# ADK-SDI-Client Programmers Guide

### Audience

The SDI Server compat client *libsdiclient* is for existing projects migrating from direct ADK use to SDI server approach.

### Organization

This guide relies on the ADK-SDI Programmers Guide. For the EMV part see ADK-EMV Contact Programmers Guide and ADK-EMV Contactless Programmers Guide. Other details can be found in ADK-TEC Programmers Guide, ADK-MSR Programmers Guide and ADK-NFC Programmers Guide.

## Introduction

libsdiclient provides an API based approach for interfacing the SDI server. It was designed to reuse the EMV-ADK API, its client libraries, libEMV_CT_client and libEMV_CTLS_Client for serialisation. The EMV-ADK client libraries call the shared EMV link libraries that are configured to use SDI mode via libsdiprotocol.

Non-EMV-ADK extensions have been added using the same infrastructure since the SDI server was not able to handle multiple connections at time of design. The original library name was libsdiemvclient which is still available but deprecated. With the extension the new name was introduced: libsdiclient.

For more sample code please refer to EMV Test App from which most examples in this document originate from. You can find its source in VDE sample (ADK full package) in directory doc/vdedemo/EmvTestApp.

## Getting Started

### Dependencies

libsdiclient depends on

- libTLV_Util
- libEMV_CT_Client
- libEMV_CTLS_Client

The static clients call the EMV-ADK link libraries which results in more dependencies to these shared libraries

- libEMV_CT_Link
- libEMV_CTLS_Link
- libTLV_Util
- libsdiprotocol
- libipc
- libipctls
- liblog (Android: libvfilog)

```
# Makefile example based on ADK make rules# Should be added to applications or libraries using libsdiclient and libsdiprotocolLIBS += -lsdiclient -lEMV_CT_Client -lEMV_CTLS_ClientLIBS += -lsdiprotocol -lEMV_CT_Link -lEMV_CTLS_Link -lTLV_Util
```

## Programming

Header files of libsdiclient are located in import/<platform>/include/sdiclient. Therefore they are included this way:

```
#include <sdiclient/sdi_emv.h> // for migration of EMV-ADK API#include <sdiclient/sdi_data.h> // provides
SDI_fetchTxnTags#include <sdiclient/sdi_if.h> // SDI client interface for non-EMV// #define
SDI_NFC_NAMESPACE_ONLY uncomment if applicable#include <sdiclient/sdi_nfc.h> // compatiblity to NFC_Interface.h
```

### EMV-API

By including sdi_emv.h EMV-ADK API functions are available as SDI counterpart. Their prefix changes from **EMV_** to **SDI_**, e.g. EMV_SetTermData becomes SDI_SetTermData. There are some differences to note:

- On responses with transaction results the SDI Server will hold back card holder sensitive data unless the card's PAN appears to be whitelisted. SDI_CT_ContinueOffline() and SDI_CT_ContinueOnline() have an additional ouput parameter of type EMV_SDI_CT_TRANSRES_STRUCT containing SDI Server added obfuscated data. The same applies to SDI_CTLS_ContinueOffline() and SDI_CTLS_ContinueOnline() with EMV_SDI_CTLS_TRANSRES_STRUCT.
- SDI_fetchTxnTags() replaces both, EMV_CT_fetchTxnTags() and EMV_CTLS_fetchTxnTags(). The SDI server will switch to appropriate technology and apply its sensitive data filters.
- SDI_CT_SmartISO() and SDI_CTLS_SmartISO() as well as SDI_CT_SER_SmartISO() and SDI_CTLS_SER_SmartISO() are blocked by SDI server and can't be used. For custom direct chip card communication an SDI plugin should be used.

Excerpt from EMV TestApp supporting both, direct ADK and SDI compat client:

```
EMV_ADK_INFO CTLS_fetchTxnTags(unsigned long options, unsigned long* requestedTags, unsigned short
noOfRequestedTags, unsigned char* tlvBuffer, unsigned short bufferLength, unsigned short* tlvDataLength ){ if
(SdiMode) return SDI_fetchTxnTags(options, requestedTags, noOfRequestedTags, tlvBuffer, bufferLength,
tlvDataLength); else return EMV_CTLS_fetchTxnTags(options, requestedTags, noOfRequestedTags, tlvBuffer,
bufferLength, tlvDataLength);} EMV_ADK_INFO CTLS_EndTransaction(unsigned long options){ if(SdiMode) return
SDI_CTLS_EndTransaction(options); else return EMV_CTLS_EndTransaction(options);}
```

