https://verifone.cloud/docs/application-development-kit-version-47/namespacejs
Updated: 17-Sep-2025

js Namespace Reference

Enumerations

enum JSTraceType { JST_HTTPGET, JST_HTTPPOST, JST_HTTPRESULT }

enum JSLogLevel { JSL_EMERGENCY =0, JSL_ALERT =1, JSL_CRITICAL =2, JSL_ERROR =3, JSL_WARNING =4, JSL_NOTICE =5, JSL_INFO =6, JSL_DEBUG =7 }

Functions

DllSpec bool	jsProcessor (void *data, const std::string &sourcecode, std::map< std::string, std::string > &arg, std::string &out, std::string &err, std::string &exitcode)
DllSpec bool	jsProcessorExt (void *data, const std::string &sourcecode, std::map< std::string, std::string > &arg, std::string &out, std::string &err, std::string &exitcode, vfihtml::ExtContext *context)
const DllSpec char *	jsProcVersion ()
DllSpec void	jsSetHttpProxy (const char *proxy)

const DllSpec char * *	jsGetHttpProxy ()
DllSpec void	jsSetConsole (void(*cb)(void *data, const char *text), void *data)
DllSpec void	jsGetConsole (void(*&cb)(void *, const char *), void *&data)
DllSpec void	jsSetNotify (int(*cb)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from), void *data)
DllSpec void	jsGetNotify (int(*&cb)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from), void *&data)
DllSpec void	jsSetNotifyAndWait (int(*cb)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from, const char *wait_id, std::string &result, int timeout_msec), void *data)
DllSpec void	jsGetNotifyAndWait (int(*&cb)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from, const char *wait_id, std::string &result, int timeout_msec), void *&data)
DllSpec void	jsSetTrace (void(*cb)(void *data, JSTraceType type, const std::string &app_id, std::map< std::string, std::string > &trace), void *data)
DllSpec void	jsGetTrace (void(*&cb)(void *data, JSTraceType type, const std::string &app_id, std::map< std::string, std::string > &trace), void *&data)
DllSpec void	jsSetLog (void(*cb)(void *data, const std::string &app_id, JSLogLevel log_level, const std::vector< std::string > &msg), void *data)
DllSpec void	jsGetLog (void(*&cb)(void *data, const std::string &app_id, JSLogLevel log_level, const std::vector< std::string > &msg), void *&data)

Enumeration Type Documentation

? JSLogLevel

enum JSLogLevel

Enumerator
JSL_EMERGENCY
JSL_ALERT
JSL_CRITICAL
JSL_ERROR
JSL_WARNING
JSL_NOTICE
JSL_INFO
JSL_DEBUG

? JSTraceType

enum JSTraceType

trace type

Enumerator	
JST_HTTPGET	trace of an HTTP GET request, data provided: "url"
JST_HTTPPOST	trace of an HTTP POST request, data provided: "url"
JST_HTTPRESULT	trace of the HTTP result, data provided: "status"

Function Documentation

? jsGetConsole()

DllSpec void js::jsGetConsole (void(*&)(void *, const char *) *cb*,
void *& *data*
)

get current setting of callback for console output

Parameters
[out] cb callback function pointer or NULL
[out] data data pointer provided to callback as first parameter

? jsGetHttpProxy()

const DllSpec char* js::jsGetHttpProxy ( )

Read current proxy setting

Returns
proxy setting.

? jsGetLog()

DllSpec void js::jsGetLog (void(*&)(void *data, const std::string &app_id, JSLogLevel log_level, const std::vector< std::string > &msg) *cb*,

void *& *data*

)

read logging callback

Parameters

[out] cb callback function pointer or NULL

[out] data data pointer provided to callback as first parameter

? jsGetNotify()

DllSpec void js::jsGetNotify (int(*&)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from) *cb*,

void *& *data*

)

get current setting of notification callback

Parameters

[out] cb callback function pointer or NULL

[out] data data pointer provided to callback as first parameter

? jsGetNotifyAndWait()

DllSpec void js::jsGetNotifyAndWait (int(*&)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from, const char *wait_id, std::string &result, int timeout_msec) *cb*,

void *& *data*

)

get current setting of notification callback

Parameters

[out] cb callback function pointer or NULL

[out] data data pointer provided to callback as first parameter

? jsGetTrace()

DllSpec void js::jsGetTrace (void(*&)(void *data, JSTraceType type, const std::string &app_id, std::map< std::string, std::string > &trace) *cb*, void *& *data*)

read HTTP trace callback

Parameters

[out] cb callback function pointer or NULL

[out] data data pointer provided to callback as first parameter

? jsProcessor()

DllSpec bool js::jsProcessor (void * *data*,
const std::string & *sourcecode*,
std::map< std::string, std::string > & *arg*,
std::string & *out*,
std::string & *err*,
std::string & *exitcode*
)

Process JavaScript script and return the data sent to stdout and stderr

Parameters

[in]	data	this parameter is unused, it is just there for compatibility with the ADKGUI scripting interface
[in]	sourcecode	string containing the Js source code
[in]	arg	key value map, can be accessed as table 'ARG' from within the JavaScript script
[out]	out	data sent to stdout
[out]	err	data sent to stderr
[out]	exitcode	parameter that was passed to exit()

Returns

true if processing was successful, false if not

? jsProcessorExt()

DllSpec bool js::jsProcessorExt (void * *data*,
const std::string & *sourcecode*,
std::map< std::string, std::string > & *arg*,

```
                                    std::string &                   out,
                                    std::string &                   err,
                                    std::string &                   exitcode,
                                    vfihtml::ExtContext *           context
                                )
```

Process JavaScript script and return the data sent to stdout and stderr. This is the variant with ADKGUI extensions support.

