NpWebApi Library Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Initialization/Termination and Check Callback

sceNpWebApiInitialize

Initialize NpWebApi library

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiInitialize(
        size_t poolSize
);
```

Arguments

poolSize	Memory pool size for the NpWebApi library

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function initializes the NpWebApi library.

The memory pool size to specify in *poolSize* must be a multiple of 4 KiB; otherwise, SCE_KERNEL_ERROR_ILLEGAL_MEMBLOCK_SIZE will return.

Examples

```
#define NP_WEBAPI_POOL_SIZE ( 64 * 1024 )

int32_t ret = 0;

ret = sceNpWebApiInitialize(NP_WEBAPI_POOL_SIZE);
if(ret < 0){
        /* Error handling */
}
```

Notes

This function is not multithread safe. Although this function may reach normal termination when it is called by multiple threads at the same time, subsequent library operation cannot be guaranteed. Make sure to program the application so that this function is not called at the same time by multiple threads.

See Also

sceNpWebApiTerminate()

sceNpWebApiTerminate

Terminate NpWebApi library

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiTerminate(
        void
);
```

Arguments

None

Return Values

Returns `SCE_OK` (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function terminates the NpWebApi library.

When called during execution of sending/receiving, forced termination will occur, sending/receiving functions will return a "processing was aborted" error, and the `sceNpWebApiTerminate()` function itself will terminate normally

Examples

```
int32_t ret = 0;

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

Notes

This function is not multithread safe. Although this function may reach normal termination when it is called by multiple threads at the same time, subsequent library operation cannot be guaranteed. Make sure to program the application so that this function is not called at the same time by multiple threads.

See Also

`sceNpWebApiInitialize()`

sceNpWebApiCheckCallback

Check a callback

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCheckCallback(
        void
);
```

Arguments

None

Return Values

Returns `SCE_OK` (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function checks the callback of a Push/service Push event.

A check is performed to see if a state exists where a callback registered with `sceNpWebApiRegisterPushEventCallback()` or `sceNpWebApiRegisterServicePushEventCallback()` will be called, and if such a state does exist, then the callback will be called with the thread context that called `sceNpWebApiCheckCallback()`.

Applications that use Push events/service Push events should regularly call this function

Examples

```
int32_t ret = 0;

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

Notes

In the NpWebApi library for PlayStation®4, `sceNpCheckCallback()` provides the feature equivalent to this function; please take care when realizing application portability.

Do not call this function from a main thread (rendering thread). This function is dependent on the processing in the threads in system processes in accordance with the inter-process communication in system processes, therefore there is a possibility that unexpectedly long periods of time may be required depending on the processing statuses of system processes.

See Also

`sceNpWebApiRegisterPushEventCallback()`,
`sceNpWebApiRegisterServicePushEventCallback()`

Requests

SCE_NP_WEBAPI_CONTENT_TYPE_APPLICATION_JSON_UTF8

Type of data to send as a request body

Definition

Value	(string)	Description
SCE_NP_WEBAPI_CONTENT_TYPE_APPLICATION_JSON_UTF8	"application/json; charset=utf-8"	Content-Type representing JSON format data

Description

This constant represents the type of data to send as a request body when executing a PSN℠ Web API. It is used as the Content-Type value for the HTTP request header.

Document serial number: 000004892117

SCE_NP_WEBAPI_HTTP_METHOD_XXX

HTTP method upon PSN℠ Web API execution

Definition

Value	(Number)	Description
SCE_NP_WEBAPI_HTTP_METHOD_GET	0	HTTP method GET
SCE_NP_WEBAPI_HTTP_METHOD_POST	1	HTTP method POST
SCE_NP_WEBAPI_HTTP_METHOD_PUT	2	HTTP method PUT
SCE_NP_WEBAPI_HTTP_METHOD_DELETE	3	HTTP method DELETE

Description

These constants represent the HTTP method to specify when executing a PSN℠ Web API.

SceNpWebApiContentParameter

Parameters for data to send as request body

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiContentParameter {
        size_t contentLength;
        const char *pContentType;
        uint8_t reserved[16];
} SceNpWebApiContentParameter;
```

Members

contentLength	Total size of the data to send as a request body upon PSN℠ Web API execution (bytes)
pContentType	Character string to be set for Content-Type of the HTTP header (ASCIIZ string)
reserved	Reserved area

Description

This structure holds the parameters for the data to send as a request body when executing a PSN℠ Web API.

When sending in JSON format (the send data format of many PSN℠ Web APIs), `SCE_NP_WEBAPI_CONTENT_TYPE_APPLICATION_JSON_UTF8` can be specified for *pContentType*.

For details on the character strings that should be set for *pContentType*, refer to "PSN℠ Web APIs Overview" and the reference documents for each Web API.

SceNpWebApiResponseInformationOption

Option information regarding Web API server response

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiResponseInformationOption {
        int32_t httpStatus;
        char *pErrorObject;
        size_t errorObjectSize;
        size_t responseDataSize;
} SceNpWebApiResponseInformationOption;
```

Members

httpStatus	HTTP status code
pErrorObject	Pointer to buffer to store response body upon server error
errorObjectSize	Size of buffer to store response body upon server error
responseDataSize	Actual size of response body upon server error

Description

This structure is for storing the server response content if an error has occurred when the Web APIs are executed with `sceNpWebApiSendRequest2()` and `sceNpWebApiSendMultipartRequest2()`.

When a server error has occurred, `sceNpWebApiSendRequest2()`/`sceNpWebApiSendMultipartRequest2()` will return an appropriate error code based on the server response. Therefore, applications do not normally need to obtain information on this structure. Use this structure in cases such as when it is desired to perform a detailed investigation of the cause of errors during development.

sceNpWebApiAbortRequest

Abort request processing

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiAbortRequest(
        int64_t requestId
);
```

Arguments

requestId Request ID

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function aborts the request processing.

The processing of sceNpWebApiSendRequest2(), sceNpWebApiSendRequest() and sceNpWebApiReadData() will be aborted. Communication with the server will be aborted, and currently processing functions will immediately return.

Examples

```
int32_t ret = 0;
int64_t requestId;

ret = sceNpWebApiAbortRequest(requestId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiSendRequest2(), sceNpWebApiSendRequest(), sceNpWebApiReadData()

sceNpWebApiCreateRequest

Create request

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCreateRequest(
        const char *pApiGroup,
        const char *pPath,
        SceNpWebApiHttpMethod method,
        const SceNpWebApiContentParameter *pContentParameter,
        int64_t *pRequestId
);
```

Arguments

pApiGroup	API group of the PSN℠ Web API to execute (ASCIIZ string)
pPath	Path of the PSN℠ Web API to execute (ASCIIZ string)
method	HTTP method upon PSN℠ Web API execution
pContentParameter	Parameters relating to data sent as a request body, or NULL
pRequestId	Storage destination for the obtained request ID

Return Values

Stores the obtained request ID in **pRequestId* and returns SCE_OK (=0) for normal termination.
Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function creates a request for executing a PSN℠ Web API. When the request is properly created, the request ID will be stored in **pRequestId*.

The path specified for *pPath* is a character string that links the resource path of the PSN℠ Web API to execute and the query string. This character string must always be percent-encoded (URL-encoded) based on RFC 3986.

For *method*, specify one of the following values.

Value	(Number)	Description
SCE_NP_WEBAPI_HTTP_METHOD_GET	0	GET method
SCE_NP_WEBAPI_HTTP_METHOD_POST	1	POST method
SCE_NP_WEBAPI_HTTP_METHOD_PUT	2	PUT method
SCE_NP_WEBAPI_HTTP_METHOD_DELETE	3	DELETE method

For **pContentParameter*, if there is data sent as a request body upon PSN℠ Web API execution, specify a SceNpWebApiContentParameter structure that stores the parameters relating to the data. If there is no data to send, specify NULL.