### SDI-API

Other SDI-API is provided by sdi_if.h. Most SDI commands, especially those with fewer parameters are provided by libsdi::SDI. In most cases the functions just return a success flag and in case of failure libsdi::SdiBase::getSdiSw12(), libsdi::SdiBase::getAdditionalResultValue() and libsdi::SdiBase::getClientError() can be called for fine-grained error handling. More complex command invocation is performed by instantiation of other libsdi::SdiBase subclasses and calling their setters for providing the input data, then starting command execution and finally invoking getters for accessing the result data from the SDI server response. These are

- libsdi::CardDetection for Technology Selection (SDI class 23)
- libsdi::SdiCrypt for cryptographic and some data interface functions (SDI classes 70 and 29)
- libsdi::ManualEntry for manual card data input (SDI command 21-02)
- libsdi::PED for PIN input functions (SDI class 22)
- libsdi::Dialog for display control of External PIN pad (SDI class 24)
- libsdi::SdiCmd Generic command for command and tags not (yet) supported by this library

Example:

```
libsdi::SdiCrypt crypt;crypt.open("myProvider");crypt.setInitialVector(myIntialVector);crypt.encrypt(myInputData
, myCipherData);crypt.close();
```

> In general libsdiclient does not provide deprecated SDI commands. If other commands are missing because not yet provided here, the SDI_SendReceive() method from libsdiprotocol can be directly called but the input data has to be set up by your application and the output data has to be parsed. There is a **generic** class libsdi::SdiCmd supporting convenient getters and setters for TLV based SDI commands.

### NFC-API

The header file sdi_nfc.h has a compiler switch SDI_NFC_NAMESPACE_ONLY that deactivates the NFC-ADK complient function aliases for the case of name conflicts. In case of client side error or negative SDI response the return value will be EMB_APP_COMM_ERROR or VAS_COMM_ERR. In this case the negative SDI SW12 value or the client side error can be read with libsdi::getNfcSW12() and libsdi::getNfcClientError().

### VOS3 API

For compatibility reasons these (VOS2-)interfaces are provided on VOS3. Parts of this functionality require VOS3-CARDS plugin.

> If the plugin is installed SDI is no more P2PE domain.

- ADK-EMV (ADK-EMV Contact Programmers Guide, ADK-EMV Contactless Programmers Guide)
- ADK-TEC (ADK-TEC Programmers Guide)
- ADK-MSR (ADK-MSR Programmers Guide)
- ADK-NFC (ADK-NFC Programmers Guide)
- libsdi::SDI::setManualPAN() replaces ADK-SEC secPutTransactionData() for the use case that PAN from manual input is required for ISO-0 PIN block.

### Connection Handling

The connection to the SDI Server is implicitly established when required and will not be closed unless libsdiprotocol's SDI_Disconnect() is called. The default destination is 127.0.0.1::12000 but other IP addresses or connection types can be configured with SDI_ProtocolInit() **before** a connection is set-up or SDI_Disconnect() can be used to apply a configuration change.

Example 1 - Simple TCP/IP connection:

```
// {"server":{"host":"127.0.0.1","port":12000}}vfiipc::JSObject server, config;server("host") =
"127.0.0.1";server("port") = 12000;config("server") = server;return SDI_ProtocolInit(0, config.dump().c_str());
```

Example 2 - TLS connection:

```
// {"server":{"ca":["CA_1.pem","CA_2.pem"],"host":"vfi-terminal","port":12000}}vfiipc::JSObject server,
config;server("host") = "vfi-terminal";server("port") = 12000;server("ca")[0] = "CA_1.pem";server("ca")[1] =
"CA_2.pem";config("server") = server;return SDI_ProtocolInit(0, config.dump().c_str());
```

Example 3 - Unix Domain Socket connection:

```
// {"server":{"socket":"/tmp/sdi.socket"}}vfiipc::JSObject server, config;server("socket") =
"/tmp/sdi.socket"config("server") = server;return SDI_ProtocolInit(0, config.dump().c_str());
```

Example 4 - Serial connection:

```
// {"server":{"serial":"/dev/ttyACM0"}}vfiipc::JSObject server, config;server("serial") =
"/dev/ttyACM0"config("server") = server;return SDI_ProtocolInit(0, config.dump().c_str());
```

