

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

# **Table of Contents**

Initialization/Termination and Check Callback	4
sceNpWebApiInitialize	5
sceNpWebApiTerminate	6
sceNpWebApiCheckCallback	7
Requests	8
SCE_NP_WEBAPI_CONTENT_TYPE_APPLICATION_JSON_UTF8	9
SCE_NP_WEBAPI_HTTP_METHOD_XXX	10
SceNpWebApiContentParameter	11
SceNpWebApiResponseInformationOption	
sceNpWebApiAbortRequest	13
sceNpWebApiCreateRequest	14
sceNpWebApiDeleteRequest	16
sceNpWebApiGetErrorCode	17
sceNpWebApiGetHttpStatusCode	18
sceNpWebApiReadData	19
sceNpWebApiSendRequest	21
sceNpWebApiSendRequest2	
Multiple Parts	25
SceNpWebApiHttpHeader	26
SceNpWebApiMultipartPartParameter	
sceNpWebApiAddMultipartPart	
sceNpWebApiCreateMultipartRequest	
sceNpWebApiSendMultipartRequest	
sceNpWebApiSendMultipartRequest2	34
sceNpWebApiSetMultipartContentType	
HTTP Headers	37
sceNpWebApiAddHttpRequestHeader	
sceNpWebApiGetHttpResponseHeaderValue	
sceNpWebApiGetHttpResponseHeaderValueLength	
Push Events	
SceNpWebApiPushEventCallback	
SceNpWebApiPushEventDataType	
sceNpWebApiCreatePushEventFilter	
sceNpWebApiDeletePushEventFilter	
sceNpWebApiRegisterPushEventCallback	
sceNpWebApiUnregisterPushEventCallback	
Service Push Events	
SceNpWebApiServicePushEventCallback	
sceNpWebApiCreateServicePushEventFilter	
sceNpWebApiDeleteServicePushEventFiltersceNpWebApiDeleteServicePushEventFilter	
sceNpWebApiRegisterServicePushEventCallback	
sceNpWebApiUnregisterServicePushEventCallbacksceNpWebApiUnregisterServicePushEventCallback	
· · · · ·	
Extended Push Events	55

	SceNpWebApiExtdPushEventExtdDataKey	56
	SceNpWebApiExtdPushEventFilterParameter	57
	SceNpWebApiExtdPushEventExtdData	58
	SceNpWebApiExtdPushEventCallback	59
	sceNpWebApiAbortHandle	60
	sceNpWebApiCreateHandle	61
	sceNpWebApiCreateExtdPushEventFilter	62
	sceNpWebApiDeleteHandle	64
	sceNpWebApiDeleteExtdPushEventFilter	65
	sceNpWebApiRegisterExtdPushEventCallback	66
	sceNpWebApiUnregisterExtdPushEventCallback	67
Utilities		
	sceNpWebApiUtilityParseNpId	69
Memory	/	70
,	SceNpWebApiMemoryPoolStats	
	sceNpWebApiGetMemoryPoolStats	72
NP Title	ID	73
	sceNpWebApiSetNpTitleId	74
	sceNpWebApiGetNpTitleId	75
Commo	on Constants	76
	SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX	
	SCE_NP_WEBAPI_EXTD_PUSH_EVENT_EXTD_DATA_KEY_LEN_MAX	
	SCE NP WEBAPI NP SERVICE NAME NONE	
	Return Codes	