Examples

```
#define API_GROUP "userProfile"
#define PATH "/v1/users/user000/profile"

int32_t ret = 0;
int64_t requestId = 0;

ret = sceNpWebApiCreateRequest(
        API_GROUP, PATH,
        SCE_NP_WEBAPI_HTTP_METHOD_GET,
        NULL, &requestId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiDeleteRequest(),SceNpWebApiContentParameter

sceNpWebApiDeleteRequest

Delete request

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiDeleteRequest(
        int64_t requestId
);
```

Arguments

requestId Request ID

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function deletes a request.

The request will be forcibly deleted when the function is called during use of a request such as during sending/receiving.

Examples

```
int32_t ret = 0;
int64_t requestId;

ret = sceNpWebApiDeleteRequest(requestId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiCreateRequest()

sceNpWebApiGetErrorCode

Generate an error code from the error response [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiGetErrorCode(
        int32_t httpStatusCode,
        const void *pErrorJson,
        size_t errorJsonSize
);
```

Arguments

httpStatusCode	HTTP status code obtained with `sceNpWebApiGetHttpStatusCode()`
pErrorJson	JSON data obtained as the error response
errorJsonSize	Size of JSON data obtained as the error response

Return Values

Always returns an error code (negative value) and reaches normal termination.

Description

Note: This API (`sceNpWebApiGetErrorCode()`) is planned to be removed in the future. By using `sceNpWebApiSendRequest2()` during Web API execution, it will be possible to obtain error codes without using `sceNpWebApiGetErrorCode()`.

This function is for obtaining an error code from the JSON data of the HTTP status code and error response when an error response is received as a result of calling a PSN℠ Web API.

When an error can be determined from the JSON data of the HTTP status code and error response, an error code starting with 0x82 will return. An error code starting with 0x82 indicates a server error; the lower 24 bits excluding 0x82 of this error code expressed as a decimal number serves as the value for determining the specific server error. Refer to the each PSN℠ Web API reference document for server error definitions. In addition, when the JSON data of the error response is invalid, an error code corresponding to the HTTP status code will return.

Examples

```
int32_t ret = 0;
// HTTP status code obtained with sceNpWebApiGetHttpStatusCode()
int32_t httpStatusCode;
// Error response JSON data obtained with sceNpWebApiReadData()
char readBuf[BUF_SIZE];
size_t readSize; // Size of the error response JSON data

ret = sceNpWebApiGetErrorCode(httpStatusCode, readBuf, readSize);

// Error handling
```

See Also

`sceNpWebApiGetHttpStatusCode()`, `sceNpWebApiReadData()`

sceNpWebApiGetHttpStatusCode

Obtain HTTP status code [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiGetHttpStatusCode(
        int64_t requestId,
        int32_t *pStatusCode
);
```

Arguments

requestId	Request ID
pStatusCode	Storage destination for the obtained HTTP status code

Return Values

Stores the obtained HTTP status code in **pStatusCode* and returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API (sceNpWebApiGetHttpStatusCode()) is planned to be removed in the future. By using sceNpWebApiSendRequest2() during Web API execution, it no longer be necessary to obtain HTTP status codes with sceNpWebApiGetHttpStatusCode().

This function obtains the HTTP status code.

After PSN℠ Web API execution (after sceNpWebApiSendRequest() has sent all of the data to send and has returned), the HTTP status code can be obtained. For the meanings of the HTTP status codes for each PSN℠ Web API, refer to "PSN℠ Web APIs Overview" and the reference documents for each PSN℠ Web API.

Examples

```
int32_t ret = 0;
int64_t requestId;
int32_t statusCode = 0;

ret = sceNpWebApiGetHttpStatusCode(requestId, &statusCode);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiSendRequest()

sceNpWebApiReadData

Receive response body

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiReadData(
        int64_t requestId,
        void *pData,
        size_t size
);
```

Arguments

requestId	Request ID
pData	Storage destination for the obtained response body
size	Size of the buffer specified with *pData* (bytes)

Return Values

Stores the obtained response body in the **pData* buffer and returns the size of the stored data for normal termination. Returns 0 if all response bodies have already been received and there is no data to be stored in the buffer.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function receives the response body of the request specified with *requestId*. The received response body will be written to the address specified with *pData*.

This function is a blocking function. The conditions for this function to return will be one of the following:

(a) Data in bytes specified with *size* was received and stored in the buffer

(b) If Content-Length is in the HTTP response header, a response body of the size specified with Content-Length was received completely and stored in the buffer

(c) `sceNetRecv()` returned 0 or a negative value after being called in the library

(d) If chunk encoded data is being received, the last chunk was received completely and stored in the buffer

(e) `sceNpWebApiAbortRequest()` was executed in another thread

In the case of (a), there is a possibility that the entire response body has not been received/stored completely. The entire response body can be obtained by repeatedly calling this function until 0 returns.

Examples

```
int64_t requestId; // Request ID of the request to send

int32_t ret = 0;
char buf[2*1024];

do {
        ret = sceNpWebApiReadData(requestId, buf, sizeof(buf));
        if(ret < 0){
              /* Error handling */
        }
        else if (ret > 0) {
              printf("sceNpWebApiReadData() read %d bytes\n", ret);
        }
} while (ret > 0);
```

Notes

This function is a blocking function. Processing may take time, therefore it should be called from a subthread.

See Also

sceNpWebApiSendRequest2(),sceNpWebApiSendRequest(),sceNpWebApiAbortRequest()

sceNpWebApiSendRequest

Send request and execute PSN℠ Web API [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiSendRequest(
        int64_t requestId,
        const void *pData,
        size_t dataSize
);
```

Arguments

requestId	Request ID
pData	Data sent as a request body (all or some), or NULL
dataSize	Size of the data pointed to by *pData* (bytes), or 0

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API (sceNpWebApiSendRequest()) is planned to be removed in the future. When sending a request and executing the Web API, use of sceNpWebApiSendRequest2() is recommended.

This function sends the request specified with *requestId* to the server and executes a PSN℠ Web API.

If there is data to send as a request body upon PSN℠ Web API execution, place the data in memory then specify *pData* and *dataSize*.

There is no need to send the entire data at once; it is also possible to partition the data and send it by calling this function multiple times. In order to properly partition and send data, specify the total size of the data for *contentLength* of **pContentParameter* when creating the request with sceNpWebApiCreateRequest().

This function is a blocking function. The conditions for this function to return are as follows:

- If there is no data to send as a request body:
  This function returns when the request has been sent to the server and the HTTP response header has been received.
- If there is data to send as a request body:
  If the entire send data size set with *contentLength* of **pContentParameter* is still being sent, this function returns when data in the size specified with this function call has been sent. If the entire send data size has been sent, this function returns when the HTTP response header has been received from the server.
- When a function that aborts request processing has been executed in another thread:
  When sceNpWebApiAbortRequest() is executed with a request being processed by sceNpWebApiSendRequest() as a target, sceNpWebApiSendRequest() immediately aborts the request processing and returns. At this time, SCE_NP_WEBAPI_ERROR_ABORTED will return.

Examples

```
int32_t ret = 0;
int64_t requestId;

ret = sceNpWebApiSendRequest(requestId, NULL, 0);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiGetHttpStatusCode(),sceNpWebApiReadData()

sceNpWebApiSendRequest2

Send request and execute Web API

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiSendRequest2(
        int64_t requestId,
        const void *pData,
        size_t dataSize,
        SceNpWebApiResponseInformationOption *pRespInfoOption
);
```

Arguments

requestId	Request ID
pData	Data to send as request body (all or part), or NULL
dataSize	Size of data pointed to by *pData* (bytes), or 0
pRespInfoOption	Structure to store server response content upon server error, or NULL

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function sends the request specified with *requestId* to the server and executes a Web API.

