotList* member of *result* argument.

NULL can only be specified for *workBuf*. When NULL is specified, only the number of save data slots satisfying the search conditions specified in *cond* will be stored in the *hitNum* member of *result* argument.

For *mountPoint*, specify the target mount point. When NULL is specified, this function will operate in the same manner as when "savedata0:" is specified.

SCE CONFIDENTIAL

Examples

```

/* Set the work buffer */
SceAppUtilWorkBuffer buffer;
memset( &buffer, 0, sizeof(SceAppUtilWorkBuffer) );

/* Calculate the work buffer size */
bufSize = SCE_APPUTIL_WORKBUF_SEARCH_SLOT_DEFAULT_ELEMENT_SIZE * slotRange;

/* Allocate work buffer area */
buffer.buf = malloc(bufSize);
buffer.bufSize = bufSize;

/* Set search conditions */
SceAppUtilSaveDataSlotSearchCond cond;
memset( &cond, 0, sizeof(SceAppUtilSaveDataSlotSearchCond) );

/* Target existing save data slots, search from save data slot ID 0 for slotRange
number of slots, and sort results in ascending order using the save data slot
ID as the key */
cond.type = SCE_APPUTIL_SAVEDATA_SLOT_SEARCH_TYPE_EXIST_SLOT;
cond.from = 0;
cond.range = slotRange;
cond.key = SCE_APPUTIL_SAVEDATA_SLOT_SORT_KEY_SLOT_ID;
cond.order = SCE_APPUTIL_SAVEDATA_SLOT_SORT_TYPE_ASCENT;

/* Set search results */
SceAppUtilSlotSearchResult result;
memset( &result, 0, sizeof(SceAppUtilSlotSearchResult) );

/* Set mount point */
SceAppUtilMountPoint mountPoint;
memset( &mountPoint, 0, sizeof(SceAppUtilMountPoint) );
strncpy((char*)mountPoint.data, "savedata0:", 10);

/* Search save data slots */
ret = sceAppUtilSaveDataSlotSearch(&buffer, &cond, &result, &mountPoint);

```

See Also

SceAppUtilWorkBuffer, SceAppUtilSaveDataSlotSearchCond,
SceAppUtilSlotSearchResult, SceAppUtilMountPoint

SCE CONFIDENTIAL

sceAppUtilSaveDataSlotSetParam

Set parameters of save data slot

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataSlotSetParam(
    SceAppUtilSaveDataSlotId slotId,
    const SceAppUtilSaveDataSlotParam *param,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

slotId Save data slot ID (0 - SCE_APPUTIL_SAVEDATA_SLOT_MAX-1)
param Save data slot parameter structure
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_SAVEDATA_SLOT_NOT_FOUND	0x80100641	Slot data does not exist for the specified save data slot ID
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
Other errors	Negative value	Fatal error

Description

This function sets the save data slot parameters specified in *param* to the save data slot specified in *slotId*.

An error returns when the save data slot with the specified ID does not exist. Moreover, if the specified save data slot ID is invalid (save data slot ID has been specified outside of the 0 to SCE_APPUTIL_SAVEDATA_SLOT_MAX-1 range), or if *param* is NULL, or if the values of *param.title*, *param.subTitle*, *param.detail*, and/or *param.iconPath* are invalid, this function returns an error.

When NULL is specified to *mountPoint*, this function will operate in the same manner as when "savedata0:" is specified.

Examples

```
SceAppUtilSaveDataSlotId slotId;
SceAppUtilSaveDataSlotParam slotParam;
memset(&slotParam, 0, sizeof(SceAppUtilSaveDataSlotParam));

/* Set the save data slot parameter set in the save data slot indicated with the
save data slot ID number 0 in slotParam. */
slotId = 0;
strncpy((char*)&slotParam.title, "TitleName", 9);
strncpy((char*)&slotParam.subTitle, "SubTitleName", 12);
strncpy((char*)&slotParam.detail, "DetailInfo", 10);
strncpy((char*)&slotParam.iconPath, "savedata0:sce_sys/icon0.png", 27);
slotParam.userParam = 255;
slotParam.sizeKiB = 10;

/* Set save data slot parameters */
ret = sceAppUtilSaveDataSlotSetParam(slotId, &slotParam, NULL);
```

Notes

Specify NULL or "savedata0:" for the argument *mountPoint*. If a different mount point is specified, a parameter error is returned.

See Also

Maximum Size of a Mount Point Name, Save Data Slot Parameter Sizes,
SceAppUtilSaveDataSlotId, SceAppUtilSaveDataSlotParam, SceAppUtilMountPoint,
sceAppUtilSaveDataSlotGetParam()

SCE CONFIDENTIAL

sceAppUtilSaveDataSlotGetParam

Get parameters of a save data slot

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataSlotGetParam(
    SceAppUtilSaveDataSlotId slotId,
    SceAppUtilSaveDataSlotParam *param,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

slotId Save data slot ID (0 - SCE_APPUTIL_SAVEDATA_SLOT_MAX-1)
param Save data slot parameter structure
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
SCE_APPUTIL_ERROR_SAVEDATA_SLOT_NOT_FOUND	0x80100641	Slot data does not exist for the specified save data slot ID
Other errors	Negative value	Fatal error

Description

This function stores parameters of the save data slot specified by its ID in *slotId* to *param*.

An error returns when the save data slot with the specified ID does not exist. Moreover, if the specified save data slot ID is invalid (save data slot ID has been specified outside of the 0 to SCE_APPUTIL_SAVEDATA_SLOT_MAX-1 range), this function returns an error.

For *mountPoint*, specify the processing-target save data directory as a mount point. When NULL is specified, this function will operate in the same manner as when "savedata0:" is specified.

Examples

```
SceAppUtilSaveDataSlotId slotId;
SceAppUtilSaveDataSlotParam slotParam;
memset(&slotParam, 0, sizeof(SceAppUtilSaveDataSlotParam));

/* Set number 0 for the save data slot ID to be obtained */
slotId = 0;

/* Get save data slot parameters */
ret = sceAppUtilSaveDataSlotGetParam(slotId, &slotParam, NULL);

/* Display obtained save data slot parameters */
printf("title = %s\n", slotParam.title);
printf("subTitle = %s\n", slotParam.subTitle);
printf("detail = %s\n", slotParam.detail);
printf("iconPath = %s\n", slotParam.iconPath);
```

Notes

Even when a slot is broken (SCE_APPUTIL_SAVEDATA_SLOT_STATUS_BROKEN is set to *status* of *param*), this API call will succeed. The broken status of a slot will only be handled as one of the slot statuses.

See Also

Maximum Size of a Mount Point Name, `SceAppUtilSaveDataSlotId`,
`SceAppUtilSaveDataSlotParam`, `SceAppUtilMountPoint`,
`sceAppUtilSaveDataSlotSetParam()`

SCE CONFIDENTIAL

sceAppUtilSaveDataDataSave

Save the save data

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataDataSave (
    const SceAppUtilSaveDataDataSlot *slot,
    const SceAppUtilSaveDataDataSaveItem *data,
    SceUInt32 dataNum,
    const SceAppUtilMountPoint *mountPoint,
    SceSize *requiredSizeKiB
);
```

Calling Conditions

Multithread safe.

Arguments

<i>slot</i>	Save data slot
<i>data</i>	Save-target data (file or directory units)
<i>dataNum</i>	Number of the data to be saved (1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM)
<i>mountPoint</i>	Mount point
<i>requiredSizeKiB</i>	Required size to save the save data (unit: KiB) (When SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA or SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS error occurs)

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA	0x80100642	Not enough virtual space to write save data
SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS	0x80100643	Not enough file system space to write save data
Other errors	Negative value	Fatal error

Description

This function saves the save data specified in *data* and *dataNum* in file or directory units.

When *slot* is specified, the corresponding save data slot ID and save data slot parameter are created or set at the same time.

Meanwhile, the save data slot parameters are not set when both NULL is specified in the save data slot parameter and *slot* is specified. The processing related to the save data slot is not performed inside the function when NULL is specified in *slot*.

Upon normal termination of the function, 0 is stored in the argument *requiredSizeKiB* and returned. When the SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA or SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS error occurs, the size of the space needed (but lacking) to save the save data specified in *data* and *dataNum* is stored and returned (unit: KiB).

This function returns a parameter error when the save data slot ID specified to *slot* is invalid (the specified save data slot ID is outside the range of 0 - SCE_APPUTIL_SAVEDATA_SLOT_MAX-1), when *data* is NULL, when *dataNum* is invalid (outside the range of 1 - SCE_APPUTIL_SAVEDATA_DATA_MAXNUM), when the content specified in *data* is invalid, and when *requiredSizeKiB* is NULL.

When NULL is specified to *mountPoint*, this function will operate in the same manner as when "savedata0:" is specified.

Notes

Set values for *data* that fulfill the following conditions. If a save is performed with values that do not fulfill these conditions, severe decreases in the write speed will occur.

- The buffer specified for *data->buf* is a 64-byte aligned allocation
- If the value specified for *data->bufSize* exceeds 256*1024 bytes, complete the processing with a single write as often as possible (*dataNum* should be 1)
- A multiple of 512 bytes is specified for *data->offset*

Examples