## Sample Code

### Card Detection

The card detection demo code starts all three SDI card detection modes for illustrating the differences.

```
bool startCardDetection(libsdi::CardDetection& cardDetection, unsigned char tech, int variant){
libsdi::CardDetection::DetectionMode mode = libsdi::CardDetection::DETECTION_MODE_BLOCKING; if (variant ==
SDI_BLOCKING) { mode = libsdi::CardDetection::DETECTION_MODE_BLOCKING;
SDI_SetDataAvailableCallback(sdiDataAvailableCallback, NULL); } else if (variant == SDI_POLLING) { mode =
libsdi::CardDetection::DETECTION_MODE_POLLING; } else if (variant == SDI_CALLBACK) { mode =
libsdi::CardDetection::DETECTION_MODE_CALLBACK; cardDetection.setCallback(sdiCardDetectionCallback, NULL); }
cardDetection.setDetectionMode(mode); std::vector<unsigned char> ucOptions(16, 0); ucOptions[4] =
EMV_CT_TRY_PPS|EMV_CT_DETECT_WRONG_ATR; cardDetection.setTecStartOptions(ucOptions); return
cardDetection.startSelection(tech, 30 == 0);}
```

### Chip Card Processing

After card detection results in CTS_CHIP (contact chip card detected) the card should be powered and in this example the PSE is read. As written above card communications is blocked by SDI server but the EMV TestApp has a plugin that sends a Select PSE command for command id SDI_PLUGIN_SELECT_PSE.

```
unsigned char atr[40];unsigned long atrLen;unsigned char status = isSdiMode() ? SDI_CT_SmartReset(EMV_CT_TRY_PPS | EMV_CT_DETECT_WRONG_ATR, atr, &atrLen); : EMV_CT_SmartReset(EMV_CT_TRY_PPS | EMV_CT_DETECT_WRONG_ATR, atr, &atrLen); if (status == EMV_ADK_SMART_STATUS_OK){ unsigned short rspLen; unsigned char rsp[261]; APP_TRACE_HEX(atr, atrLen, "ATR: \n"); unsigned char cmd[] = "\x00\xA4\x04\x00\x0E" "1PAY.SYS.DDF01"; APP_TRACE_HEX(cmd, sizeof cmd, "CMD: "); if (isSdiMode()) { std::vector<unsigned char> inOut; libsdi::SDI sdi; libsdi::SDI_SW12 sw12 = sdi.pluginCommand(SDI_PLUGIN_ID_EMVTESTAPP, SDI_PLUGIN_SELECT_PSE, inOut, inOut); if (sw12 == libsdi::SDI_SW12_SUCCESS) { if (inOut.size() > 2) { struct BTLVNode root; struct BTLVNode* rapdu; vBTLVInit(&root, NULL); if (iBTLVImport(&root, inOut.data() + 2, inOut.size() - 2, NULL, NULL) == 0 && (rapdu = pxBTLVFindTag(&root, "F0/DF01")) != NULL) { if (rapdu->uSize > sizeof rsp) { APP_TRACE("R-APDU too long: ", rapdu->uSize); status = EMV_ADK_SMART_STATUS_OVERFLOW; } else { rspLen = (unsigned short) rapdu->uSize; memcpy(rsp, rapdu->pucData, rapdu->uSize); } } vBTLVClear(&root); } } else { APP_TRACE("SDI plugin result: %04x" , sw12); status = (sw12 & 0xFF00) == 0x9000 ? (unsigned char) sw12 : EMV_ADK_SMART_STATUS_EXCHG_ERR; } } else { status = EMV_CT_SmartISO(0, (unsigned short) sizeof cmd, cmd, &rspLen, rsp, sizeof rsp); } APP_TRACE( "CT_SmartISO returns %d", status);}
```