If there is data to send as a request body upon Web API execution, place the data in memory then specify *pData* and *dataSize*.

There is no need to send the entire data at once; it is also possible to partition the data and send it by calling this function multiple times. In order to properly partition and send data, specify the total size of the data for *contentLength* of **pContentParameter* when creating the request with sceNpWebApiCreateRequest().

When a server error has occurred, the error response data will be obtained internally by this function, then an error code (0x82XXXXXX) that indicates the server error will be generated and will return as the return value of this function.

This function is a blocking function. The conditions for this function to return are as follows:

- If there is no data to send as a request body:
  This function returns when the request has been sent to the server and the HTTP response header has been received.
- If there is data to send as a request body:
  If the entire send data size set with *contentLength* of **pContentParameter* is still being sent, this function returns when data in the size specified with this function call has been sent. If the entire send data size has been sent, this function returns when the HTTP response header has been received from the server.
- When a function that aborts request processing has been executed in another thread:
  When sceNpWebApiAbortRequest() is executed with a request being processed by sceNpWebApiSendRequest2() as a target, sceNpWebApiSendRequest2() immediately aborts the request processing and returns. At this time, SCE_NP_WEBAPI_ERROR_ABORTED will return as a return value.

Examples

```
int32_t ret = 0;
int64_t requestId;

char errorObjectBuf[ERROR_OBJECT_BUF_SIZE];
memset(errorObjectBuf, 0, sizeof(errorObjectBuf));

SceNpWebApiResponseInformationOption respInfoOption;
memset(&respInfoOption, 0, sizeof(respInfoOption));
respInfoOption.pErrorObject = errorObjectBuf;
respInfoOption.errorObjectSize = sizeof(errorObjectBuf);

ret = sceNpWebApiSendRequest2(requestId, NULL, 0, &respInfoOption);
if(ret < 0){
        /* Error handling */
}
```

Notes

This function is a blocking function. Processing may take time, therefore it should be called from a subthread.

See Also

sceNpWebApiReadData()

Multiple Parts

SceNpWebApiHttpHeader

HTTP header

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiHttpHeader {
        char *pName;
        char *pValue;
} SceNpWebApiHttpHeader;
```

Members

pName HTTP header name (ASCIIZ string)
pValue HTTP header value (ASCIIZ string)

Description

This structure represents an HTTP header. It is used for specifying a header to various parts for sending multiple parts as a request body upon PSN℠ Web API execution.

SceNpWebApiMultipartPartParameter

Part parameters

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiMultipartPartParameter {
        SceNpWebApiHttpHeader *pHeaders;
        size_t headerNum
        size_t contentLength;
} SceNpWebApiMultipartPartParameter;
```

Members

pHeaders	Array of headers to insert in part in multiple parts
headerNum	Number of elements in array pointed to by *pHeaders*
contentLength	Size of data to send in a single part in multiple parts (bytes)

Description

This structure has parameters that represent a single part in data to send in multiple parts as a request body upon PSN℠ Web API execution.

For details on header strings that should be set for **pHeaders*, refer to the various Web API reference documents.

sceNpWebApiAddMultipartPart

Add part in multiple parts

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiAddMultipartPart(
        int64_t requestId,
        const SceNpWebApiMultipartPartParameter *pParam,
        int32_t *pPartIndex
);
```

Arguments

requestId	Request ID
pParam	Multiple part parameters
pPartIndex	Buffer to store the index number of the added part

Return Values

Stores the part index number in **pPartIndex* and returns SCE_OK (=0) for normal termination.
Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function adds a part to be sent in multiple parts. When part addition is successful, the part index number will be stored in **pPartIndex*, so specify this value for *partIndex* in sceNpWebApiSendMultipartRequest2() when sending the request.

Examples

```
int32_t ret = 0;
int64_t requestId;
SceNpWebApiHttpHeader headers[2];
SceNpWebApiMultipartPartParameter partParam;
int32_t partIndex = 0;

memset(&headers, 0, sizeof(headers));
headers[0].pName = "Content-Type";
headers[0].pValue = "application/json; charset=utf-8";
headers[1].pName = "Content-Description";
headers[1].pValue = "session-request";

memset(&partParam, 0, sizeof(partParam));
partParam.pHeaders = headers;
partParam.headerNum = 2;
partParam.contentLength = strlen(JSON_DATA);

ret = sceNpWebApiAddMultipartPart(
        requestId, &partParam, &partIndex);
if(ret < 0){
        /* Error handling */
}
```

See Also

`sceNpWebApiCreateMultipartRequest(),sceNpWebApiSendMultipartRequest2()`

sceNpWebApiCreateMultipartRequest

Create request for sending in multiple parts

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCreateMultipartRequest(
        const char *pApiGroup,
        const char *pPath,
        SceNpWebApiHttpMethod method,
        int64_t *pRequestId
);
```

Arguments

pApiGroup	API group of Web API to execute (ASCIIZ string)
pPath	Path of Web API to execute (ASCIIZ string)
method	HTTP method upon Web API execution
pRequestId	Destination to store obtained request ID

Return Values

Stores the obtained request ID in **pRequestId* and returns SCE_OK (=0) for normal termination.
Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function creates a request for executing a PSN℠ Web API in accordance with the sending of multiple parts. When the request is created properly, the request ID will be stored in **pRequestId*.

The path to specify for **pPath* is a string that connects the resource path of the Web API to execute and the query string. This string must always be percent-encoded (URL encoded) based on RFC 3986.

For *method*, specify one of the following values.

Value	(Number)	Description
SCE_NP_WEBAPI_HTTP_METHOD_GET	0	GET method
SCE_NP_WEBAPI_HTTP_METHOD_POST	1	POST method
SCE_NP_WEBAPI_HTTP_METHOD_PUT	2	PUT method
SCE_NP_WEBAPI_HTTP_METHOD_DELETE	3	DELETE method

Examples

```
#define API_GROUP "sessionInvitation"
#define PATH "/v1/sessions"

int32_t ret = 0;
int32_t userCtxId;
int64_t requestId = 0;

ret = sceNpWebApiCreateMultipartRequest(
        userCtxId, API_GROUP, PATH,
        SCE_NP_WEBAPI_HTTP_METHOD_POST,
        &requestId);
if(ret < 0){
        /* Error handling */
}
```

See Also

`sceNpWebApiDeleteRequest()`,`sceNpWebApiAddMultipartPart()`,
`sceNpWebApiSetMultipartContentType()`,`sceNpWebApiSendMultipartRequest2()`

sceNpWebApiSendMultipartRequest

Send multiple part request and execute Web API [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiSendMultipartRequest(
        int64_t requestId,
        int32_t partIndex,
        const void *pData,
        size_t dataSize
);
```

Arguments

requestId	Request ID
partIndex	Part index number
pData	Data to send with parts indicated by *partIndex* (all or some)
dataSize	Size of data pointed to by *pData* (bytes)

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API (sceNpWebApiSendMultipartRequest()) is planned to be removed in the future. When sending a multipart request and executing the Web API, use of sceNpWebApiSendMultipartRequest2() is recommended.

This function sends the parts specified with the *partIndex* of the request specified with *requestId* to a server. By executing this function for all parts added to a request, execution of PSN℠ Web APIs will terminate.

There is no need to send all data at once, it is also possible to send data in parts by calling this function multiple times. In order to properly send data in parts, set the total size of the data to send in parts for *contentLength* in SceNpWebApiMultipartPartParameter when adding the parts using sceNpWebApiAddMultipartPart()

This function is a blocking function. The conditions for this function to return are as follows.

- In normal cases:
  During periods when the sending of the total size of all of the send data parts set for *contentLength* has not completed, this function will return when data in the size specified when this function was called has been sent. If the sending of the total size of the send data of the last part has completed, this function will return when the HTTP response header has been received from the server.
- When a function that aborts a request in another thread has been executed:
  When sceNpWebApiAbortRequest() has been executed for a request being processed by sceNpWebApiSendMultipartRequest() (this function), this function will immediately abort the request processing and return. At this time, SCE_NP_WEBAPI_ERROR_ABORTED will return as the return value.

Examples