# sceNpWebApiInitialize

Initialize NpWebApi library

### **Definition**

### **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**

### **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**

### **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**

### **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**

### **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**

### **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()
```

**©SCEI** 



# 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;	Content-Type representing JSON
APPLICATION_JSON_UTF8	charset=utf-8"	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.



# SCE\_NP\_WEBAPI\_HTTP\_METHOD\_XXX

HTTP method upon PSN<sup>™</sup> 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<sup>SM</sup> Web API

execution (bytes)

Character string to be set for Content-Type of the HTTP header (ASCIIZ pContentType

string)

reserved Reserved area

### Description

This structure holds the parameters for the data to send as a request body when executing a PSN™

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**

### **Arguments**

requestId RequestID

### **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**

### See Also

sceNpWebApiSendRequest2(),sceNpWebApiSendRequest(),sceNpWebApiReadData()

# sceNpWebApiCreateRequest

### Create request

### **Definition**

### **Arguments**

pApi GroupAPI group of the PSN™ Web API to execute (ASCIIZ string)pPathPath of the PSN™ Web API to execute (ASCIIZ string)methodHTTP 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<sup>SM</sup> 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<sup> $\odot$ </sup> 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 $^{\text{M}}$  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**

### See Also

sceNpWebApiDeleteRequest(), SceNpWebApiContentParameter



# Document serial number: 000004892117

# sceNpWebApiDeleteRequest

### Delete request

### **Definition**

### **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**

### See Also

sceNpWebApiCreateRequest(

# sceNpWebApiGetErrorCode

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

### **Definition**

### **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<sup>SM</sup> 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()

**©SCEI** 

# sceNpWebApiGetHttpStatusCode

Obtain HTTP status code [API for preserving compatibility]

### **Definition**

### **Arguments**

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<sup>SM</sup> 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<sup>SM</sup> Web API, refer to "PSN<sup>SM</sup> Web APIs Overview" and the reference documents for each PSN<sup>SM</sup> Web API.

### **Examples**

### See Also

sceNpWebApiSendRequest()

# sceNpWebApiReadData

### Receive response body

### **Definition**

### **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**

### **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<sup>™</sup> Web API [API for preserving compatibility]

### **Definition**

### **Arguments**

requestId Request ID

pData Size 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 $^{\mathbb{N}}$  Web API

If there is data to send as a request body upon PSN<sup>SM</sup> 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 <code>contentLength</code> of <code>\*pContentParameter</code> when creating the request with <code>sceNpWebApiCreateRequest()</code>.

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

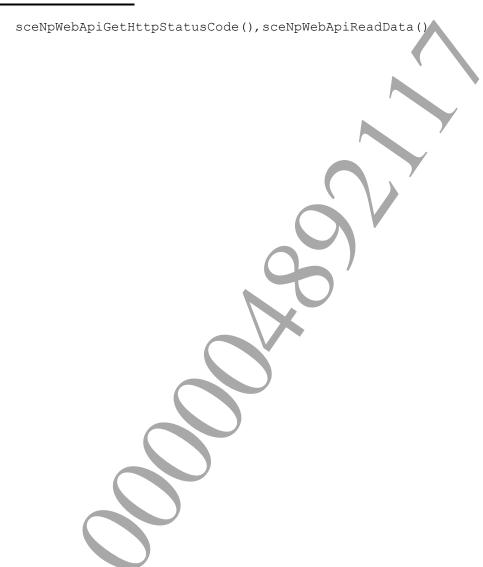
- If there is no data to send as a request body:

  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 <code>contentLength</code> of <code>\*pContentParameter</code> 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**

### See Also



# sceNpWebApiSendRequest2

Send request and execute Web API

### **Definition**

### **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 <code>contentLength</code> of <code>\*pContentParameter</code> when creating the request with <code>sceNpWebApiCreateRequest()</code>.

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 <code>contentLength</code> of <code>\*pContentParameter</code> 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.

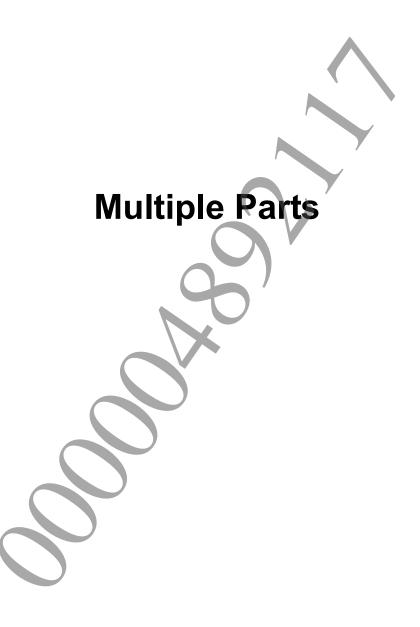
### **Examples**

### **Notes**

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

### See Also

sceNpWebApiReadData()



# 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;
```

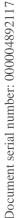
### **Members**

pHeaders Array of headers to insert in part in multiple parts Number of elements in array pointed to by pHeaders headerNum 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<sup>SM</sup> 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**

### **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; ch
headers[1].pName = "Content-Description";
                     "application/json; charset=utf-8";
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**

### **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<sup>SM</sup> 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.