```
SceAppUtilSaveDataSlotParam slotParam;
memset(&slotParam, 0, sizeof(SceAppUtilSaveDataSlotParam));
strncpy((char*)&slotParam.title, "TitleName", 9);
strncpy((char*)&slotParam.subTitle, "SubTitleName", 12);
strncpy((char*)&slotParam.detail, "DetailInfo", 10);
strncpy((char*)&slotParam.iconPath, "app0:icon/icon01.png", 21);
slotParam.userParam = 255;
slotParam.sizeKiB = 10;

/* Set the save data slot */
SceAppUtilSaveDataDataSlot dataSlot;
memset(&dataSlot, 0, sizeof(SceAppUtilSaveDataDataSlot));
dataSlot.id = 0;
dataSlot.slotParam = &slotParam;

/* Set the structure for saving save data data*/
SceChar8 dataPath[SCE_APPUTIL_SAVEDATA_DATA_PATH_MAXSIZE];
SceAppUtilSaveDataDataSaveItem data[1];
memset(&dataPath, 0, sizeof(dataPath));
memset(&data, 0, sizeof(SceAppUtilSaveDataDataSaveItem));
strncpy((char*)dataPath, "savedata-000.dat", 16 );
data[0].dataPath = dataPath;
```

SCE CONFIDENTIAL

```

/* Set the content to be saved into the save data specified in data[0].dataPath
*/
data[0].buf = ...
data[0].bufSize = ...

/* Set the data save mode for save data specified in data[0].dataPath. */
data[0].mode = SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_FILE;

/* Set the offset position from the start of the file when writing data into the
save data specified in data[0].dataPath */
data[0].offset = 0;

SceSize requiredSizeKiB;

/* Save the save data */
ret = sceAppUtilSaveDataDataSave(&dataSlot, data, 1,
    NULL, &requiredSizeKiB );
if( ret != SCE_OK )
{
    if( ret == SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA ||
        ret == SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS )
    {
        /* Free space is short by requiredSizeKiB to save the save data */
        printf( "requiredSizeKiB = %d KiB\n", requiredSizeKiB );

        /* Describe an appropriate processing to allocate required free space on
the application side */
        ...
    }
    else
    {
        /* When other errors occur */
        ...
    }
}
else
{
    /* Save data is saved successfully */
    ...
}

```

Notes

For the *dataPath* of the *data* argument, a path to a directory up to 4 levels deep can be specified. (for example "directory1/directory2/directroy3/data4.dat"). Also, as shown in the examples, specify the *dataPath* with omitting mount point names such as "savedata0:", etc. Also, length must be up to SCE_APPUTIL_SAVEDATA_DATA_PATH_MAXSIZE, including the NULL terminator. The length of each directory name and file name must be up to 64 characters, including the NULL terminator. Operation when saving save data differs based on the data save mode for save data specified in *mode* of the *data* argument. If SCE_APPUTIL_SAVEDATA_DATA_SAVE_MODE_DIRECTORY is specified (that is, if creating directories only), NULL must be set in the relevant *SceAppUtilSaveDataDataSaveItem* structure's member variable *buf*, and 0 must be set in the member variables *bufSize* and *offset*. 1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM can be specified for the argument *dataNum*. (It means from one to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM number of save data can be saved at a time by calling this function once.) The upper limit is 1 MiB for the individual or total size of the save data that can be saved at a time.

Specify NULL or "savedata0:" for the argument *mountPoint*. If another mount point is specified, a parameter error is returned.

Also, for debugging purposes, it is possible to simulate the occurrences of an insufficient file system free space error and insufficient save data free space error when writing save data. This can be done by editing the settings application's **★Debug Settings -> Game -> Fake Free Space (FS) and Fake Free Space (Quota)** items (only in the menu of the Development Kit (DevKit) and Testing Kit (TestKit)). Use these features when testing application operation in case file system free space is insufficient or remaining space for the save data quota is insufficient respectively. For details, refer to the "System Software Overview" document.

Use this function always to save the save data.

Also refer to "Save Data User's Guide" document on how to save save data.

See Also

Maximum Size of a Mount Point Name, Maximum Data Path Size for Save Data,
Data Save Mode for Save Data, Maximum Writable Size for Save Data,
`SceAppUtilSaveDataDataSaveMode`, `SceAppUtilSaveDataDataSaveItem`,
`SceAppUtilSaveDataDataSlot`, `SceAppUtilMountPoint`,
`sceAppUtilSaveDataDataRemove()`

SCE CONFIDENTIAL

sceAppUtilSaveDataDataRemove

Remove save data

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataDataRemove (
    const SceAppUtilSaveDataDataSlot *slot,
    const SceAppUtilSaveDataDataRemoveItem *data,
    SceUInt32 dataNum,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

<i>slot</i>	Save data slot
<i>data</i>	Data of removal-target save data (file or directory units)
<i>dataNum</i>	Number of the data to be removed (1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM)
<i>mountPoint</i>	Mount point

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZE D	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
Other errors	Negative value	Fatal error

SCE CONFIDENTIAL

Description

This function removes save data specified in *data* and *dataNum* in file or directory units.

In case *slot* is specified, processing is also carried out to the save data slots with their specified IDs.

Of the specified *data*, even if one data has

SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_KEEP_SLOT specified to the *mode* member variable, after removal of all save data succeeds, the save data slots will be updated with the contents specified by the *slotParam* member variable of *slot*.

If SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_KEEP_SLOT is not specified, as a default operation, save data slots will be deleted after all save data is removed.

If *slot* is not specified, only the save data will be removed.

This function returns a parameter error when a save data slot ID specified to *slot* is invalid (the specified save data slot ID is outside the range of 0 - SCE_APPUTIL_SAVEDATA_SLOT_MAX-1), when *data* is NULL, when *dataNum* is invalid (outside the range of 1 - SCE_APPUTIL_SAVEDATA_DATA_MAXNUM), and when the content specified in *data* is invalid. When NULL is specified to *mountPoint*, this function will operate in the same manner as when "savedata0:" is specified.

Examples

```
/* Set the save data slot */
SceAppUtilSaveDataDataSlot dataSlot;
memset(&dataSlot, 0, sizeof(SceAppUtilSaveDataDataSlot));
dataSlot.id = 0;

/* Set the structure for removing save data */
SceChar8 dataPath[SCE_APPUTIL_SAVEDATA_DATA_PATH_MAXSIZE];
SceAppUtilSaveDataDataRemoveItem data[1];
memset(&dataPath, 0, sizeof(dataPath));
memset(&data, 0, sizeof(SceAppUtilSaveDataDataRemoveItem));
strncpy((char*)dataPath, "savedata-000.dat", 16 );
data[0].dataPath = dataPath;

/* Set the data removal mode for save data set in data[0].dataPath. */
data[0].mode = SCE_APPUTIL_SAVEDATA_DATA_REMOVE_MODE_DEFAULT;

/* Remove save data */
ret = sceAppUtilSaveDataDataRemove( &dataSlot, data, 1, NULL );
```

Notes

The limitations on the removal-target path of the save data file or directory specified in *dataPath* of *data* are equivalent to those for the *dataPath* member of the *SceAppUtilSaveDataDataSaveItem* structure, the second argument of *sceAppUtilSaveDataDataSave()*.

1 to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM can be specified for the argument *dataNum*. (It means from one to SCE_APPUTIL_SAVEDATA_DATA_MAXNUM save data files or directories can be removed at a time by calling this function once.)

Specify NULL or "savedata0:" in the *mountPoint* argument. If other mount points are specified, a parameter error will return.

SCE CONFIDENTIAL

See Also

Maximum Size of a Mount Point Name, Maximum Data Path Size for Save Data,
Data Removal Mode for Save Data, SceAppUtilSaveDataDataRemoveMode,
SceAppUtilSaveDataDataRemoveItem, SceAppUtilSaveDataDataSlot,
SceAppUtilMountPoint, sceAppUtilSaveDataDataSave ()

000004892117

SCE CONFIDENTIAL

sceAppUtilSaveDataGetQuota

Get maximum save data space and currently used space

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveDataGetQuota (
    SceSize *quotaSizeKiB,
    SceSize *usedSizeKiB,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

<i>quotaSizeKiB</i>	Variable for obtaining the maximum capacity that has been set for the virtual space for the save data (NULL can be specified. Unit: KiB)
<i>usedSizeKiB</i>	Variable for obtaining the space currently used by the save data area (NULL can be specified. Unit: KiB)
<i>mountPoint</i>	Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Save data directory is not mounted
Other errors	Negative value	Fatal error

Description

This function stores in the arguments *quotaSizeKiB* and *usedSizeKiB* the maximum capacity and currently used space of the save data's virtual space. If NULL is specified in both *quotaSizeKiB* and *usedSizeKiB*, a parameter error will occur.

Specify NULL or "savedata0:" in the *mountPoint* argument. If other mount points are specified, a parameter error will return.

Examples