```
int32_t ret = 0;
int64_t requestId;
int32_t partIndex;

ret = sceNpWebApiSendMultipartRequest(requestId, partIndex, POST_DATA,
POST_DATA_SIZE);
if(ret < 0){
        /* Error handling */
}
```

Notes

This function is a blocking function. Processing may take time, therefore it should be called from a subthread.

See Also

`sceNpWebApiCreateMultipartRequest()`,`sceNpWebApiAddMultipartPart()`, `sceNpWebApiGetHttpStatusCode()`,`sceNpWebApiReadData()`

sceNpWebApiSendMultipartRequest2

Send multipart request and execute Web API

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiSendMultipartRequest2(
        int64_t requestId,
        int32_t partIndex,
        const void *pData,
        size_t dataSize,
        SceNpWebApiResponseInformationOption *pRespInfoOption
);
```

Arguments

requestId	Request ID
partIndex	Part index number
pData	Data to send in part indicated by *partIndex* (all or part)
dataSize	Size of data pointed to by *pData* (bytes)
pRespInfoOption	Structure to store server response content upon server error, or NULL

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function sends the part specified by *partIndex* of the request specified by *requestId* to the server. Execute this function for all the parts added to the request to complete the execution of the Web API.

There is no need to send the entire data all at once. It is possible to call this function multiple times to send the data in increments. To correctly send data in increments, set the size of the entire data for that part to send to *contentLength* of SceNpWebApiMultipartPartParameter upon adding the part with sceNpWebApiAddMultipartPart().

When a server error has occurred, the error response data will be obtained internally by this function, then an error code (0x82XXXXXX) that indicates the server error will be generated and will return as the return value of this function.

This function is a blocking function. The conditions for this function to return are as follows:

- Normally:
  While the total data size of all parts set in *contentLength* is not sent, this function returns when the data size - specified upon this function call - has been sent. When the total data size of the last part to send has been sent, this function returns when an HTTP response header has been received from the server.
- When a function to abort request processing has been executed in another thread:
  When sceNpWebApiAbortRequest() is executed for a request that is being processed by this function as a target, this function immediately aborts the request processing and returns. In this case, this function returns SCE_NP_WEBAPI_ERROR_ABORTED.

Examples

```
int32_t ret = 0;
int64_t requestId;
int32_t partIndex;

char errorObjectBuf[ERROR_OBJECT_BUF_SIZE];
memset(errorObjectBuf, 0, sizeof(errorObjectBuf));

SceNpWebApiResponseInformationOption respInfoOption;
memset(&respInfoOption, 0, sizeof(respInfoOption));
respInfoOption.pErrorObject = errorObjectBuf;
respInfoOption.errorObjectSize = sizeof(errorObjectBuf);

ret = sceNpWebApiSendMultipartRequest2(requestId, partIndex, POST_DATA,
POST_DATA_SIZE, &respInfoOption);
if(ret < 0){
        /* Error handling */
}
```

Notes

This function is a blocking function. Processing may take time, therefore it should be called from a subthread.

See Also

sceNpWebApiCreateMultipartRequest(), sceNpWebApiAddMultipartPart(), sceNpWebApiReadData()

sceNpWebApiSetMultipartContentType

Set HTTP header Content-Type value upon sending multiple parts

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiSetMultipartContentType(
        int64_t requestId,
        const char *pTypeName,
        const char *pBoundary
);
```

Arguments

requestId	Request ID
pTypeName	Content-Type name for multipart/mixed, etc. (ASCIIZ string)
pBoundary	Boundary string (ASCIIZ string). Specify NULL when it is not to be changed

Return Values

Returns `SCE_OK` (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function is used when changing the HTTP header Content-Type value upon sending of multiple parts.

The default value is "`Content-Type: multipart/mixed; boundary=`*`random string`*".

Examples

```
int32_t ret = 0;
int64_t requestId;

ret = sceNpWebApiSetMultipartContentType(
        requestId, "multipart/mixed", "my_boundary_string");
if(ret < 0){
        /* Error handling */
}
```

See Also

`sceNpWebApiCreateMultipartRequest()`

HTTP Headers

sceNpWebApiAddHttpRequestHeader

Add an HTTP request header

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiAddHttpRequestHeader(
        int64_t requestId,
        const char *pFieldName,
        const char *pValue
);
```

Arguments

requestId	Request ID
pFieldName	Field name of the HTTP header to add (ASCIIZ string)
pValue	Value of the HTTP header to add (ASCIIZ string)

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function adds an HTTP header to the HTTP request upon PSN℠ Web API execution. After creating the request with sceNpWebApiCreateRequest() and before sending the request with sceNpWebApiSendRequest2()/sceNpWebApiSendRequest(), add an HTTP header with this function.

Examples

```
int32_t ret = 0;
int64_t requestId;

ret = sceNpWebApiAddHttpRequestHeader(
        requestId, "My-Header-Name", "My-Header-Value");
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiGetHttpResponseHeaderValue()

sceNpWebApiGetHttpResponseHeaderValue

Get the value of the HTTP response header

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiGetHttpResponseHeaderValue(
        int64_t requestId,
        const char *pFieldName,
        char *pValue,
        size_t valueSize
);
```

Arguments

requestId	Request ID
pFieldName	Field name of the HTTP header to obtain (ASCIIZ string)
pValue	Buffer for storing the value of the obtained HTTP header
valueSize	Size of the buffer specified to *pValue*

Return Values

Stores the value of the obtained HTTP header in *pValue*[] and returns SCE_OK (=0) upon normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function obtains the value of the HTTP header of the HTTP response upon PSN℠ Web API execution. The value of the HTTP response header can be obtained with this function after the normal termination of sceNpWebApiSendRequest2()/sceNpWebApiSendRequest(). Before obtaining the value of the HTTP response header with this function, execute sceNpWebApiGetHttpResponseHeaderValueLength() to obtain the size of the HTTP response header, prepare a sufficient buffer, and specify the buffer to *pValue*.

Examples

```
int32_t ret = 0;
int64_t requestId;
char value[BUF_SIZE];

ret = sceNpWebApiGetHttpResponseHeaderValue(
        requestId, "My-Header-Name", value, sizeof(value));
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiGetHttpResponseHeaderValueLength()

sceNpWebApiGetHttpResponseHeaderValueLength

Get the length of the value of the HTTP response header

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiGetHttpResponseHeaderValueLength(
        int64_t requestId,
        const char *pFieldName,
        size_t *pValueLength
);
```

Arguments

requestId	Request ID
pFieldName	Field name of the HTTP header to obtain (ASCIIZ string)
pValueLength	Destination to store the length of the value of the HTTP header to obtain

Return Values

Stores the obtained length of the value of the HTTP header in **pValueLength* and returns `SCE_OK` (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function obtains the length of the value of a specific HTTP header of an HTTP response upon executing a PSN℠ Web API. The length of the value of a specific HTTP header can be obtained with this function after the normal termination of `sceNpWebApiSendRequest2()`/`sceNpWebApiSendRequest()`. Before obtaining the value of the HTTP response header with `sceNpWebApiGetHttpResponseHeaderValue()`, use this function to obtain the length of the value and to prepare a sufficient buffer.