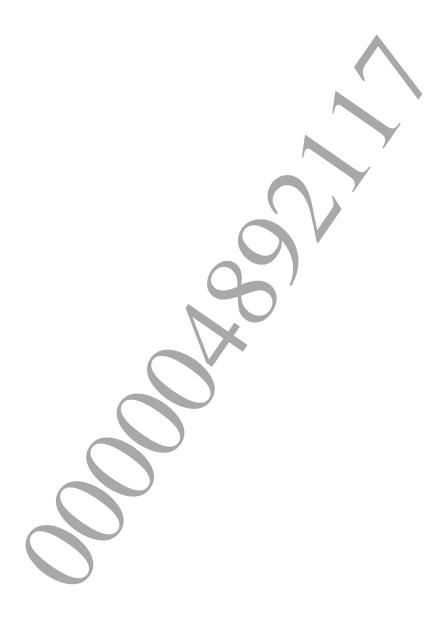
Value	(Number)	Description
SCE_NP_WEBAPI_HTTP_METHOD_GET	0	GET method
SCE_NP_WEBAPI_HTTP_METHOD_POST	1	POST method
SCE_NP_WEBAPI_HTTP_METHOD_PUT	2	PUT method
SCE_NP_WEBAPI_HTTP_METHOD_DELETE	3	DELETE method

### **Examples**

**©SCEI** 

### See Also

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



# sceNpWebApiSendMultipartRequest

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

### **Definition**

### **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 <code>partIndex</code> of the request specified with <code>requestId</code> to a server. By executing this function for all parts added to a request, execution of PSN<sup>SM</sup> 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 <code>contentLength</code> in <code>SceNpWebApiMultipartPartParameter</code> when adding the parts using <code>sceNpWebApiAddMultipartPart()</code>

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 <code>contentLength</code> 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**

### **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**

### **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 <code>partIndex</code> of the request specified by <code>requestId</code> 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 <code>contentLength</code> of <code>SceNpWebApiMultipartPartParameter</code> upon adding the part with <code>sceNpWebApiAddMultipartPart()</code>.

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 <code>contentLength</code> 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**

### **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**

### **Arguments**

```
requestId Request ID

pTypeName Content-Type name for multipart/mixed, etc. (ASCIIZ string)

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**

### See Also

sceNpWebApiCreateMultipartRequest()



# sceNpWebApiAddHttpRequestHeader

# Add an HTTP request header

# **Definition**

# **Arguments**

requestId Request ID

pFieldName Field name of the HTTP header to add (ASCIIZ string)

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 $^{\text{M}}$  Web API execution. After creating the request with sceNpWebApiCreateRequest() and before sending the request with sceNpWebApiSendRequest()/sceNpWebApiSendRequest(), add an HTTP header with this function.

# **Examples**

# See Also

sceNpWebApiGetHttpResponseHeaderValue()

# sceNpWebApiGetHttpResponseHeaderValue

Get the value of the HTTP response header

### **Definition**

# **Arguments**

requestId Request ID

pFieldName Field name of the HTTP header to obtain (ASCIIZ string)
pValue
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 sceNpWebApiSendRequest()/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**

# See Also

sceNpWebApiGetHttpResponseHeaderValueLength()

# sceNpWebApiGetHttpResponseHeaderValueLength

Get the length of the value of the HTTP response header

### **Definition**

# **Arguments**