Parameters

[in]	data	this parameter is unused, it is just there for compatibility with the ADKGUI scripting interface
[in]	sourcecode	string containing the Js source code
[in]	arg	key value map, can be accessed as table 'ARG' from within the JavaScript script
[out]	out	data sent to stdout
[out]	err	data sent to stderr
[out]	exitcode	parameter that was passed to exit()
[in,out]	context	extended context, if NULL GUI extensions are disabled.

Returns

true if processing was successful, false if not

? jsProcVersion()

```
const DllSpec char* js::jsProcVersion (  )
```

Returns

version information

? jsSetConsole()

```
DllSpec void js::jsSetConsole ( void(*)(void *data, const char *text)  cb,
                                    void *                              data
                                )
```

set callback for console output

Parameters

[in] cb callback function pointer or NULL

[in] data data pointer provided to callback as first parameter

The callback takes the following parameters:

Parameters

[in] data data pointer

[in] text string to be sent to console

The console callback is global, only one callback may be set. Setting the callback function is not thread safe, i.e. appropriate actions have to be taken to protect against race conditions.

? jsSetHttpProxy()

DllSpec void js::jsSetHttpProxy (const char * *proxy*)

Set Http-Proxy

Parameters

[in] proxy Proxy setting, e.g. http://localhost:8888 or NULL

? jsSetLog()

DllSpec void js::jsSetLog (void(*)(void *data, const std::string &app_id, JSLogLevel log_level, const std::vector< std::string > &msg) *cb*, void * *data*)

install logging callback

Parameters

[in] cb callback function pointer or NULL

[in] data data pointer provided to callback as first parameter

The callback takes the following parameters:

Parameters

[in] data data pointer

[in] log_level logging level 0-7 (emergency,alert,critical,

[in] app_id application ID (as found in ARGV["cp_appId"])

[in] msg log message parameters in the order they were provided to log.info()/log.debug(),log.error()

The log callback is global, only one callback may be set. Setting the callback function is not thread safe, i.e. appropriate actions have to be taken to protect against race conditions.

? jsSetNotify()

DllSpec void js::jsSetNotify (int(*)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from) *cb*, void * *data*)

set callback for sending notifications

Parameters

[in] cb callback function pointer or NULL

[in] data data pointer provided to callback as first parameter

The callback takes the following parameters:

Parameters

[in] data	data pointer
[in] to	destination address
[in] notification_id	notification id
[in] json_param	string containing the JSON encoded parameters
[in] flags	optional flags
[in] from	optional source address, if NULL use default source address

Returns

0 in case of success or negative error code

The notification callback is global, only one callback may be set. Setting the callback function is not thread safe, i.e. appropriate actions have to be taken to protect against race conditions.

? jsSetNotifyAndWait()

DllSpec void js::jsSetNotifyAndWait (int(*)(void *data, const char *to, const char *notification_id, const char *json_param, unsigned flags, const char *from, const char *wait_id, std::string &result, int timeout_msec) *cb*, void * *data*)

set callback for sending notifications

Parameters

[in] cb callback function pointer or NULL

[in] data data pointer provided to callback as first parameter

The callback takes the following parameters:

Parameters

[in]	data	data pointer
[in]	to	destination address
[in]	notification_id	notification id
[in]	json_param	string containing the JSON encoded parameters
[in]	flags	optional flags
[in]	from	optional source address, if NULL use default source address
[in]	wait_id	wait for notification with this id
[out]	result	received notification
[in]	timeout_msec	timeout in milliseconds

Returns

0 in case of success or negative error code

The notification callback is global, only one callback may be set. Setting the callback function is not thread safe, i.e. appropriate actions have to be taken to protect against race conditions.

? jsSetTrace()

DllSpec void js::jsSetTrace (void(*)(void *data, JSTraceType type, const std::string &app_id, std::map< std::string, std::string > &trace) *cb*, void * *data*)

install HTTP trace callback

Parameters

[in] cb callback function pointer or NULL

[in] data data pointer provided to callback as first parameter

The callback takes the following parameters:

Parameters

[in] data data pointer
[in] type trace type
[in] app_id application ID (as found in ARGV["cp_appId"])
[in] trace key value map containing trace parameters

The trace callback is global, only one callback may be set. Setting the callback function is not thread safe, i.e. appropriate actions have to be taken to protect against race conditions.