Examples

```
int32_t ret = 0;
int64_t requestId;
size_t valueLength = 0;

ret = sceNpWebApiGetHttpResponseHeaderValueLength(
        requestId, "My-Header-Name", &valueLength);
if(ret < 0){
        /* Error handling */
}
```

See Also

`sceNpWebApiGetHttpResponseHeaderValue()`

Push Events

SceNpWebApiPushEventCallback

Callback function that is notified of Push events [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
typedef void (*SceNpWebApiPushEventCallback)(
        int32_t callbackId,
        const SceNpPeerAddress *pTo,
        const SceNpPeerAddress *pFrom,
        const SceNpWebApiPushEventDataType *pDataType,
        const char *pData,
        size_t dataLen,
        void *pUserArg
        );
```

Members

callbackId	Callback ID
pTo	User to be notified of Push event
pFrom	User that caused Push event
pDataType	Data type for notified Push event
pData	Data associated with notified Push event, or NULL
dataLen	Size of data associated with notified Push event, or 0
pUserArg	User data

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving Push events.

This is the callback function that is notified of received Push events. The Push event that matches the Push event data type of the Push event filter created with `sceNpWebApiCreatePushEventFilter()` will be notified.

To have this callback function called, have the application call `sceNpWebApiCheckCallback()` at regular intervals.

Whether data is attached to the Push event depends on the type of Push event - refer to the each PSN℠ Web API Reference document. If data is not attached, NULL will be passed to *pData* and 0 will be passed to *dataLen*.

SceNpWebApiPushEventDataType

Push Event Data Type [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiPushEventDataType {
        char val[SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX + 1];
} SceNpWebApiPushEventDataType;
```

Members

val Buffer to store the character string that indicates the Push event data type

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving Push events.

This structure indicates the data type of a Push event. It is used when specifying a Push event to be received with a Push event filter and when identifying the data type of a received Push event.

For details on Push event data types, refer to "PSN℠ Web API Overview" and the reference documents for each PSN℠ Web API.

sceNpWebApiCreatePushEventFilter

Create Push event filter [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCreatePushEventFilter(
        const SceNpWebApiPushEventDataType *pDataType,
        size_t dataTypeNum
);
```

Arguments

pDataType	Array for the data types of Push events to receive
dataTypeNum	Number of elements for the array represented by *pDataType*

Return Values

Returns the filter ID (positive value) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving Push events.

This function creates a filter for specifying the data types of Push events to receive. When registering a Push event callback function, specify the filter ID issued upon normal termination of this function.

Examples

```
int32_t ret = 0;
int32_t filterId = 0;

SceNpWebApiPushEventDataType dataType;
memset(&dataType, 0, sizeof(dataType));
snprintf(dataType.val, SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX,
        "datatype");

ret = sceNpWebApiCreatePushEventFilter(&dataType, 1);
if(ret < 0){
        /* Error handling */
}
filterId = ret;
```

See Also

sceNpWebApiDeletePushEventFilter(),sceNpWebApiRegisterPushEventCallback()

sceNpWebApiDeletePushEventFilter

Delete Push event filter [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiDeletePushEventFilter(
        int32_t filterId
);
```

Arguments

filterId Filter ID for the Push event filter to delete

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving Push events.

This function deletes a created Push event filter.

Examples

```
int32_t ret = 0;
int32_t filterId;

ret = sceNpWebApiDeletePushEventFilter(filterId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiCreatePushEventFilter()

sceNpWebApiRegisterPushEventCallback

Register Push event callback function [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiRegisterPushEventCallback(
        int32_t filterId,
        SceNpWebApiPushEventCallback cbFunc,
        void *pUserArg
);
```

Arguments

filterId	Push event filter ID indicating the Push event to receive
cbFunc	Callback function that is notified of Push events
pUserArg	User data

Return Values

Returns the callback ID (positive value) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving Push events.

This function registers a callback function that is notified of the Push events to receive. Before registering a callback function with this function, calling `sceNpWebApiCreatePushEventFilter()` to create a Push event filter to indicate the Push events to receive is required.

Examples

```
int32_t ret = 0;
int32_t filterId;
int32_t callbackId = 0;

ret = sceNpWebApiRegisterPushEventCallback(
        filterId, cbFunc, NULL);
if(ret < 0){
        /* Error handling */
}
callbackId = ret;
```

See Also

`sceNpWebApiUnregisterPushEventCallback()`,`sceNpWebApiCreatePushEventFilter()`

sceNpWebApiUnregisterPushEventCallback

Unregister Push event callback function [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiUnregisterPushEventCallback(
        int32_t callbackId
);
```

Arguments

callbackId Callback ID of the callback function to unregister

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving Push events.

This function unregisters a registered Push event callback function.

Examples

```
int32_t ret = 0;
int32_t callbackId;

ret = sceNpWebApiUnregisterPushEventCallback(
        callbackId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiRegisterPushEventCallback()

Service Push Events

SceNpWebApiServicePushEventCallback

Callback function that is notified of service Push events [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
typedef void (*SceNpWebApiServicePushEventCallback)(
        int32_t callbackId,
        const char *pNpServiceName,
        SceNpServiceLabel npServiceLabel,
        const SceNpPeerAddress *pTo,
        const SceNpPeerAddress *pFrom,
        const SceNpWebApiPushEventDataType *pDataType,
        const char *pData,
        size_t dataLen,
        void *pUserArg
        );
```

Members

callbackId	Callback ID
pNpServiceName	NP service name
npServiceLabel	NP service label
pTo	User to be notified of Push event
pFrom	User that caused Push event
pDataType	Data type for notified Push event
pData	Data associated with notified Push event, or NULL
dataLen	Size of data associated with notified Push event, or 0
pUserArg	User data

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving service Push events.

This is the callback function that is notified of received service Push events. The service Push event that matches the Push event data type of the Push event filter created with `sceNpWebApiCreateServicePushEventFilter()` will be notified.

To have this callback function called, have the application call `sceNpWebApiCheckCallback()` at regular intervals.

Whether data is attached to service Push event depends on the type of service Push event - refer to the each PSN℠ Web API reference document. If data is not attached, NULL will be passed to *pData* and 0 will be passed to *dataLen*.

sceNpWebApiCreateServicePushEventFilter

Create service Push event filter [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCreateServicePushEventFilter(
        int32_t handleId,
        const char *pNpServiceName,
        SceNpServiceLabel npServiceLabel,
        const SceNpWebApiPushEventDataType *pDataType,
        size_t dataTypeNum
);
```

Arguments

handleId	Handle ID
pNpServiceName	NP service name
npServiceLabel	NP service label. If the NP service label to use is not explicitly specified when requesting NP service usage, specify SCE_NP_DEFAULT_SERVICE_LABEL.
pDataType	Array for the data types of Push events to receive
dataTypeNum	Number of elements for the array represented by *pDataType*

Return Values

Returns the filter ID (positive value) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving service Push events.

This function creates a filter for specifying the data types of service Push events to receive. Note that this function is a blocking function that is associated with network access. When registering a Push event callback function, specify the filter ID issued upon normal termination of this function.