```
SceSize quotaSizeKiB, usedSizeKiB;

/* Obtain maximum capacity and currently used space of save data virtual space */
ret = sceAppUtilSaveDataGetQuota( &quotaSizeKiB, &usedSizeKiB, NULL );
```


Notes

If save data mount point to be obtained is mounted on host0, *usedSizeKiB* is always 0.

If save data quota information is not set to the save data mount point to be obtained, *quotaSizeKiB* is 0. Save data quota is set with the application's system file (param.sfo). For the methods to start up the application with the system file specified, refer to the "Application Development Process Overview" document.

With regard to *usedSizeKiB*, given that the system automatically creates files/directories in the area for saving save data, a given size will be consumed even if the application has not saved any save data. For details, refer to the "Save Data User's Guide" document.

Also, by editing the settings application's **★Debug Settings -> Game -> Fake Free Space (Quota)** item for debugging purposes (only in the menu of the DevKit and TestKit), it is possible to simulate a state where the current used space of save data equals the maximum capacity. For details, refer to the "System Software Overview" document.

See Also

Maximum Size of a Mount Point Name, *SceAppUtilMountPoint*

SCE CONFIDENTIAL

sceAppUtilPspSaveDataGetDirNameList

Get a list of directory names for save data for PSP™

Definition

```
#include <apputil/apputil_psp.h>
SceInt32 sceAppUtilPspSaveDataGetDirNameList(
    const SceAppUtilPspSaveDataDirName *prefix,
    SceAppUtilPspSaveDataDirName *dirNameList,
    SceUInt32 dirNameListNum,
    SceUInt32 *hitNum
);
```

Calling Conditions

Multithread safe.

Arguments

<i>prefix</i>	Prefix of directory names for list-target save data for PSP™
<i>dirNameList</i>	Destination to store directory name list of save data for PSP™
<i>dirNameListNum</i>	Number of elements in <i>dirNameList</i>
<i>hitNum</i>	Number of save data for PSP™ matching <i>prefix</i>

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
Other errors	Negative value	Fatal error

Description

This function obtains a list of directory names of save data for PSP™ saved on PlayStation®Vita.

Specify the prefix of the directory names of save data for PSP™ you wish to obtain in *prefix*. Save data for PSP™ with the matching directory name prefix will be stored to *dirNameList*.

Specify the array size of *dirNameList* to *dirNameListNum*. When the number of applicable save data exceeds the value of *dirNameListNum*, the storing of directory names will end at *dirNameListNum*, but the actual number of applicable save data will be stored in *hitNum*.

A parameter error will occur when NULL is specified to *prefix*, *dirNameList*, *hitNum*, or when a value greater than SCE_APPUTIL_PSP_SAVEDATA_DIRNAME_LIST_MAXNUM is set to *dirNameListNum*. A parameter error will also occur when one of the following character strings are specified to *prefix*.

- Empty character string
- Character string starting with an underscore (`_`) or "SCE_"
- Character string exceeding the length of
SCE_APPUTIL_PSP_SAVEDATA_PARAMSFO_DIRECTORY_SIZE
- Character string less than 0x20, greater than 0x7E, and/or including `'\', '/', '!', '*', '?', '"', '<', '>', '|'`

SCE CONFIDENTIAL

Examples

```
SceAppUtilPspSaveDataDirName
dirNameList[SCE_APPUTIL_PSP_SAVEDATA_DIRNAME_LIST_MAXNUM];
SceUInt32 hitNum = 0;

/* Target directories of save data for PSP(TM) starting with U*/
SceAppUtilPspSaveDataDirName prefix = {.data = "U"};

/* Obtain list of save data for PSP(TM) */
nRet = sceAppUtilPspSaveDataGetDirNameList(
    &prefix,
    (SceAppUtilPspSaveDataDirName*)dirNameList,
    sizeof(dirNameList)/sizeof(dirNameList[0]),
    &hitNum );
```

Notes

The obtained list of save data for PSP™ can be displayed by passing the list obtained with this feature to the Save Data Dialog library.

See Also

SCE_APPUTIL_PSP_SAVEDATA_DIRNAME_LIST_MAXNUM,
SceAppUtilPspSaveDataParamSfoSize, SceAppUtilPspSaveDataDirName

SCE CONFIDENTIAL

sceAppUtilPspSaveDataLoad

Load save data for PSP™

Definition

```
#include <apputil/apputil_psp.h>
SceInt32 sceAppUtilPspSaveDataLoad(
    SceAppUtilPspSaveDataParams *loadParam
);
```

Calling Conditions

Multithread safe.

Arguments

loadParam Parameters for save data loading

Return Values

Returns the file size of the loaded save data file for normal termination. Returns SCE_OK (0) when the call only entails the obtainment of PARAM.SFO information.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_ERROR_ERRNO_ENOENT	0x80010002	Save data directory/file does not exist
SCE_APPUTIL_ERROR_PSP_SAVEDATA_DATA_BROKEN	0x80100670	Save data is corrupted
SCE_APPUTIL_ERROR_PSP_SAVEDATA_FILETYPE_MISMATCH	0x80100671	Attempted to load protected data file as a normal file
SCE_APPUTIL_ERROR_PSP_SAVEDATA_DATABUF_SIZE	0x80100672	Buffer to load does not have enough space
Other errors	Negative value	Fatal error

Description

This function loads save data for PSP™ based on the contents specified to *loadParam*. A parameter error occurs when NULL is specified to *loadParam* or when the contents of *loadParam* are invalid. For contents that can be specified to *loadParam*, also refer to the *SceAppUtilPspSaveDataParams* section.

This function returns an error depending on the contents specified to *loadParam.dataBuf* and *loadParam.dataBufSize* if they do not meet the following conditions.

- The buffer size is a multiple of 16
- The buffer size is at least 16 bytes more than the load-target save data file
- The buffer size is SCE_APPUTIL_PSP_SAVEDATA_DATA_BUF_SIZE_MAX or less

SCE CONFIDENTIAL

Examples

```

SceAppUtilPspSaveDataParams param;
SceAppUtilPspSaveDataParamSfo paramSfo;
void *dataBuf;
SceUInt32 dataBufSize = 1024 * 1024;

/* The protected data file ID differs by title */
SceUChar8 secureFileId[SCE_APPUTIL_PSP_SAVEDATA_SECUREFILEID_SIZE] = {
    '0x0', '0x0', '0x0', '0x0', '0x0', '0x0', '0x0', '0x0',
    '0x0', '0x0', '0x0', '0x0', '0x0', '0x0', '0x0', '0x0' };

/* Initialize the structure */
memset( &param, 0x0, sizeof(param) );
memset( &paramSfo, 0x0, sizeof(paramSfo) );

/* Specify directory name of load-target save data for PSP(TM) */
strncpy( param.dirName, "ABCD00001PROFILE00", sizeof(param.dirName) );

/* Specify load-target filename */
strncpy( param.fileName, "DATA.BIN", sizeof(param.fileName) );

/* Specify whether the file specified to fileName is a protected data file or
a normal file */
param.fileType = SCE_APPUTIL_PSP_SAVEDATA_TYPE_SECUREFILE;

/* Specify the version of the save data for PSP(TM) specified to dirName */
param.dataVersion = SCE_APPUTIL_PSP_SAVEDATA_VERSION_2;

/* Specify the protected data file ID if the file specified to fileName is an
encrypted file */
/* (Specify NULL if the file specified to fileName is a normal file)*/
param.secureFileId = secureFileId;

/* Specify the structure to obtain PARAM.SFO contents if desired (can be skipped)
*/
param.paramSfo = &paramSfo;

/* Specify buffer to load file */
dataBuf = malloc(dataBufSize);
param.dataBuf = dataBuf;
param.dataBufSize = dataBufSize;
/* Execute save data for PSP(TM) load */
ret = sceAppUtilPspSaveDataLoad( &param );

```

See Also

```

SceAppUtilPspSaveDataParams, SCE_APPUTIL_PSP_SAVEDATA_DATA_BUF_SIZE_MAX,
SCE_APPUTIL_PSP_SAVEDATA_SECUREFILEID_SIZE

```

SCE CONFIDENTIAL

sceAppUtilAddcontMount

Mount additional content root directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAddcontMount(
    const SceAppUtilTitleId *titleId,
    const SceAppUtilPassCode *passCode
);
```

Calling Conditions

Multithread safe.

Arguments

titleId Additional content title ID
passCode Passcode set to the additional content

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Additional content root directory is already mounted
SCE_APPUTIL_ERROR_PASSCODE_MISMATCH	0x80100606	Passcode does not match
SCE_ERROR_ERRNO_ENOENT	0x80010002	Additional content root directory does not exist
Other errors	Negative value	Fatal error

Description

This function mounts the additional content root directory specified in *titleId*.

For *passCode*, specify the passcode set to that additional content.

After the call of this function succeeds, the additional content directory will become accessible via the drive called "addcont1:".

The title ID of the additional content specified in *titleId* must be set in advance to INSTALL_DIR_ADDCONT_ADD1 - 7 in the application's param.sfo. A parameter error will return if any other title ID is specified.