```
requestIdRequest IDpFieldNameField name of the HTTP header to obtain (ASCIIZ string)pValueLengthDestination 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

 ${\tt sceNpWebApiSendRequest2()/sceNpWebApiSendRequest().} Before obtaining the value of the HTTP response header with {\tt sceNpWebApiGetHttpResponseHeaderValue(), use this function to obtain the length of the value and to prepare a sufficient buffer.}$ 

# **Examples**

### See Also

sceNpWebApiGetHttpResponseHeaderValue()



# **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

pToUser to be notified of Push eventpFromUser that caused Push eventpDataTypeData 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<sup>SM</sup> 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**

# **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**

# **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**

# See Also

sceNpWebApiDeletePushEventFilter(), sceNpWebApiRegisterPushEventCallback()

# sceNpWebApiDeletePushEventFilter

Delete Push event filter [API for preserving compatibility]

# **Definition**

# **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**

# See Also

sceNpWebApiCreatePushEventFilter()

# sceNpWebApiRegisterPushEventCallback

Register Push event callback function [API for preserving compatibility]

# **Definition**

# **Arguments**

filterIdPush event filter ID indicating the Push event to receivecbFuncCallback function that is notified of Push eventspUserArgUser 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 <code>sceNpWebApiCreatePushEventFilter()</code> to create a Push event filter to indicate the Push events to receive is required.

# **Examples**

# See Also

sceNpWebApiUnregisterPushEventCallback(), sceNpWebApiCreatePushEventFilter()

# sceNpWebApiUnregisterPushEventCallback

Unregister Push event callback function [API for preserving compatibility]

# **Definition**

# **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**

# See Also

sceNpWebApiRegisterPushEventCallback()



# **SceNpWebApiServicePushEventCallback**

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

### **Definition**

# **Members**

callbackIdCallback IDpNpServiceNameNP service namenpServiceLabelNP service label

pToUser to be notified of Push eventpFromUser that caused Push eventpDataTypeData type for notified Push event

pDataData associated with notified Push event, or NULLdataLenSize 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

 ${\tt sceNpWebApiCreateServicePushEventFilter()} \ \ will \ be \ notified.$ 

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<sup>™</sup> 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**

# **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.

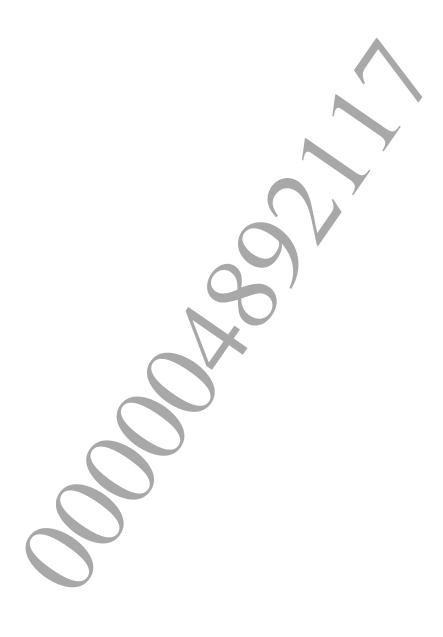
This function creates a filter for specifying the data types of service Push events to receive. Note that this function is a blocking function that is associated with network access. When registering a Push event callback function, specify the filter ID issued upon normal termination of this function.

# **Examples**

**©SCEI** 

# See Also

sceNpWebApiDeleteServicePushEventFilter(),
sceNpWebApiRegisterServicePushEventCallback()



# sceNpWebApiDeleteServicePushEventFilter

Delete service Push event filter [API for preserving compatibility]

# **Definition**

# **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**

# See Also

sceNpWebApiCreateServicePushEventFilter()

# sceNpWebApiRegisterServicePushEventCallback

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

### **Definition**

# **Arguments**

filterIdPush event filter ID indicating the Push event to receivecbFuncCallback function that is notified of service Push eventspUserArgUser 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**