Examples

```
#define NP_SERCIE_NAME "inGamePresence"

int32_t ret = 0;
int32_t handleId;
int32_t filterId = 0;

SceNpWebApiPushEventDataType dataType;
memset(&dataType, 0, sizeof(dataType));
snprintf(dataType.val, SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX,
        "datatype");

// Blocking function
ret = sceNpWebApiCreateServicePushEventFilter(
        handleId, NP_SERVICE_NAME, SCE_NP_DEFAULT_SERVICE_LABEL,
        &dataType, 1);
```

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

See Also

```
sceNpWebApiDeleteServicePushEventFilter(),
sceNpWebApiRegisterServicePushEventCallback()
```

sceNpWebApiDeleteServicePushEventFilter

Delete service Push event filter [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiDeleteServicePushEventFilter(
        int32_t filterId
);
```

Arguments

filterId	Filter ID of Push event filter to delete

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving service Push events.

This function deletes a created service Push event filter.

Examples

```
int32_t ret = 0;
int32_t filterId;

ret = sceNpWebApiDeleteServicePushEventFilter(filterId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiCreateServicePushEventFilter()

sceNpWebApiRegisterServicePushEventCallback

Register service Push event callback function [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiRegisterServicePushEventCallback(
        int32_t filterId,
        SceNpWebApiServicePushEventCallback cbFunc,
        void *pUserArg
);
```

Arguments

filterId	Push event filter ID indicating the Push event to receive
cbFunc	Callback function that is notified of service Push events
pUserArg	User data

Return Values

Returns the callback ID (positive value) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving service Push events.

This function registers a callback function that is notified of received service Push events. Before registering a callback function with this function, calling `sceNpWebApiCreateServicePushEventFilter()` to create a Push event filter to indicate the service Push events to receive is required.

Examples

```
int32_t ret = 0;
int32_t filterId;
int32_t callbackId = 0;

ret = sceNpWebApiRegisterServicePushEventCallback(
        filterId, cbFunc, NULL);
if(ret < 0){
        /* Error handling */
}
callbackId = ret;
```

See Also

`sceNpWebApiUnregisterServicePushEventCallback()`,
`sceNpWebApiCreateServicePushEventFilter()`

sceNpWebApiUnregisterServicePushEventCallback

Unregister service Push event callback function [API for preserving compatibility]

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiUnregisterServicePushEventCallback(
        int32_t callbackId
);
```

Arguments

callbackId Callback ID of callback function to unregister

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving service Push events.

This function unregisters a registered service Push event callback function.

Examples

```
int32_t ret = 0;
int32_t callbackId;

ret = sceNpWebApiUnregisterServicePushEventCallback(
        callbackId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiRegisterServicePushEventCallback()

Extended Push Events

SceNpWebApiExtdPushEventExtdDataKey

Extended Push event extended data key

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiExtdPushEventExtdDataKey {
        char val[SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX + 1];
} SceNpWebApiExtdPushEventExtdDataKey;
```

Members

val Buffer to store character string that represents extended data key for extended Push event

Description

This structure represents an extended data key for an extended Push event. It is used when specifying an extended data key for an extended Push event to be received with an extended Push event filter, and when identifying extended data for a received extended Push event.

For details on extended data keys for extended Push events, refer to the "PSN℠ Web APIs Overview" document and each Web API reference document.

See Also

SceNpWebApiExtdPushEventFilterParameter, SceNpWebApiExtdPushEventExtdData

SceNpWebApiExtdPushEventFilterParameter

Extended Push event filter parameters

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiExtdPushEventFilterParameter {
        SceNpWebApiPushEventDataType dataType;
        char padding[3];
        SceNpWebApiExtdPushEventExtdDataKey *pExtdDataKey;
        size_t extdDataKeyNum;
} SceNpWebApiExtdPushEventFilterParameter;
```

Members

dataType	Data type of extended Push event to receive
padding	Padding
pExtdDataKey	Array of extended data keys for extended Push event to receive
extdDataKeyNum	Number of elements in array represented by *pExtdDataKey*

Description

This structure represents the parameters to specify when creating an extended Push event filter.

See Also

sceNpWebApiCreateExtdPushEventFilter()

SceNpWebApiExtdPushEventExtdData

Extended Push event extended data

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiExtdPushEventExtdData {
        SceNpWebApiExtdPushEventExtdDataKey extdDataKey;
        char padding[3];
        char *pData;
        size_t dataLen;
} SceNpWebApiExtdPushEventExtdData;
```

Members

extdDataKey	Extended data key for received extended Push event
padding	Padding
pData	Extended data for received extended Push event
dataLen	Length of extended data for received extended Push event

Description

This structure represents extended data for a received extended Push event.

See Also

SceNpWebApiExtdPushEventCallback

SceNpWebApiExtdPushEventCallback

Callback function that is notified of extended Push events

Definition

```
#include <np/np_webapi.h>
typedef void (*SceNpWebApiExtdPushEventCallback)(
        int32_t callbackId,
        const char *pNpServiceName,
        SceNpServiceLabel npServiceLabel,
        const SceNpPeerAddress *pTo,
        const SceNpPeerAddress *pFrom,
        const SceNpWebApiPushEventDataType *pDataType,
        const char *pData,
        size_t dataLen,
        const SceNpWebApiExtdPushEventExtdData *pExtdData,
        size_t extdDataNum,
        void *pUserArg
        );
```

Members

callbackId	Callback ID
pNpServiceName	NP service name, or NULL
npServiceLabel	NP service label, or SCE_NP_INVALID_SERVICE_LABEL
pTo	User to notify of Push event
pFrom	User where Push event occurred
pDataType	Data type of notified extended Push event
pData	Data to attach to notified extended Push event, or NULL
dataLen	Size of data to attach to notified extended Push event, or 0
pExtdData	Array of extended data to attach to notified extended Push event, or NULL
extdDataNum	Number of elements in array represented by *pExtdData*
pUserArg	User data

Description

This is a callback function that is notified of a received extended Push event. It is possible for an extended Push event that matches the data type of the extended Push event filter created with sceNpWebApiCreateExtdPushEventFilter() to be notified and then obtain the extended data that matches the extended data key.

An application should call sceNpWebApiCheckCallback() at regular intervals in order for this callback function to be called.

Whether data or extended data is attached to an extended Push event or not depends on the type of Push event, so refer to each Web API reference document. If data will not be attached, NULL will be passed to *pData*, and 0 will be passed to *dataLen*. If extended data that matches the extended data key does not exist, NULL will be passed to *pExtdData*, and 0 will be passed to *extdDataNum*.

sceNpWebApiAbortHandle

Abort handle processing

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiAbortHandle(
        int32_t handleId
);
```

Arguments

handleId Handle ID of handle for which processing will be aborted

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function aborts the processing of the sceNpWebApiCreateExtdPushEventFilter() and sceNpWebApiCreateServicePushEventFilter() being executed with the handle indicated by *handleId*.

Examples

```
int32_t ret = 0;
int32_t handleId;

ret = sceNpWebApiAbortHandle(handletId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiCreateExtdPushEventFilter(),
sceNpWebApiCreateServicePushEventFilter()

sceNpWebApiCreateHandle

Create handle

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCreateHandle(
        void
);
```

Arguments

None

Return Values

Returns the handle ID (positive value) for normal termination.
Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function creates the handle required for execution of `sceNpWebApiCreateExtdPushEventFilter()` and `sceNpWebApiCreateServicePushEventFilter()`.

Examples

```
int32_t ret = 0;
int32_t handleId = 0;

ret = sceNpWebApiCreateHandle();
if(ret < 0){
        /* Error handling */
}
handleId = ret;
```

See Also

`sceNpWebApiDeleteHandle()`,`sceNpWebApiAbortHandle()`,
`sceNpWebApiCreateExtdPushEventFilter()`,
`sceNpWebApiCreateServicePushEventFilter()`

sceNpWebApiCreateExtdPushEventFilter

Create extended Push event filter

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiCreateExtdPushEventFilter(
        int32_t handleId,
        const char *pNpServiceName,
        SceNpServiceLabel npServiceLabel,
        const SceNpWebApiExtdPushEventFilterParameter *pFilterParam,
        size_t filterParamNum
);
```

Arguments

handleId	Handle ID
pNpServiceName	NP service name
npServiceLabel	NP service label or SCE_NP_DEFAULT_SERVICE_LABEL
pFilterParam	Array of extended Push event filter parameters
filterParamNum	Number of elements in array represented by *pFilterParam*

Return Values

Returns the filter ID (positive value) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function creates a filter for specifying extended Push events to receive. When registering an extended Push event callback function, specify the filter ID issued upon normal termination of this function.