Examples

```
/* Specify additional content to mount */
SceAppUtilTitleId titleId;
memset(&titleId, 0, sizeof(titleId));
strncpy((char*)titleId.data, "ABCD00001", sizeof(titleId)-1 );

/* Specify additional content passcode (note NULL terminator is excluded) */
SceAppUtilPassCode passCode;
memset(&passCode, 0, sizeof(passCode));
strncpy((char*)passCode.data, "AAAABBBBCCCCDDDEEEEEFFFFGGGGHHHH",
sizeof(passCode) );

/* Mount additional content root directory */
ret = sceAppUtilAddcontMount(&titleId, &passCode);
```

See Also

SceAppUtilTitleId, SceAppUtilPassCode, sceAppUtilAddcontUmount()

SCE CONFIDENTIAL

sceAppUtilAddcontUmount

Unmount additional content root directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAddcontUmount( void );
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Additional content root directory is being accessed
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Additional content root directory is not mounted
Other errors	Negative value	Fatal error

Description

This function unmounts the additional content root directory. After this function succeeds, the "addcont1:" drive will no longer exist.

Examples

```
/* Unmount additional content root directory */
ret = sceAppUtilAddcontUmount();
```

See Also

sceAppUtilAddcontMount()

SCE CONFIDENTIAL

sceAppUtilDrmOpen

Open additional contents

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilDrmOpen (
    const SceAppUtilDrmAddcontId *dirName,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

dirName Additional contents directory name
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_ERROR_ERRNO_ENOENT	0x80010002	Specified additional content does not exist
SCE_ERROR_ERRNO_ENOSPC	0x8001001c	Exceeding maximum number of additional contents that can be opened at the same time
SCE_ERROR_ERRNO_EALREADY	0x80010078	Additional content is already opened
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Additional content root directory is not mounted
SCE_APPUTIL_ERROR_DRM_NO_ENTITLEMENT	0x80100660	No entitlement to the additional content
Other errors	Negative value	Fatal error

Description

This function opens the additional content directory specified in *dirName* and enables access to that directory.

This function returns a parameter error when *dirName* is NULL and when the value of *dirName.data* is invalid.

For *mountPoint*, specify the target mount point. When NULL is specified, this function will operate in the same manner as when "addcont0:" is specified.

SCE CONFIDENTIAL

Examples

```
/* Specify additional contents to be opened */
SceAppUtilDrmAddcontId dirName;
memset(&dirName, 0, sizeof(SceAppUtilDrmAddcontId));
strncpy((char*)&dirName.data, "0000111122223333", 16 );

/* Open additional contents */
ret = sceAppUtilDrmOpen( &dirName, NULL );
```

Notes

The number of additional contents that can be opened simultaneously is up to 16.

See Also

Maximum Size of a Mount Point Name, Additional Content Directory Name Size,
SceAppUtilMountPoint, SceAppUtilDrmAddcontId, sceAppUtilDrmClose()

SCE CONFIDENTIAL

sceAppUtilDrmClose

Close additional content

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilDrmClose (
    const SceAppUtilDrmAddcontId *dirName,
    const SceAppUtilMountPoint *mountPoint
);
```

Calling Conditions

Multithread safe.

Arguments

dirName Additional contents directory name
mountPoint Mount point structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Additional content directory is being accessed
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Additional content root directory does not exist
Other errors	Negative value	Fatal error

Description

This function closes the additional content directory specified in *dirName*. Since the maximum number of additional content directories that can be opened at the same time is 16, when loading more than 16 additional contents, close appropriate additional contents as necessary. After this function succeeds, the target additional content directory will become inaccessible.

This function returns a parameter error when the *dirName* argument is NULL and when the value of *dirName.data* is invalid.

For *mountPoint*, specify the target mount point. When NULL is specified, this function will operate in the same manner as when "addcont0:" is specified.

Examples

```
/* Specify additional contents to be closed */
SceAppUtilDrmAddcontId dirName;
memset(&dirName, 0, sizeof(SceAppUtilDrmAddcontId));
strncpy((char*)&dirName.data, "0000111122223333", 16 );

/* Close additional contents */
ret = sceAppUtilDrmClose( &dirName, NULL );
```

©SCEI

SCE CONFIDENTIAL

See Also

Maximum Size of a Mount Point Name, Additional Content Directory Name Size,
SceAppUtilMountPoint, SceAppUtilDrmAddcontId, sceAppUtilDrmOpen ()

000004892117

SCE CONFIDENTIAL

sceAppUtilBgdlGetStatus

Get the status of the background download list

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilBgdlGetStatus(
    SceAppUtilBgdlStatus *status
);
```

Calling Conditions

Multithread safe.

Arguments

status Background download list status structure

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
Other errors	Negative value	Fatal error

Description

This function receives the status of the system's background download list and inputs it in the *status* argument.

A parameter error results if *status* is NULL, if an invalid value is specified in *status.type*, or if *status.reserved* is not filled with 0s.

Examples

```
/* Get background download list status */

/* Initialize structure */
SceAppUtilBgdlStatus bgdlStat;
memset(&bgdlStat, 0, sizeof(SceAppUtilBgdlStatus));

/* Get background download status of additional contents and upgrade license to
the full version*/
bgdlStat.type = SCE_APPUTIL_BGDL_STATUS_TYPE_ADDCONT_AND_LICENSE;
ret = sceAppUtilBgdlGetStatus( &bgdlStat );
```

See Also

SceAppUtilBgdlStatus

SCE CONFIDENTIAL

sceAppUtilPhotoMount

Mount the photo data directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilPhotoMount( void );
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns SCE_OK(0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Photo data directory is already mounted
SCE_APPUTIL_ERROR_PHOTO_DEVICE_NOT_FOUND	0x80100680	Cannot find device for photo data
SCE_APPUTIL_ERROR_MOUNT_NUM_LIMIT_OVER	0x80100686	Mount number limit exceeded
Other errors	Negative value	Fatal error

Description

This function mounts the photo data directory.

After the call of this API is complete, read access to the photo data file is enabled through the drive "photo0:".

Up to two types of data from among photo data, music data, and video data can be mounted at the same time.

Examples

```
/* Mount the photo data directory */
ret = sceAppUtilPhotoMount();
```

See Also

sceAppUtilPhotoUmount(), Drive Names, "Photo Import Dialog Overview" document

SCE CONFIDENTIAL

sceAppUtilPhotoUmount

Unmount the photo data directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilPhotoUmount( void );
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Photo data directory is not mounted
Other errors	Negative value	Fatal error

Description

This function unmounts the photo data directory. The drive "photo0:" will no longer exist after the call of this API is complete.

Examples

```
/* Unmount the photo data directory */
ret = sceAppUtilPhotoUmount();
```

See Also

sceAppUtilPhotoMount(), "Photo Import Dialog Overview" document

SCE CONFIDENTIAL

sceAppUtilMusicMount

Mount music data directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilMusicMount( void );
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Music data directory is already mounted
SCE_APPUTIL_ERROR_MUSIC_DEVICE_NOT_FOUND	0x80100685	Cannot find device for music data
SCE_APPUTIL_ERROR_MOUNT_NUM_LIMIT_OVER	0x80100686	Mount number limit exceeded
Other errors	Negative value	Fatal error

Description

This function mounts the music data directory.

After the call of this API is complete, read access to the music data file is enabled through the drive "music0:".

Up to two types of data from among photo data, music data, and video data can be mounted at the same time.

Examples

```
/* Mount the music data directory */
ret = sceAppUtilMusicMount();
```

See Also

sceAppUtilMusicUmount(), Drive Names

SCE CONFIDENTIAL

sceAppUtilMusicUmount

Unmount music data directory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilMusicUmount( void );
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Returns SCE_OK(0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Music data directory is not mounted
Other errors	Negative value	Fatal error

Description

This function unmounts the music data directory. The drive "music0:" will no longer exist after the call of this API is complete.

Examples

```
/* Unmount the music data directory */
ret = sceAppUtilMusicUmount();
```

See Also

sceAppUtilMusicMount()

SCE CONFIDENTIAL

sceAppUtilExtVideoMount

Mount video data directory

Definition

```
#include <apputil_ext.h>
SceInt32 sceAppUtilExtVideoMount( void );
```

Calling Conditions

Multithread safe.
The application utility additional module must be loaded.
The thread stack must have 1 KiB free.

Arguments

None

Return Values

Returns SCE_OK (0) for normal termination.
Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	Video data directory is already mounted
SCE_APPUTIL_ERROR_MOUNT_NUM_LIMIT_OVER	0x80100686	Mount number limit exceeded
SCE_APPUTIL_ERROR_STACKSIZE_TOO_SHORT	0x801006a0	Insufficient stack size
Other errors	Negative value	Fatal error

Description

This function mounts the video data directory.
After the call of this API is complete, read access to the video data file is enabled through the drive "video0:". Up to two types of data from among photo data, music data, and video data can be mounted at the same time.

Notes

This function must be called in a state where the application utility additional module are loaded and 1 KiB is free in the thread stack.