# See Also

```
sceNpWebApiUnregisterServicePushEventCallback(),
sceNpWebApiCreateServicePushEventFilter()
```

# sceNpWebApiUnregisterServicePushEventCallback

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

# **Definition**

# **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**

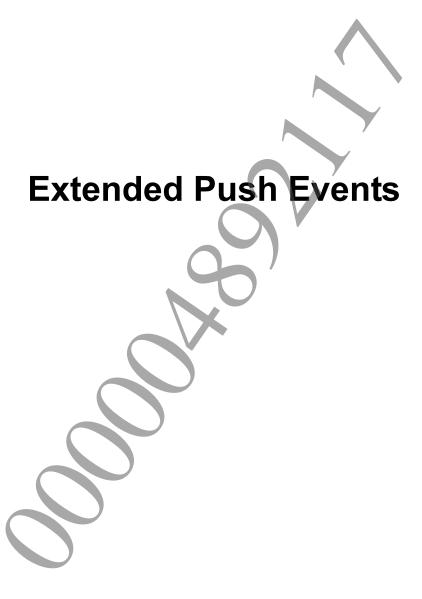
Note: This API is provided for preserving compatibility. Using the APIs that handle extended Push events is recommended for receiving service Push events.

This function unregisters a registered service Push event callback function.

# **Examples**

# See Also

sceNpWebApiRegisterServicePushEventCallback()



# 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

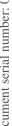
# **Description**

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**

# **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**

# **Members**

extdDataKeyExtended data key for received extended Push eventpaddingPaddingpDataExtended data for received extended Push eventdataLenLength 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 <code>sceNpWebApiCreateExtdPushEventFilter()</code> 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**

# **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**

# See Also

sceNpWebApiCreateExtdPushEventFilter(),
sceNpWebApiCreateServicePushEventFilter()

# sceNpWebApiCreateHandle

# Create handle

# **Definition**

# **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**

# See Also

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

# sceNpWebApiCreateExtdPushEventFilter

Create extended Push event filter

### **Definition**

# **Arguments**

handleIdHandle IDpNpServiceNameNP service namenpServiceLabelNP service label or SCE\_NP\_DEFAULT\_SERVICE\_LABELpFilterParamArray of extended Push event filter parametersfilterParamNumNumber 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**

```
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()

# Document serial number: 000004892117

# sceNpWebApiDeleteHandle

# Delete handle

# **Definition**

# **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**

# See Also

sceNpWebApiCreateHandle()

# sceNpWebApiDeleteExtdPushEventFilter

Delete extended Push event filter

# **Definition**

# **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**

# See Also

sceNpWebApiCreateExtdPushEventFilter()

# sceNpWebApiRegisterExtdPushEventCallback

Register extended Push event callback function

### **Definition**

# **Arguments**

filterIdExtended Push event filter ID indicating the extended Push event to receivecbFuncCallback function that is notified of extended Push eventspUserArgUser 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**

# See Also