**SDI command mapping table**

| CLA | INS | P1 | P2 | SDI name | libsdiclient API |
|---|---|---|---|---|---|
| 20 | 00 | 00 | 00 | Sync/Turn On | *deprecated* |
| 20 | 01 | 00 | 00 | Turn Off | *deprecated* |
| 20 | 02 | 00 | 00 | Sys Abort (with response) | SDI_SendSysAbort() |
| 20 | 02 | 00 | 01 | Sys Abort (without response) | *not available* |
| 20 | 03 | 00 | 00 | Sys Get Config | *not available*, instead use libsdi::SDI::getProperty() |
| 20 | 04 | 00 | 00 | Sys Get Status | libsdi::SDI::getDateTime(), libsdi::SDI::getLanguage(), libsdi::SDI::getCardDataEntryDeactivation(), libsdi::SDI::getCardDataEntryMode() |
| 20 | 05 | 00 | 00 | Sys Selftest | *deprecated* |

| | | | | | |
|---|---|---|---|---|---|
| 20 | 09 | 00 | 00 | Sys Set Status | libsdi::SDI::setDateTime(), libsdi::SDI::setLanguage(), libsdi::SDI::setCardDataEntryDeactivation(), libsdi::SDI::setCardDataEntryMode() |
| 20 | 10 | 00 | 00 | Sys Get KSN | *not available* |
| 20 | 11 | 00 | 00 | Sys Auth | *not available* |
| 20 | 13 | 00 | 00 | Set Idle Text | libsdi::SDI::setIdleText() |
| 20 | 14 | 00 | 00 | Software Upload Start | libsdi::SDI::sysUploadStart() |
| 20 | 15 | 00 | 00 | Software Upload Transfer | libsdi::SDI::sysUploadTransfer() |
| 20 | 16 | 00 | 00 | Software Upload Finalize | libsdi::SDI::sysUploadFinalize() |
| 20 | 17 | 00 | 00 | Shutdown | libsdi::SDI::sysShutdown() |
| 20 | 17 | 00 | 01 | Reboot | libsdi::SDI::sysReboot() |
| 20 | 17 | 00 | 02 | Sleep | libsdi::SDI::sysSleep() |
| 20 | 17 | 00 | 03 | Hibernate | libsdi::SDI::sysHybernate() |
| 20 | 18 | 00 | 00 | Show MAC Desktop | libsdi::SDI::showMacDesktop() |
| 20 | 19 | 00 | 00 | Set Property (Integer) | libsdi::SDI::setProperty(vfisysinfo::SYSPropertyInt, int, bool) |

| | | | | | |
|---|---|---|---|---|---|
| 20 | 19 | 00 | 01 | Set Property (String) | libsdi::SDI::setProperty(vfisysinfo::SYSPropertyString, const std::string &, bool) |
| 20 | 1A | 00 | 00 | Get Property (Integer) | libsdi::SDI::getProperty(vfisysinfo::SYSPropertyInt, bool)<br><br>libsdi::SDI::getProperty(vfisysinfo::SYSPropertyInt, int&, bool) |
| 20 | 1A | 00 | 01 | Get Property (String) | libsdi::SDI::getProperty(vfisysinfo::SYSPropertyString, bool)<br><br>libsdi::SDI::getProperty(vfisysinfo::SYSPropertyString, std::string&, bool) |
| 20 | 18 | 00 | 00 | Install Sponsor Cert | libsdi::SDI::installSponsorCert() |
| 20 | 1C | 00 | 00 | Get SDI version info | libsdi::SDI::getVersionInfo() |
| 20 | 1D | 00 | 00 | Check For Update | libsdi::SDI::checkForUpdate() |
| 20 | 1E | 00 | 00 | Get last install error | libsdi::SDI::getLastInstallError() |
| 20 | 20 | 00 | 00 | External Button | libsdi::SDI::externalButton() |
| 20 | 21 | 00 | 00 | Enable EPP | libsdi::SDI::enableEpp() |
| 20 | 21 | 00 | 01 | Disable EPP | libsdi::SDI::disableEpp() |
| 20 | 22 | 00 | 00 | Factory Reset | libsdi::SDI::factoryReset() |
| 20 | 23 | 00 | 00 | Read Keystore Certificate | libsdi::SDI::readCertificate() |