Examples

```
/* Load the additional module */
Ret = sceSysmoduleLoadModule( SCE_SYSMODULE_APPUTIL_EXT );

/* Mount the video data directory */
ret = sceAppUtilExtVideoMount();
```

©SCEI

SCE CONFIDENTIAL

See Also

`sceSysmoduleLoadModule()`, `sceAppUtilExtVideoUmount()`, Drive Names, "Video Import Dialog Overview" document

000004892117

SCE CONFIDENTIAL

sceAppUtilExtVideoUmount

Unmount video data directory

Definition

```
#include <apputil_ext.h>
SceInt32 sceAppUtilExtVideoUmount( void );
```

Calling Conditions

Multithread safe.
The application utility additional module must be loaded.
The thread stack must have 1 KiB free.

Arguments

None

Return Values

Returns SCE_OK (0) for normal termination.
Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Video data directory is not mounted
SCE_APPUTIL_ERROR_STACKSIZE_TOO_SHORT	0x801006a0	Insufficient stack size
Other errors	Negative value	Fatal error

Description

This function unmounts the video data directory. The drive "video0:" will no longer exist after the call of this API is complete.

Notes

This function must be called in a state where the application utility additional module is loaded and 1 KiB is free in the thread stack.

Examples

```
/* Unmount the video data directory */
ret = sceAppUtilExtVideoUmount();
```

See Also

sceAppUtilExtVideoMount(), "Video Import Dialog Overview" document

sceAppUtilSystemParamGetInt

Get system parameters (integer values)

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSystemParamGetInt(
    SceSystemParamId paramId,
    SceInt32 *value
);
```

Calling Conditions

Not multithread safe.

Arguments

paramId System parameter ID (defined in system_param.h)
value Variable for obtaining results

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
Other errors	Negative value	Fatal error

Description

This function obtains the system parameters (integer values) set to PlayStation®Vita.

For *paramId*, it is possible to specify one of the following values defined in system_param.h. Otherwise, a parameter error will occur.

Value	(Number)	Description
SCE_SYSTEM_PARAM_ID_LANG	1	Language settings
SCE_SYSTEM_PARAM_ID_DATE_FORMAT	4	Date display format
SCE_SYSTEM_PARAM_ID_TIME_FORMAT	5	Time display format
SCE_SYSTEM_PARAM_ID_TIME_ZONE	6	Time zone offset
SCE_SYSTEM_PARAM_ID_SUMMERTIME	7	Daylight savings time

The obtained values are stored in *value*. If NULL is specified, a parameter error will occur.

For details on each setting value, refer to the "Appendix: System Parameters" chapter of this document.

*Although a system parameter is changed outside the application, it is not allowed to obtain the value repeatedly through polling to follow the changes of the parameter. You can follow the changes by obtaining the value only when the application is started up and resumes after being suspended.

SCE CONFIDENTIAL

Examples

```
/* Obtain language settings set to PlayStation(R)Vita */
SceInt32 language;

ret = sceAppUtilSystemParamGetInt( SCE_SYSTEM_PARAM_ID_LANG, &language );
```

See Also

System Parameter ID, Language Settings, SceSystemParamId,
sceAppUtilSystemParamGetString()

000004892117

SCE CONFIDENTIAL

sceAppUtilSystemParamGetString

Get system parameters (character strings)

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSystemParamGetString (
    SceSystemParamId paramId,
    SceChar8 *buf,
    SceSize bufSize
);
```

Calling Conditions

Not multithread safe.

Arguments

paramId System parameter ID (defined in system_param.h)
buf Result obtaining buffer
bufSize Result obtaining buffer size

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized

Description

This function obtains the system parameters (character strings) set to PlayStation®Vita.

For *paramId*, it is possible to specify the following value defined in system_param.h. Otherwise, a parameter error will occur.

Value	(Number)	Description
SCE_SYSTEM_PARAM_ID_USER_NAME	3	User name for display

The obtained character strings are stored in *buf*. If NULL is specified, a parameter error will occur. For *bufSize*, specify the buffer size of *buf*. If the specified size is smaller than the following value, a parameter error will occur.

Value	(Number)	Description
SCE_SYSTEM_PARAM_USER_NAME_MAXSIZE	17	Maximum user name size for display (including the NULL terminator)

*Although a system parameter is changed outside the application, it is not allowed to obtain the value repeatedly through polling to follow the changes of the parameter. You can follow the changes by obtaining the value only when the application is started up and resumes after being suspended.

Notes

User name for display that is obtained with SCE_SYSTEM_PARAM_ID_USER_NAME is determined as follows.

- Before PlayStation®Vita signs in: user name (initial value: user###)
One of the values between 000 and 999 is displayed for ###.
The user name can be changed from Friends application only before the sign-up operation.
- After PlayStation®Vita signed in: Online ID of Sony Entertainment Network account
Once PlayStation®Vita signs in, the online ID is obtained even after the sign-out operation.
Note that the user name returns if the online ID information cannot be read due to a malfunction.
- When an internal error occurs: fixed value (User999)
Fixed value returns if the user name could not also be obtained.

This API does not return an error other than a parameter error or an error generated when the library is not initialized. Thus, user name for display is always obtained. The application does not need to be aware of the content of the returned character string; display the returned character string as it is.

It is recommended to design the application to display an arbitrary character string in case this function returns an error.

Examples

```
/* Obtain the user name set to PlayStation(R)Vita */
SceChar8 buf[SCE_SYSTEM_PARAM_USER_NAME_MAXSIZE];

ret = sceAppUtilSystemParamGetString(
    SCE_SYSTEM_PARAM_ID_USER_NAME,
    &buf,
    SCE_SYSTEM_PARAM_USER_NAME_MAXSIZE );
```

See Also

System Parameter ID, User Name Size for Display, SceSystemParamId,
sceAppUtilSystemParamGetInt()

SCE CONFIDENTIAL

sceAppUtilAppParamGetInt

Get application parameters (integer values)

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAppParamGetInt (
    SceAppUtilAppParamId paramId,
    SceInt32 *value
);
```

Calling Conditions

Multithread safe.

Arguments

paramId Application parameter ID
value Result obtaining variable

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
Other errors	Negative value	Fatal error

Description

This function obtains the application parameters (integer values) set in the application.

For *paramId*, it is possible to specify the following constant definition. Specifying any other value results in a parameter error.

Value	(Number)	Description
SCE_APPUTIL_APPPARAM_ID_SKU_FLAG	0	SKU flag

The obtained parameters are stored in *value*. Specifying NULL results in a parameter error.

Examples

```
/* Obtain the parameters of the SKU flag set in the application. */
SceInt32 skuflag;

ret = sceAppUtilAppParamGetInt( SCE_APPUTIL_APPPARAM_ID_SKU_FLAG, &skuflag );
```

See Also

Application Parameter ID, SKU Flag

sceAppUtilSaveSafeMemory

Save data in a buffer into the safe memory

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilSaveSafeMemory (
    const void *buf,
    SceSize bufSize,
    SceOff offset
);
```

Calling Conditions

Multithread safe.

Arguments

<i>buf</i>	Buffer of the data to be saved into the safe memory
<i>bufSize</i>	Size of the buffer to be saved into the safe memory
<i>offset</i>	Offset to the safe memory write position

Return Values

Returns the size of data written in the safe memory for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
Other errors	Negative value	Fatal error

Description

This function saves data stored in a buffer to a safe memory area.

For *buf* and *bufSize*, specify the buffer of the data to be saved into the safe memory and the buffer size respectively. For *offset*, specify the offset to the safe memory write position.

This function returns a parameter error when NULL is specified to *buf*, and when the sum of *bufSize* plus *offset* exceeds SCE_APPUTIL_SAFEMEMORY_MEMORY_SIZE.

Since an application cannot directly refer to the memory area of the safe memory, use this function for saving data into the safe memory.

Also, by editing the settings application's ★**Debug Settings** -> **Game** -> **Init Safe Memory** item for debugging purposes (only in the menu of the DevKit and TestKit), it is possible to forcibly initialize safe memory at application startup. For details, refer to the "System Software Overview" document.