```
sceNpWebApiUnregisterExtdPushEventCallback(),
sceNpWebApiCreateExtdPushEventFilter()
```

# sceNpWebApiUnregisterExtdPushEventCallback

Unregister extended Push event callback function

# **Definition**

# **Arguments**

callback Id 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).

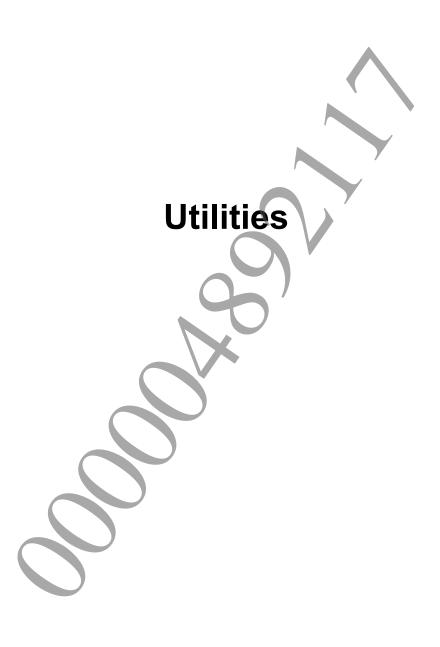
# **Description**

This function unregisters a registered extended Push event callback function.

# **Examples**

# See Also

sceNpWebApiRegisterExtdPushEventCallback()



# sceNpWebApiUtilityParseNpId

Convert JSON format data npld to SceNpld structure

# **Definition**

# **Arguments**

pJsonNpId
pNpId

npId value included in the PSN $^{\text{M}}$  Web API JSON format data Storage destination for the results of the npId value converted to an SceNpId structure, or NULL

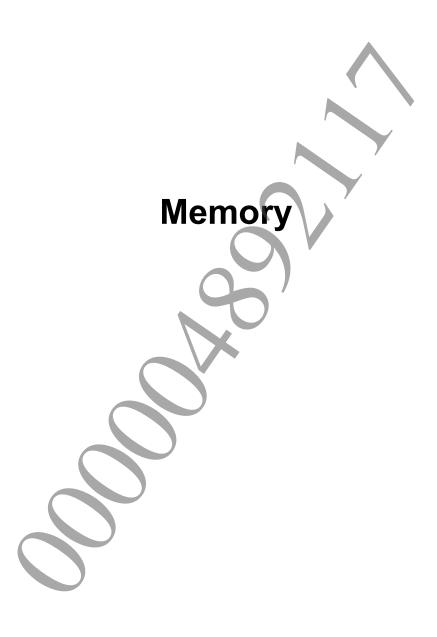
# **Return Values**

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<sup>SM</sup> Web API execution result and converts it to an SceNpId structure.

# **Examples**



# **SceNpWebApiMemoryPoolStats**

# NpWebApi library memory information

# **Definition**

# **Members**

poolSize
maxInuseSize
currentInuseSize
reserved

Memory pool size for the NpWebApi library Maximum memory size used by the NpWebApi library Size of the memory currently being used by the NpWebApi library

Reserved area

# **Description**

This structure indicates NpWebApi library memory information.



# sceNpWebApiGetMemoryPoolStats

Get NpWebApi library memory information

# **Definition**

# **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)

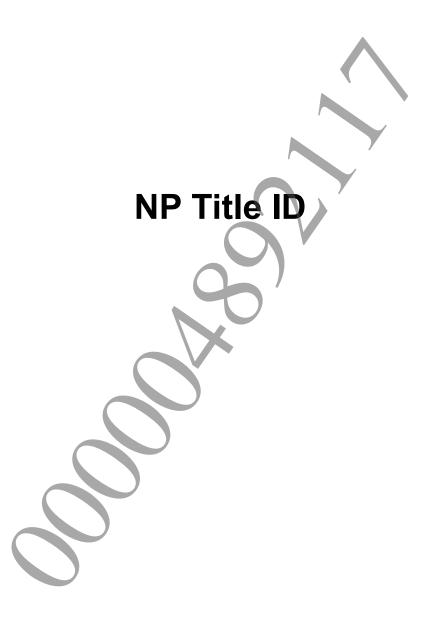
# **Description**

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

# **Examples**

### See Also

SceNpWebApiMemoryPoolStats



# sceNpWebApiSetNpTitleId

Set the NP Title ID and NP Title Secret

# **Definition**

# **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  $\bigstar$ **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**

# **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()





# SCE\_NP\_WEBAPI\_PUSH\_EVENT\_DATA\_TYPE\_LEN MAX

Maximum length for Push event data type

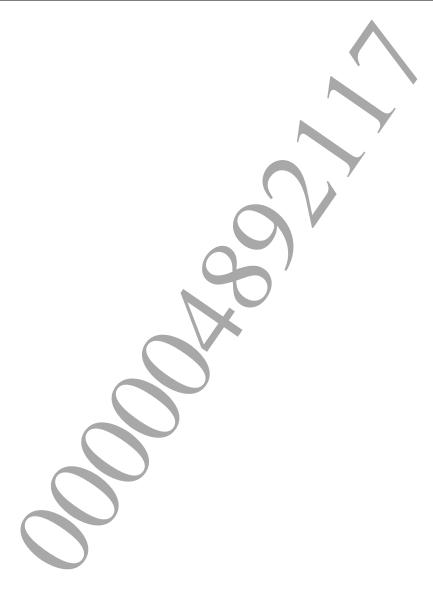
Value	(Number)	Description
SCE_NP_WEBAPI_PUSH_EVENT_DATA_TYPE_LEN_MAX	64	Maximum length for Push
		event data type



# SCE\_NP\_WEBAPI\_EXTD\_PUSH\_EVENT\_EXTD\_DAT A\_KEY\_LEN\_MAX

Maximum length for extended Push event data key

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



# SCE\_NP\_WEBAPI\_NP\_SERVICE\_NAME\_NONE

Constant indicating that there is no NP service name

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

Value	(Number)	Description
SCE NP WEBAPI ERROR	0x80552901	Insufficient memory