For *npServiceLabel*, specify the NP service label that was specified during the request for NP service usage. When not explicitly specifying an NP service label, specify SCE_NP_DEFAULT_SERVICE_LABEL.

Examples

```
#define NP_SERCIE_NAME "npServiceName"

int32_t ret = 0;
int32_t handleId;
int32_t filterId = 0;

SceNpWebApiPushEventDataType dataType[2];
SceNpWebApiExtdPushEventFilterParameter filterParam[2];
SceNpWebApiExtdPushEventExtdDataKey extdDataKey1[2], extdDataKey2[2];

memset(dataType, 0, sizeof(dataType));
snprintf(dataType[0].val, SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX,
        "dataType1");
snprintf(dataType[1].val, SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX,
        "dataType2");

memset(extdDataKey1, 0, sizeof(extdDataKey1));
```

```
snprintf(extdDataKey1[0].val,
        SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX, "key1-1");
snprintf(extdDataKey1[1].val,
        SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX, "key1-2");

memset(extdDataKey2, 0, sizeof(extdDataKey2));
snprintf(extdDataKey2[0].val,
        SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX, "key2-1");
snprintf(extdDataKey2[1].val,
        SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX, "key2-2");

memset(filterParam, 0, sizeof(filterParam));
memcpy(&filterParam[0].dataType, &dataType[0],
        sizeof(SceNpWebApiPushEventDataType));
filterParam[0].pExtdDataKey = extdDataKey1;
filterParam[0].extdDataKeyNum = 2;
memcpy(&filterParam[1].dataType, &dataType[1],
        sizeof(SceNpWebApiPushEventDataType));
filterParam[1].pExtdDataKey = extdDataKey2;
filterParam[1].extdDataKeyNum = 2;

// Blocking function
ret = sceNpWebApiCreateExtdPushEventFilter(
        handleId, NP_SERVICE_NAME, SCE_NP_DEFAULT_SERVICE_LABEL,
        filterParam, 2);
if(ret < 0){
        /* Error handling */
}
filterId = ret;
```

Notes

This function is a blocking function. Processing may take time, therefore it should be called from a subthread.

See Also

```
sceNpWebApiDeleteExtdPushEventFilter(),
sceNpWebApiRegisterExtdPushEventCallback()
```

sceNpWebApiDeleteHandle

Delete handle

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiDeleteHandle(
        int32_t handleId
);
```

Arguments

handleId Handle ID of handle to delete

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function deletes a created handle.

Examples

```
int32_t ret = 0;
int32_t handleId;

ret = sceNpWebApiDeleteHandle(handleId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiCreateHandle()

sceNpWebApiDeleteExtdPushEventFilter

Delete extended Push event filter

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiDeleteExtdPushEventFilter(
        int32_t filterId
);
```

Arguments

filterId Filter ID for the extended Push event filter to delete

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function deletes a created extended Push event filter.

Examples

```
int32_t ret = 0;
int32_t filterId;

ret = sceNpWebApiDeleteExtdPushEventFilter(filterId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiCreateExtdPushEventFilter()

sceNpWebApiRegisterExtdPushEventCallback

Register extended Push event callback function

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiRegisterExtdPushEventCallback(
        int32_t filterId,
        SceNpWebApiExtdPushEventCallback cbFunc,
        void *pUserArg
);
```

Arguments

filterId	Extended Push event filter ID indicating the extended Push event to receive
cbFunc	Callback function that is notified of extended Push events
pUserArg	User data

Return Values

Returns the callback ID (positive value) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function registers a callback function that is notified of received extended Push events. Before registering a callback function with this function, calling `sceNpWebApiCreateExtdPushEventFilter()` to create an extended Push event filter to indicate the extended Push events to receive is required.

Specify the callback ID issued for normal termination of this function when unregistering an extended Push event callback function.

Examples

```
int32_t ret = 0;
int32_t filterId;
int32_t callbackId = 0;

ret = sceNpWebApiRegisterExtdPushEventCallback(
        filterId, cbFunc, NULL);
if(ret < 0){
        /* Error handling */
}
callbackId = ret;
```

See Also

`sceNpWebApiUnregisterExtdPushEventCallback()`,
`sceNpWebApiCreateExtdPushEventFilter()`

sceNpWebApiUnregisterExtdPushEventCallback

Unregister extended Push event callback function

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiUnregisterExtdPushEventCallback(
        int32_t callbackId
);
```

Arguments

callbackId Callback ID of the callback function to unregister

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function unregisters a registered extended Push event callback function.

Examples

```
int32_t ret = 0;
int32_t callbackId;

ret = sceNpWebApiUnregisterExtdPushEventCallback(
        callbackId);
if(ret < 0){
        /* Error handling */
}
```

See Also

sceNpWebApiRegisterExtdPushEventCallback()

Utilities

sceNpWebApiUtilityParseNpId

Convert JSON format data npId to SceNpId structure

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiUtilityParseNpId(
        const char *pJsonNpId,
        SceNpId *pNpId
);
```

Arguments

pJsonNpId	npId value included in the PSN℠ Web API JSON format data
pNpId	Storage destination for the results of the npId value converted to an SceNpId structure, or NULL

Return Values

Stores the conversion results in **pNpId* and returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function parses a key npId value included in JSON format response data which can be obtained with the PSN℠ Web API execution result and converts it to an SceNpId structure.

Examples

```
#define JSON_NPID_VALUE "dXRpbGl0eVBhcnNlTnBJZEBhMC51cy9wczQ="

int32_t ret = 0;
SceNpId npId;

ret = sceNpWebApiUtilityParseNpId(JSON_NPID_VALUE, &npId);
if(ret < 0){
        /* Error handling */
}
```

Memory

SceNpWebApiMemoryPoolStats

NpWebApi library memory information

Definition

```
#include <np/np_webapi.h>
typedef struct SceNpWebApiMemoryPoolStats {
        size_t poolSize;
        size_t maxInuseSize;
        size_t currentInuseSize;
        int32_t reserved;
} SceNpWebApiMemoryPoolStats;
```

Members

poolSize	Memory pool size for the NpWebApi library
maxInuseSize	Maximum memory size used by the NpWebApi library
currentInuseSize	Size of the memory currently being used by the NpWebApi library
reserved	Reserved area

Description

This structure indicates NpWebApi library memory information.

sceNpWebApiGetMemoryPoolStats

Get NpWebApi library memory information

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiGetMemoryPoolStats(
        SceNpWebApiMemoryPoolStats *pCurrentStat
);
```

Arguments

pCurrentStat Storage destination for memory information obtain results, or NULL

Return Values

Stores the obtained memory information in **pCurrentStat* and returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details)

Description

This function obtains NpWebApi library memory information. Use it when determining the memory pool size to specify during initialization, etc.