SCE CONFIDENTIAL

Examples

```
SceChar8 buf[1024];
SceOff offset = 0;

/* Set data content to save */
buf = ...

/* Save data into the safe memory */
ret = sceAppUtilSaveSafeMemory( buf, sizeof(buf), offset );
```

See Also

```
sceAppUtilLoadSafeMemory()
```

000004892117

sceAppUtilLoadSafeMemory

Read data saved in the safe memory into a buffer

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilLoadSafeMemory(
    void *buf,
    SceSize bufSize,
    SceOff offset
);
```

Calling Conditions

Multithread safe.

Arguments

<i>buf</i>	Buffer for storing the data obtained from the safe memory
<i>bufSize</i>	Size of the buffer for storing the data obtained from the safe memory
<i>offset</i>	Offset to the safe memory read position

Return Values

Returns the size of data read from the safe memory for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
Other errors	Negative value	Fatal error

Description

This function reads data saved in the safe memory.

For *buf* and *bufSize*, specify the buffer for storing the data read from the safe memory and the buffer size respectively. For *offset*, specify the offset to the safe memory read position.

This function returns a parameter error when NULL is specified to *buf*, and when the sum of *bufSize* plus *offset* exceeds SCE_APPUTIL_SAFEMEMORY_MEMORY_SIZE. Since an application cannot directly refer to the memory area of the safe memory, use this function for reading data from the safe memory.

Also, by editing the settings application's ★**Debug Settings** -> **Game** -> **Init Safe Memory** item for debugging purposes (only in the menu of the DevKit and TestKit), it is possible to forcibly initialize safe memory at application startup. For details, refer to the "System Software Overview" document.

SCE CONFIDENTIAL

Examples

```
SceChar8 buf[1024];
SceOff offset = 0;

/* Read data from the safe memory */
ret = sceAppUtilLoadSafeMemory( buf, sizeof(buf), offset );

/* Use the content read from the safe memory */
buf = ...
```

See Also

```
sceAppUtilSaveSafeMemory()
```

000004892117

SCE CONFIDENTIAL

sceAppUtilStoreBrowse

Browse Title Store of PlayStation®Store and purchase products

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilStoreBrowse (
    SceAppUtilStoreBrowseParam *param
);
```

Calling Conditions

Multithread safe.

Arguments

param Store browsing parameter

Return Values

Returns SCE_OK (0) for normal termination.

Returns one of the following error codes (negative value) for errors.

Value	(Number)	Description
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter error (details below)
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_BUSY	0x80100603	The contents are currently mounted. (details below)
Other errors	Negative value	Fatal error

Description

This function starts up the Title Store application that is used to browse and purchase the products displayed in the Title Store of PlayStation®Store or to redeem a promotion code

To the argument *param*, set a parameter to specify the operation of the Title Store application.

A parameter error results if *param* is NULL, if the value of *param.type* is invalid, or if the value of *param.id* is invalid.

If a directory or file under "addcont0:" is accessed while this API is being called, the SCE_APPUTIL_ERROR_BUSY error results. Call this API after terminating all accesses to "addcont0:" beforehand. If the additional contents directory is opened using `sceAppUtilDrmOpen()` but no file is accessed, calling this API is successful, but note that the additional contents opened using `sceAppUtilDrmOpen()` are automatically closed.

Examples

```
SceAppUtilStoreBrowseParam param;

/* Browse the products */
param.type = SCE_APPUTIL_STORE_BROWSE_TYPE_PRODUCT2;
param.id = "IV0002-NPXS00004_00-0000111122223333";
ret = sceAppUtilStoreBrowse( param );
```

SCE CONFIDENTIAL

See Also

Store Browsing Type, SceAppUtilStoreBrowseParam, sceAppUtilDrmOpen()

000004892117

sceAppUtilLaunchWebBrowser

Startup Internet Browser application with specified feature

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilLaunchWebBrowser(
    SceAppUtilWebBrowserParam *param
);
```

Calling Conditions

Multithread safe.

Arguments

param Internet Browser parameter

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function starts up the Internet Browser application.

To the argument *param*, set a parameter to specify the operation of the Internet Browser application.

Examples

```
SceAppUtilWebBrowserParam param;

/* Open URL */
#define SAMPLE_URL "http://www.scedev.net"
param.wbstr = SAMPLE_URL;
param.wbstrLength = sizeof(SAMPLE_URL) - 1;
param.launchMode = SCE_APPUTIL_WEBBROWSER_LAUNCH_APP_NORMAL |
    SCE_APPUTIL_WEBBROWSER_LAUNCH_CMD_OPENURL;
ret = sceAppUtilLaunchWebBrowser( param );
```

See Also

Internet Browser Application Startup Type, Internet Browser Application Startup Command Type, SceAppUtilWebBrowserParam

SCE CONFIDENTIAL

sceAppUtilAddCookieWebBrowser

Write out cookie data to be read (added) to the Internet Browser

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilAddCookieWebBrowser(
    SceAppUtilWebBrowserAddCookieParam *param
);
```

Calling Conditions

Multithread safe.

Arguments

param Cookie parameters to add to the Internet Browser

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function writes out cookies to be read (added) by the Internet Browser.

Specify cookie parameters to be read by the Internet Browser to *param*.

When the Internet Browser is subsequently started up, the written out data will be read.

Examples

```
#define SAMPLE_URL "http://www.scedev.net"
#define SAMPLE_COOKIE_NAME "key"
#define SAMPLE_COOKIE_VALUE "value"

SceAppUtilWebBrowserAddCookieParam param;

/* 0-clear */
memset( &param, 0, sizeof(param) );

/* Write out cookies for the Internet Browser */

param.wbstr = SAMPLE_URL;
param.wbstrName = SAMPLE_COOKIE_NAME;
param.wbstrValue = SAMPLE_COOKIE_VALUE;

SceInt32 ret = sceAppUtilAddCookieWebBrowser( param );
```

See Also

SceAppUtilWebBrowserAddCookieParam, sceAppUtilResetCookieWebBrowser(),
sceAppUtilLaunchWebBrowser()

SCE CONFIDENTIAL

sceAppUtilResetCookieWebBrowser

Initialize cookie data

Definition

```
#include <apputil.h>
SceInt32 sceAppUtilResetCookieWebBrowser(
    SceAppUtilWebBrowserResetCookieParam *param
);
```

Calling Conditions

Multithread safe.

Arguments

param Parameters to initialize cookie data to add to the Internet Browser

Return Values

Returns SCE_OK (0) for normal termination.

Returns an error code (negative value) described in the chapter Error Codes for errors.

Description

This function deletes the cookie data written out by `sceAppUtilAddCookieWebBrowser()`. The *param* argument is reserved for future extension.

Examples

```
SceAppUtilWebBrowserResetCookieParam param;

/* 0-clear */
memset( &param, 0, sizeof(param) );

/* Delete cookie data to be added */
SceInt32 ret = sceAppUtilResetCookieWebBrowser( param );
```

See Also

`SceAppUtilWebBrowserResetCookieParam`, `sceAppUtilAddCookieWebBrowser()`, `sceAppUtilLaunchWebBrowser()`

Error Codes

000004892117

Common Error Codes

List of common error codes

Definition

Value	(Number)	Description
SCE_OK	0	Normal termination
SCE_APPUTIL_ERROR_PARAMETER	0x80100600	Parameter is invalid
SCE_APPUTIL_ERROR_NOT_INITIALIZED	0x80100601	Library is not initialized
SCE_APPUTIL_ERROR_NO_MEMORY	0x80100602	Failed to allocate memory
SCE_APPUTIL_ERROR_BUSY	0x80100603	Contents are currently mounted
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Contents are currently not mounted
SCE_APPUTIL_ERROR_NO_PERMISSION	0x80100605	Permission for calling the function is not given (No APIs of the current SDK version return this error)
SCE_APPUTIL_ERROR_PASSCODE_MISMATCH	0x80100606	Passcode does not match

Description

Every function provided by the application utility returns SCE_OK (0) for normal termination or the above-described return codes (negative value) for abnormal termination.

SCE CONFIDENTIAL

Error Codes for Event Notification

List of error codes for event notification

Definition

Value	(Number)	Description
SCE_OK	0	Normal termination
SCE_APPUTIL_ERROR_APPEVENT_PARSE_INVALID_DATA	0x80100620	Result of event parse is invalid

Description

Every function provided by the application utility returns SCE_OK (0) for normal termination or the above-described return codes (negative value) for abnormal termination.

SCE CONFIDENTIAL

Error Codes for Save Data

List of error codes for save data

Definition

Value	(Number)	Description
SCE_OK	0	Normal termination
SCE_APPUTIL_ERROR_SAVEDATA_SLOT_EXISTS	0x80100640	Slot data exists in specified save data slot ID
SCE_APPUTIL_ERROR_SAVEDATA_SLOT_NOT_FOUND	0x80100641	Slot data does not exist in specified save data slot ID
SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_QUOTA	0x80100642	Remaining space for the save data quota is not enough to write the save data in the save data area
SCE_APPUTIL_ERROR_SAVEDATA_NO_SPACE_FS	0x80100643	Free space is not enough to write the save data in the system

Description

Every function provided by the application utility returns SCE_OK (0) for normal termination or the above-described return codes (negative value) for abnormal termination.

SCE CONFIDENTIAL

Error Codes for Additional Content

List of error codes for additional content

Definition

Value	(Number)	Description
SCE_OK	0	Normal termination
SCE_APPUTIL_ERROR_NOT_MOUNTED	0x80100604	Any additional content is not installed
SCE_ERROR_ERRNO_ENOENT	0x80010002	Specified additional content does not exist
SCE_ERROR_ERRNO_ENOSPC	0x8001001c	Exceeding maximum number of additional contents that can be opened at the same time
SCE_ERROR_ERRNO_EALREADY	0x80010078	Additional content is already opened
SCE_APPUTIL_ERROR_DRM_NO_ENTITLEMENT	0x80100660	No entitlement to the additional content

Description

Every function provided by the application utility returns SCE_OK (0) for normal termination or the above-described return codes (negative value) for abnormal termination.