OUT OF MEMORY		, , , , , , , , , , , , , , , , , , , ,
SCE NP WEBAPI ERROR	0x80552902	Argument is invalid
INVALID_ARGUMENT		
SCE_NP_WEBAPI_ERROR_	0x80552903	Library context ID is invalid (will not return from the
INVALID_LIB_CONTEXT_ID		APIs in this library)
SCE NP WEBAPI ERROR	0x80552904	Library context could not be found (may return
LIB_CONTEXT_NOT_FOUND		when an invalid request ID has been assigned)
SCE NP WEBAPI ERROR	0x80552905	User context could not be found (may return when
USER CONTEXT NOT FOUND	0,100002	an invalid request ID has been assigned)
SCE NP WEBAPI ERROR	0x80552906	Request could not be found
REQUEST NOT FOUND	0.000332300	request could not be round
SCE NP WEBAPI ERROR	0x80552907	Function to be executed during a signed-in state was
NOT SIGNED IN	0.000332307	executed during a non-signed in state
SCE NP WEBAPI ERROR	0x80552908	Content parameter is invalid
INVALID CONTENT	0x00332900	Content parameter is invalid
PARAMETER		
SCE NP WEBAPI ERROR	0x80552909	Processing was aborted
ABORTED	0.000332303	Trocessing was aborted
SCE NP WEBAPI ERROR	0x8055290a	User context generated with the specified online ID
USER CONTEXT ALREADY	0.000002300	already exists (will not return from the APIs in this
EXIST		library)
SCE ND WEBART FRACE	0x8055290b	Push event filter was not found
SCE_NP_WEBAPI_ERROR_ PUSH EVENT FILTER	UX6033290D	rush event inter was not found
NOT FOUND	\ X	
SCE NP WEBAPI ERROR	0x8055290c	Push event callback was not found
PUSH EVENT CALLBACK	0.000002500	1 dsh event canback was not round
NOT FOUND		
SCE NP WEBAPI ERROR	0x8055290d	Handle was not found
HANDLE NOT FOUND	0.10000230u	Transact was not round
SCE NP WEBAPI ERROR	0x8055290e	Service Push event filter was not found
SERVICE PUSH EVENT		
FILTER NOT FOUND		
SCE_NP_WEBAPI_ERROR	0x8055290f	Service Push event callback was not found
SERVICE_PUSH_EVENT		
CALLBACK_NOT_FOUND		
SCE_NP_WEBAPI_ERROR_	0x80552910	Signed in user was not found
SIGNED_IN_USER_NOT_		
FOUND		
SCE_NP_WEBAPI_ERROR_	0x80552911	Library context is being used and termination
LIB_CONTEXT_BUSY		processing cannot be carried out (will not return
		from the APIs in this library)
SCE NP WEBAPI ERROR	0x80552912	User context is being used and cannot be deleted
USER_CONTEXT_BUSY		(will not return from the APIs in this library)
SCE NP WEBAPI ERROR	0x80552913	Request is being used and cannot be deleted (will not
REQUEST BUSY		return from the APIs in this library)
SCE NP WEBAPI ERROR	0x80552914	HTTP status code is invalid (will not return from the
INVALID HTTP STATUS CODE	0700002914	APIs in this library)
		111 15 111 11115 11111 at y j

Value	(Number)	Description
SCE NP WEBAPI ERROR	0x80552915	Specified an HTTP header for which specification is
