

sdiprotocol.h File Reference

SDI protocol library interface.

[More...](#)

[Go to the source code of this file.](#)

Macros

#define [SDI_PROTOCOL_ERR_IO](#) -1
read, write or protocol error [More...](#)

#define [SDI_PROTOCOL_ERR_CONCURRENT_USE](#) -2
SDI protocol library currently used by another thread, maybe try later. [More...](#)

#define [SDI_PROTOCOL_ERR_CONNECT](#) -3
connection establishment to SDI server failed [More...](#)

#define [SDI_PROTOCOL_ERR_OVERFLOW](#) -4
response buffer too small, response has been dropped [More...](#)

#define [SDI_PROTOCOL_ERR_PARAM](#) -5
wrong formal parameter like NULL pointer [More...](#)

#define [SDI_PROTOCOL_ERR_OTHER](#) -6
any other problem like thread creation, memory allocation, etc. [More...](#)

#define [SDI_PROTOCOL_ERR_NO_RECEIVE](#) -7
command is suppressed, therefore, no [SDI_Receive\(\)](#) must be called afterwards [More...](#)

#define [SDI_PROTOCOL_INIT_OPT_TRACE_EMV_CBK](#) 0x01
trace using the EMV callback [More...](#)

#define [SDI_PROTOCOL_INIT_OPT_TRACE_SYSLOG](#) 0x02
use syslog instead of EMV callback for trace [More...](#)

#define [SDI_PROTOCOL_INIT_OPT_TRACE_ADK_LOG](#) 0x04

Typedefs

typedef [SDI_DATA_AVAILABLE_CALLBACK_TYPE](#) (void *context)

callback function type declaration for [SDI_SetDataAvailableCallback\(\) More...](#)

typedef [SDI_CALLBACK_TYPE](#)) (unsigned char *dataIn, unsigned short sizeIn, unsigned char *dataOut,
void(* unsigned short *sizeOut, void *context)

callback function type declaration for [SDI_SetSdiCallback\(\) More...](#)

typedef [CARD_DETECTED_CALLBACK_TYPE](#)) (unsigned char *dataIn, unsigned short sizeIn, void
void(* *context)

callback function type declaration for [SDI_SetCardDetectedCallback\(\) More...](#)

typedef [EOF_DETECTED_CALLBACK_TYPE](#)) (void *context)

void(*

callback function type declaration for [SDI_SetEOFDetectedCallback\(\) More...](#)

Functions

int [SDI_ProtocolInit](#) (unsigned options, const char *settings)
Configure the library, connection setup. [More...](#)

int [SDI_Send](#) (const unsigned char *data, int dataLength)
Send message to SDI server. [More...](#)

int [SDI_Receive](#) (unsigned char *responseBuffer, int responseBufferSize)
Receive response from SDI server. [More...](#)

int [SDI_SendReceive](#) (const unsigned char *data, int dataLength, unsigned char *responseBuffer,
int responseBufferSize)
Send message to SDI server and receive the response. [More...](#)

unsigned char * [SDI_GetStoredResponse](#) (unsigned *dataLen)

int [SDI_Disconnect](#) (void)
Shut down the connection to SDI server. [More...](#)

int [SDI_SetDataAvailableCallback](#) ([SDI_DATA_AVAILABLE_CALLBACK_TYPE](#) cb, void
*context)
Set a callback to be called if a SDI server response is pending. [More...](#)

int [SDI_SetSdiCallback](#) ([SDI_CALLBACK_TYPE](#) cb, void *context)
Set a callback function to be called when SDI server sends a callback message. [More...](#)

int [SDI_SetCardDetectedCallback](#) ([CARD_DETECTED_CALLBACK_TYPE](#) cb, void *context)
Set a callback function to be called when SDI server sends the Card Detected callback message.
[More...](#)

void [SDI_SendSysAbort](#) (void)
Send SYS ABORT command '20 02'. [More...](#)

void [SDI_SendExternalButton](#) (void)
Send External Button command '20 20' always with P2 = 1 (suppress response) [More...](#)

int [SDI_SetEOFDetectedCallback](#) ([EOF_DETECTED_CALLBACK_TYPE](#) cb, void *context)
Set a callback function to be called when the connection to SDI server is closed by EOF. [More...](#)

int [SDI_SetLoggingCallback](#) (void(*cb)(const char *, void *), void *context)
Set a callback function receiving SDI logging callbacks. [More...](#)

Detailed Description

SDI protocol library interface.

Typedef Documentation

? CARD_DETECTED_CALLBACK_TYPE

```
typedef void(* CARD_DETECTED_CALLBACK_TYPE) (unsigned char *dataIn, unsigned short sizeIn, void *context)
```

callback function type declaration for [SDI_SetCardDetectedCallback\(\)](#)

Parameters

- [in] dataIn Input data to send to the application: complete SDI Server callback message
- [in] sizeIn Length of input data
- [out] context Application context data pointer

? EOF_DETECTED_CALLBACK_TYPE

```
typedef void(* EOF_DETECTED_CALLBACK_TYPE) (void *context)
```

callback function type declaration for [SDI_SetEOFDetectedCallback\(\)](#)

Parameters

- [out] context Application context data pointer

? SDI_CALLBACK_TYPE

```
typedef void(* SDI_CALLBACK_TYPE) (unsigned char *dataIn, unsigned short sizeIn, unsigned char *dataOut, unsigned short *sizeOut, void *context)
```

callback function type declaration for [SDI_SetSdiCallback\(\)](#)

Parameters

- [in] dataIn Input data to send to the application: complete SDI Server callback message
- [in] sizeIn Length of input data
- [out] dataOut TLV stream of output data (callback response) to send back to the SDI Server. Might be NULL pointer for one-way callbacks

[in,out] sizeOut	Input: size of dataOut buffer; Output: size of output data. Must be set to zero for one-way callbacks
[out] context	Application context data pointer

? SDI_DATA_AVAILABLE_CALLBACK_TYPE

typedef void(* SDI_DATA_AVAILABLE_CALLBACK_TYPE) (void *context)

callback function type declaration for [SDI_SetDataAvailableCallback\(\)](#)

Function Documentation

? SDI_Disconnect()

int SDI_Disconnect (void)

Shut down the connection to SDI server.