SCE CONFIDENTIAL

Error Codes for Content Data Mounting

List of error codes for content data mounting

Definition

Value	(Number)	Description
SCE_OK	0	Normal termination
SCE_APPUTIL_ERROR_PHOTO_DEVICE_NOT_FOUND	0x80100680	Device for photo data does not exist
SCE_APPUTIL_ERROR_MUSIC_DEVICE_NOT_FOUND	0x80100685	Device for music data does not exist

Description

Every function provided by the application utility returns SCE_OK (0) for normal termination or the above-described return codes (negative value) for abnormal termination.

Parameter Error Codes

Codes output when an error occurs for a parameter which is set for application utility

Definition

No.	Description
1	NULL was specified for <code>SceAppUtilInitParam</code>
2	<code>SceAppUtilInitParam.reserved</code> is not filled with 0's
3	<code>SceAppUtilInitParam.workBufSize</code> is not 0
4	NULL was specified for <code>SceAppUtilBootParam</code>
5	<code>SceAppUtilBootParam.reserved</code> is not filled with 0s
20	NULL was specified for <code>SceAppUtilPassCode</code>
21	The format of <code>SceAppUtilPassCode.data</code> is invalid
26	NULL was specified for <code>SceAppUtilTitleId</code>
28	NULL was not specified for <code>SceAppUtilTitleId</code>
29	<code>SceAppUtilTitleId.padding</code> is not filled with 0s
30	The character format of <code>SceAppUtilTitleId.data</code> is invalid
31	The <code>SceAppUtilTitleId.data</code> string length is invalid
32	<code>SceAppUtilSaveDataSlotId</code> is invalid
33	NULL was specified for <code>SceAppUtilMountPoint</code>
34	NULL was not specified for <code>SceAppUtilMountPoint</code>
35	The character format of <code>SceAppUtilMountPoint.data</code> is invalid
36	The <code>SceAppUtilMountPoint.data</code> string length is invalid
37	NULL was specified for <code>SceAppUtilSaveDataSlotParam</code>
38	NULL was not specified for <code>SceAppUtilSaveDataSlotParam</code>
39	The character format of <code>SceAppUtilSaveDataSlotParam.title</code> is invalid
40	The <code>SceAppUtilSaveDataSlotParam.title</code> string length is invalid
41	The character format of <code>SceAppUtilSaveDataSlotParam.subTitle</code> is invalid
42	The <code>SceAppUtilSaveDataSlotParam.subTitle</code> string length is invalid
43	The character format of <code>SceAppUtilSaveDataSlotParam.detail</code> is invalid
44	<code>SceAppUtilSaveDataSlotParam.detail</code> string length is invalid
45	The character format of <code>SceAppUtilSaveDataSlotParam.iconPath</code> is invalid
46	The <code>SceAppUtilSaveDataSlotParam.iconPath</code> string length is invalid
47	<code>SceAppUtilSaveDataSlotParam.sizeKiB</code> is invalid
48	<code>SceAppUtilSaveDataSlotParam.modifiedTime</code> is invalid
49	<code>SceAppUtilSaveDataSlotParam.reserved</code> is not filled with 0's
50	<code>SceAppUtilSaveDataDataSlot.slotParam</code> is invalid Called a function for saving the save data for a save data slot ID in which the slot data does not exist without setting the save data slot parameter
51	NULL was specified for <code>SceAppUtilSaveDataDataSaveItem</code>
52	<code>dataNum</code> specified for <code>sceAppUtilSaveDataDataSave()</code> is invalid
53	NULL was specified for <code>requiredSizeKiB</code> specified for <code>sceAppUtilSaveDataDataSave()</code>
54	NULL was specified for <code>SceAppUtilSaveDataDataSaveItem.dataPath</code>
55	The character format of <code>SceAppUtilSaveDataDataSaveItem.dataPath</code> is invalid
56	The <code>SceAppUtilSaveDataDataSaveItem.dataPath</code> string length is invalid
57	NULL was specified for <code>SceAppUtilSaveDataDataSaveItem.buf</code>
58	NULL was not specified for <code>SceAppUtilSaveDataDataSaveItem.buf</code>
59	<code>SceAppUtilSaveDataDataSaveItem.bufSize</code> is not 0
60	<code>SceAppUtilSaveDataDataSaveItem.offset</code> is not 0
61	<code>SceAppUtilSaveDataDataSaveItem.bufSize</code> is too large
62	<code>SceAppUtilSaveDataDataSaveItem.padding</code> is not filled with 0s
63	<code>SceAppUtilSaveDataDataSaveItem.mode</code> is invalid

SCE CONFIDENTIAL

No.	Description
64	SceAppUtilSaveDataDataSaveItem. <i>reserved</i> is not filled with 0s
65	NULL was specified for SceAppUtilSaveDataDataRemoveItem
66	<i>dataNum</i> specified for <i>sceAppUtilSaveDataDataRemove()</i> is invalid
67	NULL was specified for SceAppUtilSaveDataDataRemoveItem. <i>dataPath</i>
68	The character format of SceAppUtilSaveDataDataRemoveItem. <i>dataPath</i> is invalid
69	The SceAppUtilSaveDataDataRemoveItem. <i>dataPath</i> string length is invalid
70	SceAppUtilSaveDataDataRemoveItem. <i>mode</i> is invalid
71	SceAppUtilSaveDataDataRemoveItem. <i>reserved</i> is not filled with 0s
72	NULL was specified both for <i>quotaSizeKiB</i> and <i>usedSizeKiB</i> specified for <i>sceAppUtilSaveDataGetQuota()</i>
80	NULL was specified for SceAppUtilDrmAddcontId
81	The character format of SceAppUtilDrmAddcontId. <i>data</i> is invalid
82	The SceAppUtilDrmAddcontId string length is invalid
100	<i>paramId</i> specified for <i>sceAppUtilSystemParamGet*()</i> is invalid
101	NULL was specified for <i>value/buf</i> specified for <i>sceAppUtilSystemParamGet*()</i>
102	<i>bufSize</i> specified for <i>sceAppUtilSystemParamGet*()</i> is insufficient
110	NULL was specified for <i>buf</i> specified for <i>sceAppUtilSaveSafeMemory()/sceAppUtilLoadSafeMemory()</i>
111	<i>bufSize</i> specified for <i>sceAppUtilSaveSafeMemory()/sceAppUtilLoadSafeMemory()</i> is too large
120	<i>paramId</i> specified for <i>sceAppUtilAppParamGet*()</i> is invalid
121	NULL was specified for <i>value/buf</i> specified for <i>sceAppUtilAppParamGet*()</i>
130	NULL was specified for SceAppUtilAppEventParam
131	<i>eventParam.type</i> specified for <i>sceAppUtilAppEventParse*()</i> is invalid
132	NULL was specified for SceAppUtilNpInviteMessageParam
133	NULL was specified for SceAppUtilNpAppDataMessageParam
135	NULL was specified for SceAppUtilNearGiftParam
136	NULL was specified for SceAppUtilLiveAreaParam
140	NULL was specified for SceAppUtilStoreBrowseParam
141	SceAppUtilStoreBrowseParam. <i>type</i> is invalid
142	SceAppUtilStoreBrowseParam. <i>id</i> is invalid
150	NULL was specified for SceAppUtilWebBrowserParam
151	SceAppUtilWebBrowserParam. <i>wbstr</i> is invalid
152	SceAppUtilWebBrowserParam. <i>wbstrLength</i> is invalid
153	SceAppUtilWebBrowserParam. <i>launchMode</i> is invalid
154	URI generated by SceAppUtilWebBrowserParam is invalid
190	SceAppUtilSaveDataSlotSearchCond is NULL
191	SceAppUtilSaveDataSlotSearchCond. <i>type</i> is invalid
192	SceAppUtilSaveDataSlotSearchCond. <i>from</i> is invalid
193	SceAppUtilSaveDataSlotSearchCond. <i>range</i> is invalid
194	SceAppUtilSaveDataSlotSearchCond. <i>key</i> is invalid
195	SceAppUtilSaveDataSlotSearchCond. <i>order</i> is invalid
196	SceAppUtilSaveDataSlotSearchCond. <i>reserved</i> is not filled with 0s
197	SceAppUtilWorkBuffer. <i>buf</i> is NULL
198	SceAppUtilWorkBuffer. <i>bufSize</i> is invalid
199	SceAppUtilWorkBuffer. <i>reserved</i> is not filled with 0s
200	SceAppUtilSlotSearchResult is NULL
201	SceAppUtilSlotSearchResult. <i>reserved</i> is not filled with 0s
210	<i>status</i> specified for <i>sceAppUtilBgdlGetStatus()</i> is NULL
211	SceAppUtilBgdlStatus. <i>reserved</i> is not filled with NULL
212	SceAppUtilBgdlStatus. <i>type</i> is invalid
230	SceAppUtilPspSaveDataParams is NULL