PROHIBITED_HTTP_HEADER		prohibited (will not return from the APIs in this
		library)
SCE_NP_WEBAPI_ERROR_	0x80552916	Executed a function for which usage is prohibited.
PROHIBITED_FUNCTION_CALL		<u> </u>
SCE_NP_WEBAPI_ERROR_	0x80552917	The part of the multiple parts specified by the part
MULTIPART_PART_NOT_FOUND		index number does not exist
SCE_NP_WEBAPI_ERROR_	0x80552918	Specified parameter is too long
PARAMETER_TOO_LONG	0.00550010	A
SCE_NP_WEBAPI_ERROR_ HANDLE BUSY	0x80552919	Attempted to delete a handle while the handle was
MANDLE_BOST		being used (will not return from the APIs in this
CCE ND WEDADT EDDOD	0x8055291a	library)
SCE_NP_WEBAPI_ERROR_ LIB CONTEXT MAX	0x8055291a	Reached the maximum number of library contexts
	0x8055291b	(will not return from the APIs in this library)
SCE_NP_WEBAPI_ERROR_ USER CONTEXT MAX	0x80552910	Reached the maximum number of user contexts (will
SCE NP WEBAPI ERROR	0x8055291c	not return from the APIs in this library)
EXTD PUSH EVENT	0x8055291C	Extended Push event filter was not found
FILTER NOT FOUND		
SCE NP WEBAPI ERROR	0x8055291d	Extended Push event callback was not found
EXTD_PUSH_EVENT_		
CALLBACK_NOT_FOUND		
SCE_NP_WEBAPI_ERROR_	0x8055291e	Request is already sent
AFTER_SEND	0.00550.04	N. A. W. H. A.
SCE_NP_WEBAPI_ERROR_ NOT INITIALIZED	0x80552a01	Not initialized
SCE NP WEBAPI ERROR	0x80552a02	Already initialized
ALREADY INITIALIZED	000000000000000000000000000000000000000	Aircudy initialized
SCE NP WEBAPI ERROR	0x80552a03	SSL error: common name check of the server
SSL_ERR_CN_CHECK		certificate failed
SCE_NP_WEBAPI_ERROR_	0x80552a04	SSL error: no certificate for the root CA certificate
SSL_ERR_UNKNOWN_CA		that issued the server certificate
SCE_NP_WEBAPI_ERROR_	0x80552a05	SSL error: the validity period of the server certificate
SSL_ERR_NOT_AFTER_CHECK		has expired
SCE_NP_WEBAPI_ERROR_	0x80552a06	SSL error: the validity period of the server certificate
SSL_ERR_NOT_BEFORE_CHECK		has not started yet
SCE_NP_WEBAPI_ERROR_	0x80552a07	SSL error: format of the server certificate is invalid
SSL ERR INVALID CERT	2 2255 33	
SCE_NP_WEBAPI_ERROR_	0x80552a08	SSL error: internal error of the library
SSL_ERR_INTERNAL SCE NP WEBAPI ERROR	0x80552a09	DCNIS Mob ADI usa sa fuana this title is tomas a suit
TITLE IS BANNED	0x00552a09	PSN™ Web API usage from this title is temporarily
SCE NP WEBAPI ERROR	0x80552a0a	suspended nptitle.dat file not found
NP TITLE DAT NOT FOUND	UXOUSSZaUa	Tipune.dat me not round
SCE NP WEBAPI ERROR	0x80552a0b	Title ID does not match NP Title ID
TITLE_ID_NOT_MATCHED_	3.11.12.2.2.40.2	Check param.sfo and nptitle.dat or the value
TO_NP_TITLE_ID		specified for sceNpWebApiSetNpTitleId().
	I	1 - F

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