Examples

```
int32_t ret = 0;
SceNpWebApiMemoryPoolStats stats;

memset(&stats, 0, sizeof(stats));
ret = sceNpWebApiGetMemoryPoolStats(&stats);
if(ret < 0){
        /* Error handling */
}
```

See Also

SceNpWebApiMemoryPoolStats

NP Title ID

sceNpWebApiSetNpTitleId

Set the NP Title ID and NP Title Secret

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiSetNpTitleId(
        const SceNpTitleId *pTitleId,
        const SceNpTitleSecret *pTitleSecret
);
```

Arguments

pTitleId	NP Title ID
pTitleSecret	NP Title Secret

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function sets the NP Title ID and NP Title Secret.

The NP Title ID and NP Title Secret to be set with this function are only valid when the **Release Check Mode** of ★**Debug Settings** is **Development Mode**. This function does not use the NP Title ID and NP Title Secret stored in the nptitle.dat file; use this function to specify an arbitrary NP Title ID and NP Title Secret from the program.

When the **Release Check Mode** of ★**Debug Settings** is **Release Mode**, the NP Title ID and NP Title Secret set with this function will be ignored and nptitle.dat will always be used.

See Also

SceNpTitleId, SceNpTitleSecret

sceNpWebApiGetNpTitleId

Get the NP Title ID and NP Title Secret

Definition

```
#include <np/np_webapi.h>
int32_t sceNpWebApiGetNpTitleId(
        SceNpTitleId *pTitleId,
        SceNpTitleSecret *pTitleSecret
);
```

Arguments

pTitleId	Area to store the NP Title ID
pTitleSecret	Area to store the NP Title Secret

Return Values

Returns `SCE_OK` (=0) for normal termination.

Returns an error code (negative value) for errors (refer to "Return Codes" for details).

Description

This function obtains the NP Title ID and NP Title Secret set using `sceNpWebApiSetNpTitleId()`.

See Also

`sceNpWebApiSetNpTitleId()`

Common Constants

SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX

Maximum length for Push event data type

Definition

Value	(Number)	Description
SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX	64	Maximum length for Push event data type

SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX

Maximum length for extended Push event data key

Definition

Value	(Number)	Description
SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX	32	Maximum length for extended Push event data key

SCE_NP_WEBAPI_NP_SERVICE_NAME_NONE

Constant indicating that there is no NP service name

Definition

Value	(Number)	Description
SCE_NP_WEBAPI_NP_SERVICE_NAME_NONE	NULL	There is no NP service name

Return Codes

List of return codes returned by the NpWebApi library

Definition

Value	(Number)	Description
SCE_NP_WEBAPI_ERROR_OUT_OF_MEMORY	0x80552901	Insufficient memory
SCE_NP_WEBAPI_ERROR_INVALID_ARGUMENT	0x80552902	Argument is invalid
SCE_NP_WEBAPI_ERROR_INVALID_LIB_CONTEXT_ID	0x80552903	Library context ID is invalid (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_LIB_CONTEXT_NOT_FOUND	0x80552904	Library context could not be found (may return when an invalid request ID has been assigned)
SCE_NP_WEBAPI_ERROR_USER_CONTEXT_NOT_FOUND	0x80552905	User context could not be found (may return when an invalid request ID has been assigned)
SCE_NP_WEBAPI_ERROR_REQUEST_NOT_FOUND	0x80552906	Request could not be found
SCE_NP_WEBAPI_ERROR_NOT_SIGNED_IN	0x80552907	Function to be executed during a signed-in state was executed during a non-signed in state
SCE_NP_WEBAPI_ERROR_INVALID_CONTENT_PARAMETER	0x80552908	Content parameter is invalid
SCE_NP_WEBAPI_ERROR_ABORTED	0x80552909	Processing was aborted
SCE_NP_WEBAPI_ERROR_USER_CONTEXT_ALREADY_EXIST	0x8055290a	User context generated with the specified online ID already exists (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_PUSH_EVENT_FILTER_NOT_FOUND	0x8055290b	Push event filter was not found
SCE_NP_WEBAPI_ERROR_PUSH_EVENT_CALLBACK_NOT_FOUND	0x8055290c	Push event callback was not found
SCE_NP_WEBAPI_ERROR_HANDLE_NOT_FOUND	0x8055290d	Handle was not found
SCE_NP_WEBAPI_ERROR_SERVICE_PUSH_EVENT_FILTER_NOT_FOUND	0x8055290e	Service Push event filter was not found
SCE_NP_WEBAPI_ERROR_SERVICE_PUSH_EVENT_CALLBACK_NOT_FOUND	0x8055290f	Service Push event callback was not found
SCE_NP_WEBAPI_ERROR_SIGNED_IN_USER_NOT_FOUND	0x80552910	Signed in user was not found
SCE_NP_WEBAPI_ERROR_LIB_CONTEXT_BUSY	0x80552911	Library context is being used and termination processing cannot be carried out (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_USER_CONTEXT_BUSY	0x80552912	User context is being used and cannot be deleted (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_REQUEST_BUSY	0x80552913	Request is being used and cannot be deleted (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_INVALID_HTTP_STATUS_CODE	0x80552914	HTTP status code is invalid (will not return from the APIs in this library)

Value	(Number)	Description
SCE_NP_WEBAPI_ERROR_ PROHIBITED_HTTP_HEADER	0x80552915	Specified an HTTP header for which specification is prohibited (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_ PROHIBITED_FUNCTION_CALL	0x80552916	Executed a function for which usage is prohibited.
SCE_NP_WEBAPI_ERROR_ MULTIPART_PART_NOT_FOUND	0x80552917	The part of the multiple parts specified by the part index number does not exist
SCE_NP_WEBAPI_ERROR_ PARAMETER_TOO_LONG	0x80552918	Specified parameter is too long
SCE_NP_WEBAPI_ERROR_ HANDLE_BUSY	0x80552919	Attempted to delete a handle while the handle was being used (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_ LIB_CONTEXT_MAX	0x8055291a	Reached the maximum number of library contexts (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_ USER_CONTEXT_MAX	0x8055291b	Reached the maximum number of user contexts (will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_ EXTD_PUSH_EVENT_ FILTER_NOT_FOUND	0x8055291c	Extended Push event filter was not found
SCE_NP_WEBAPI_ERROR_ EXTD_PUSH_EVENT_ CALLBACK_NOT_FOUND	0x8055291d	Extended Push event callback was not found
SCE_NP_WEBAPI_ERROR_ AFTER_SEND	0x8055291e	Request is already sent
SCE_NP_WEBAPI_ERROR_ NOT_INITIALIZED	0x80552a01	Not initialized
SCE_NP_WEBAPI_ERROR_ ALREADY_INITIALIZED	0x80552a02	Already initialized
SCE_NP_WEBAPI_ERROR_ SSL_ERR_CN_CHECK	0x80552a03	SSL error: common name check of the server certificate failed
SCE_NP_WEBAPI_ERROR_ SSL_ERR_UNKNOWN_CA	0x80552a04	SSL error: no certificate for the root CA certificate that issued the server certificate
SCE_NP_WEBAPI_ERROR_ SSL_ERR_NOT_AFTER_CHECK	0x80552a05	SSL error: the validity period of the server certificate has expired
SCE_NP_WEBAPI_ERROR_ SSL_ERR_NOT_BEFORE_CHECK	0x80552a06	SSL error: the validity period of the server certificate has not started yet
SCE_NP_WEBAPI_ERROR_ SSL_ERR_INVALID_CERT	0x80552a07	SSL error: format of the server certificate is invalid
SCE_NP_WEBAPI_ERROR_ SSL_ERR_INTERNAL	0x80552a08	SSL error: internal error of the library
SCE_NP_WEBAPI_ERROR_ TITLE_IS_BANNED	0x80552a09	PSN℠ Web API usage from this title is temporarily suspended
SCE_NP_WEBAPI_ERROR_ NP_TITLE_DAT_NOT_FOUND	0x80552a0a	nptitle.dat file not found
SCE_NP_WEBAPI_ERROR_ TITLE_ID_NOT_MATCHED_ TO_NP_TITLE_ID	0x80552a0b	Title ID does not match NP Title ID Check param.sfo and nptitle.dat or the value specified for `sceNpWebApiSetNpTitleId()`.

In addition to the above error codes, `sceNpWebApiSendRequest2()` and `sceNpWebApiSendMultipartRequest2()` return an error code starting with 0x82. An error code starting with 0x82 indicates a server error; the lower 24 bits excluding 0x82 of this error code expressed as a decimal number serves as the value for determining the specific server error. Refer to the each PSN℠ Web API reference document for server error definitions.