©SCEI

No.	Description
231	SceAppUtilPspSaveDataParams.reserved is invalid
232	SceAppUtilPspSaveDataParams.reserved2 is invalid
233	SceAppUtilPspSaveDataParams.dirName is invalid
235	SceAppUtilPspSaveDataParams.fileName is invalid
236	SceAppUtilPspSaveDataParams.fileType is invalid
237	SceAppUtilPspSaveDataParams.dataVersion is invalid
238	SceAppUtilPspSaveDataParams.secureFileId is NULL
239	SceAppUtilPspSaveDataParams.secureFileId is not NULL
240	SceAppUtilPspSaveDataParams.secureFileId is invalid
241	SceAppUtilPspSaveDataParams.dataBuf is NULL
242	SceAppUtilPspSaveDataParams.dataBufSize is invalid
243	prefix specified for sceAppUtilPspSaveDataGetDirNameList() is NULL
244	prefix specified for sceAppUtilPspSaveDataGetDirNameList() is invalid
245	dirNameList specified for sceAppUtilPspSaveDataGetDirNameList() is NULL
246	dirNameListNum specified for sceAppUtilPspSaveDataGetDirNameList() is invalid
247	hitNum specified for sceAppUtilPspSaveDataGetDirNameList() is NULL
250	NULL was specified for SceAppUtilWebBrowserAddCookieParam
251	When integrating the character string of SceAppUtilWebBrowserAddCookieParam, the maximum number of characters for a cookie was exceeded
252	The SceAppUtilWebBrowserAddCookieParam.wbstr character format is invalid
253	The SceAppUtilWebBrowserAddCookieParam.wbstr string length is invalid
254	The SceAppUtilWebBrowserAddCookieParam.wbstrName character format is invalid
255	The SceAppUtilWebBrowserAddCookieParam.wbstrName string length is invalid
256	The SceAppUtilWebBrowserAddCookieParam.wbstrValue character format is invalid
257	The SceAppUtilWebBrowserAddCookieParam.wbstrValue string length is invalid
258	The SceAppUtilWebBrowserAddCookieParam.wbstrExpires character format is invalid
259	The SceAppUtilWebBrowserAddCookieParam.wbstrExpires string length is invalid
260	The SceAppUtilWebBrowserAddCookieParam.wbstrPath character format is invalid
261	The SceAppUtilWebBrowserAddCookieParam.wbstrPath string length is invalid
262	The SceAppUtilWebBrowserAddCookieParam.wbstrDomain character format is invalid
263	The SceAppUtilWebBrowserAddCookieParam.wbstrDomain string length is invalid
270	NULL was specified for SceAppUtilScreenShotNotification
271	NULL was specified for SceAppUtilSessionInvitationParam
272	NULL was specified for SceAppUtilGameCustomDataParam
400	NULL was specified for SceAppUtilNpActivityParam
401	NULL was specified for SceAppUtilTeleportParam

Description

If the parameter specified for the functions provided by the application utility is incorrect (i.e. the return code is SCE_APPUTIL_ERROR_PARAMETER), the above-described codes which give the detailed explanation of the error will be outputted on the debugger console.

Example:

```
***** SceAppUtil Parameter Error : XX *****
```

(XX is one of the numbers described above)

```
***** SceAppUtil Parameter Error : XX [ dataIndex : YY ]*****
```

(YY is the index of the structure for saving/removing the save data specified in the save data saving/removing function)

Appendix: System Parameters

SCE CONFIDENTIAL

System Parameter ID

System Parameter ID Constant

Definition

Value	(Number)	Description
SCE_SYSTEM_PARAM_ID_LANG	1	Language settings (integer value)
SCE_SYSTEM_PARAM_ID_USER_NAME	3	User name for display (character strings, UTF-8)
SCE_SYSTEM_PARAM_ID_DATE_FORMAT	4	Date display format (integer value)
SCE_SYSTEM_PARAM_ID_TIME_FORMAT	5	Time display format (integer value)
SCE_SYSTEM_PARAM_ID_TIME_ZONE	6	Time zone offset (integer value)
SCE_SYSTEM_PARAM_ID_SUMMERTIME	7	Daylight savings time (integer value, 0: disabled, 1: enabled)

Description

These constant definitions are used when obtaining system parameters currently set to PlayStation®Vita. Specify them in the first argument of `sceAppUtilSystemParamGetInt()` or `sceAppUtilSystemParamGetString()` to obtain the values of the relevant system parameters.

Notes

These constant definitions are included in `system_param.h`.

The value obtained with `SCE_SYSTEM_PARAM_ID_TIME_ZONE` indicates the difference from the Greenwich mean time and is represented in minutes.

- GMT+09:00 -> +540
- GMT-04:00 -> -240

See Also

Language Settings, User Name Size for Display, `SceSystemParamId`, `sceAppUtilSystemParamGetInt()`, `sceAppUtilSystemParamGetString()`

SCE CONFIDENTIAL

Language Settings

Constants representing language codes

Definition

Value	(Number)	Description
SCE_SYSTEM_PARAM_LANG_JAPANESE	0	Japanese
SCE_SYSTEM_PARAM_LANG_ENGLISH_US	1	English (United States)
SCE_SYSTEM_PARAM_LANG_FRENCH	2	French
SCE_SYSTEM_PARAM_LANG_SPANISH	3	Spanish
SCE_SYSTEM_PARAM_LANG_GERMAN	4	German
SCE_SYSTEM_PARAM_LANG_ITALIAN	5	Italian
SCE_SYSTEM_PARAM_LANG_DUTCH	6	Dutch
SCE_SYSTEM_PARAM_LANG_PORTUGUESE_PT	7	Portuguese (Portugal)
SCE_SYSTEM_PARAM_LANG_RUSSIAN	8	Russian
SCE_SYSTEM_PARAM_LANG_KOREAN	9	Korean
SCE_SYSTEM_PARAM_LANG_CHINESE_T	10	Chinese (traditional)
SCE_SYSTEM_PARAM_LANG_CHINESE_S	11	Chinese (simplified)
SCE_SYSTEM_PARAM_LANG_FINNISH	12	Finnish
SCE_SYSTEM_PARAM_LANG_SWEDISH	13	Swedish
SCE_SYSTEM_PARAM_LANG_DANISH	14	Danish
SCE_SYSTEM_PARAM_LANG_NORWEGIAN	15	Norwegian
SCE_SYSTEM_PARAM_LANG_POLISH	16	Polish
SCE_SYSTEM_PARAM_LANG_PORTUGUESE_BR	17	Brazilian Portuguese (Brazil)
SCE_SYSTEM_PARAM_LANG_ENGLISH_GB	18	English (United Kingdom)
SCE_SYSTEM_PARAM_LANG_TURKISH	19	Turkish

Description

These constant definitions are used when obtaining the languages currently set to PlayStation®Vita. By specifying SCE_SYSTEM_PARAM_ID_LANG in the first argument of `sceAppUtilSystemParamGetInt()`, one of the above values is stored in the second argument and returned.

Notes

These constant definitions are included in `system_param.h`.

See Also

System Parameter ID, `SceSystemParamId`, `SceSystemParamLang`, `sceAppUtilSystemParamGetInt()`

SCE CONFIDENTIAL

User Name Size for Display

Maximum size when showing user name for display

Definition

Value	(Number)	Description
SCE_SYSTEM_PARAM_USER_NAME_MAXSIZE	17	Maximum user name size for display

Description

This is a constant definition of the maximum size of the buffer for obtaining results used when obtaining the user name (SCE_SYSTEM_PARAM_USER_NAME_MAXSIZE characters including the NULL terminator) currently set to PlayStation®Vita. It is possible to obtain the user name by specifying SCE_SYSTEM_PARAM_ID_USER_NAME in the first argument of `sceAppUtilSystemParamGetString()`, and the buffer for obtaining result as well as the above size in the second and third arguments.

Notes

This constant definition is included in `system_param.h`.

The obtained user name will be the online ID when signed up to Sony Entertainment Network, and the user name set on PlayStation®Vita when not signed up.

See Also

System Parameter ID, `SceSystemParamId`, `sceAppUtilSystemParamGetString()`

SCE CONFIDENTIAL

SceSystemParamId

System parameter ID type

Definition

```
#include <system_param.h>
typedef SceUInt32 SceSystemParamId;
```

Description

This is a system parameter ID type used when obtaining system parameters.

It is used as the type of *paramId*, the first argument of the functions for obtaining system parameters `sceAppUtilSystemParamGetInt()` and `sceAppUtilSystemParamGetString()`.

In the *paramId* argument, specify the system parameter ID indicated with `SCE_SYSTEM_PARAM_ID_XXX`.

See Also

System Parameter ID, `sceAppUtilSystemParamGetInt()`,
`sceAppUtilSystemParamGetString()`

SCE CONFIDENTIAL

SceSystemParamLang

System parameter (language settings) type

Definition

```
#include <system_param.h>
typedef SceInt32 SceSystemParamLang;
```

Description

This type represents the value of the system parameter (language settings).

See Also

System Parameter ID, Language Settings, sceAppUtilSystemParamGetInt()