Might be called if the connection shall be provided to another process, for shutdown or in case of unrecoverable communication errors.

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_GetStoredResponse()

unsigned char* SDI_GetStoredResponse (unsigned * *dataLen*)

Access stored response in case of SDI_PROTOCOL_ERR_OVERFLOW

To be called by the same thread that got the error code. Data will be no more available if meanwhile another thread had the same problem.

Parameters

[out] dataLen length of stored response data

Returns

buffer containing stored response CAUTION: To be released with free by caller

? SDI_ProtocolInit()

```
int SDI_ProtocolInit ( unsigned    options,  
                      const char * settings  
                      )
```

Configure the library, connection setup.

See Programmer's Guide for connection handling

Parameters

- [in] options bit mask for configuration flags [SDI Protocol Initialization Options](#)
- [in] settings configuration parameters JSON string

Settings (all optional/conditional): {"server": {"host":<string>, "port":<integer>, "ca":<string|string[]>}}

Parameters

- settings.server.host host IP address or host name
- settings.server.port TCP/IP port
- settings.server.ca For TLS connections, path to Certification Authority PEM file (concatenated) or list of PEM file paths
- settings.server.socket path to Unix Domain Socket (will make other parameters ignored)
- settings.server.serial path to serial device (will make other parameters ignored)

Default values: {"server": {"host": "127.0.0.1", "port": 12000}}

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_Receive()

```
int SDI_Receive ( unsigned char * responseBuffer,  
                 int             responseBufferSize  
                 )
```

Receive response from SDI server.

For instance uses infinite timeout

Parameters

- [in] responseBuffer buffer to take the payload of SDI server response
- [in] responseBufferSize length of responseBuffer

Returns

length of response in case of success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_Send()

```
int SDI_Send ( const unsigned char * data,
               int                  dataLength
               )
```

Send message to SDI server.

Will establish connection if necessary

Parameters

[in] *data* data buffer with payload
[in] *dataLength* length of payload data

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_SendExternalButton()

```
void SDI_SendExternalButton ( void )
```

Send External Button command '20 20' always with P2 = 1 (suppress response)

This command informs the SDI-Server in Headless Mode that the external button is pressed

? SDI_SendReceive()

```
int SDI_SendReceive ( const unsigned char * data,
                     int                  dataLength,
                     unsigned char *      responseBuffer,
                     int                  responseBufferSize
                     )
```

Send message to SDI server and receive the response.

Combines SDI_Send and SDI_Receive

Parameters

[in] *data* data buffer with payload
[in] *dataLength* length of payload data
[in] *responseBuffer* buffer to take the payload of SDI server response
[in] *responseBufferSize* length of responseBuffer

Returns

length of response in case of success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_SendSysAbort()

void SDI_SendSysAbort (void)

Send SYS ABORT command '20 02'.

This command is available even when SendReceive is in progress. This is useful if a dialog during a callback needs to be aborted.

? SDI_SetCardDetectedCallback()

```
int SDI_SetCardDetectedCallback ( CARD\_DETECTED\_CALLBACK\_TYPE cb,  
                                void * context  
                                )
```

Set a callback function to be called when SDI server sends the Card Detected callback message.

The callback function is called in the receiver thread and should not block.

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_SetDataAvailableCallback()

```
int SDI_SetDataAvailableCallback ( SDI\_DATA\_AVAILABLE\_CALLBACK\_TYPE cb,  
                                void * context  
                                )
```

Set a callback to be called if a SDI server response is pending.

Set a callback that will be invoked the callback when SDIresponse data is pending after [SDI_Send\(\)](#).

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_SetEOFDetectedCallback()

```
int SDI_SetEOFDetectedCallback ( EOF\_DETECTED\_CALLBACK\_TYPE cb,
```

```

        void *
    )
        context

```

Set a callback function to be called when the connection to SDI server is closed by EOF.

The callback function is called in the receiver thread before termination and should not block.

Do not call any API function from inside callback EOF_DETECTED_CALLBACK_TYPE.

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)

? SDI_SetLoggingCallback()

```

int SDI_SetLoggingCallback ( void(*) (const char *, void *) cb,
                             void *
                             context
                             )

```

Set a callback function receiving SDI logging callbacks.

Supported by VR1 only

Parameters

[in] cb callback function f(const char *message*, void context)

[in] context pointer given as parameter context to invocation of parameter cb

? SDI_SetSdiCallback()

```

int SDI_SetSdiCallback ( SDI\_CALLBACK\_TYPE cb,
                          void *
                          context
                          )

```

Set a callback function to be called when SDI server sends a callback message.

The SDI Callback is called in the receiver thread and should not block.

Returns

0 for success else SDI protocol error code, see [SDI Protocol Error Codes](#)