| | | | | | |
|---|---|---|---|---|---|
| 20 | 24 | 00 | 00 | Force Currency Abbreviation | libsdi::SDI::setUseCurrencyAbbreviation() |
| | | | | | |
| 21 | 00 | 00 | 00 | MSR Enable | *deprecated* |
| 21 | 01 | 00 | 00 | MSR Read | libsdi::CardDetection::startMsrRead() |
| 21 | 02 | 00 | 00 | MSR Card Data Entry | libsdi::ManualEntry::start() |
| 21 | 03 | 00 | 00 | MSR Set Options | libsdi::CardDetection::msrSetOptions() |
| 21 | 04 | 00 | 00 | MSR Switch LEDs | libsdi::SDI::msrSwitchLeds() |
| 21 | 05 | 00 | 00 | MSR Set Sensitivity | libsdi::SDI::msrSetSensitivty() |
| | | | | | |
| 22 | 00 | 00 | 00 | PED Enable | *not available* |
| 22 | 00 | 00 | 01 | PED Disable | *not available* |
| 22 | 01 | 00 | 00 | PED Get PIN | libsdi::PED::startPinInput() |
| 22 | 02 | 00 | 00 | PED Set PIN Timeout | libsdi::PED::setDefaultTimeout() |
| 22 | 03 | 00 | 00 | PED start PIN entry | libsdi::PED::startPinEntry() |
| 22 | 04 | 00 | 00 | PED poll PIN entry | libsdi::PED::pollPinEntry() |
| 22 | 05 | 00 | 00 | PED stop PIN entry | libsdi::PED::stopPinEntry() |
| 22 | 06 | 00 | 00 | PED Set PIN Input Parameter | libsdi::PED::sendPinInputParameters() |
| | | | | | |
| 23 | 00 | 00 | 00 | Card Detect Enable | *not available* |
| 23 | 01 | 00 | NN | Card Detection | libsdi::CardDetection::startSelection() |

| | | | | | |
|---|---|---|---|---|---|
| 23 | 02 | 00 | 00 | Wait Card Removal | libsdi::SDI::waitCardRemoval()<br><br>libsdi::SDI::sendWaitCardRemoval() |
| 23 | 03 | 00 | 00 | Start Card Detection | libsdi::CardDetection::startSelection() |
| 23 | 04 | 00 | 00 | Poll Card Detection | libsdi::CardDetection::pollTechnology() |
| 23 | 05 | 00 | 00 | Stop Card Detection | libsdi::CardDetection::stopSelection() |
| 23 | 06 | 00 | 00 | Add Technology | libsdi::CardDetection::addTechnology() |
| 23 | 07 | 00 | 00 | Remove Technology | libsdi::CardDetection::removeTechnology() |
| 23 | 08 | 00 | 00 | Provide Callback Response | *implicit call* |
| | | | | | |
| 24 | 00 | 00 | 00 | Display Enable | *deprecated* |
| 24 | 01 | 00 | 00 | Display Text | *deprecated* |
| 24 | 02 | 00 | 00 | Display Text with Confirmation | *deprecated* |
| 24 | 03 | 00 | 00 | Handle Display | libsdi::Dialog::display() |
| 24 | 04 | 00 | 00 | Handle Secure Input | libsdi::Dialog::secureInput() |
| 24 | 05 | 00 | 00 | Handle Menu | libsdi::Dialog::menu() |
| 24 | 06 | 00 | 00 | Handle Card Request Display | libsdi::Dialog::requestCard() |
| 24 | 07 | 00 | 00 | Display Idle Screen | libsdi::Dialog::idleScreen() |

| | | | | | |
|---|---|---|---|---|---|
| 24 | 08 | 00 | 00 | Handle Signature Capture | libsdi::Dialog::captureSignature() |
| 24 | 09 | 00 | 00 | Activate LEDs | libsdi::Dialog::showLedArea() |
| 24 | 0A | 00 | 00 | Handle HTML Dialog | libsdi::Dialog::htmlDialog() |
| 24 | 0B | 00 | 00 | Get Async Display Result | libsdi::Dialog::getAsyncResult() |
| | | | | | |
| 25 | 00 | 00 | 00 | Set Printer Property (Integer) | libsdi::SDI::setPrinterProperty(vfiprt::PrtPropertyInt, int) |
| 25 | 00 | 00 | 01 | Set Printer Property (String) | libsdi::SDI::setPrinterProperty(vfiprt::PrtPropertyString, const std::string&) |
| 25 | 01 | 00 | 00 | Get Printer Property (Integer) | libsdi::SDI::getPrinterProperty(vfiprt::PrtPropertyInt, int&) |
| 25 | 01 | 00 | 01 | Get Printer Property (String) | libsdi::SDI::getPrinterProperty(vfiprt::PrtPropertyString, std::string&) |
| 25 | 02 | 00 | 00 | Print HTML | libsdi::SDI::printHtml() |
| 25 | 04 | 00 | 00 | Print Bitmap | libsdi::SDI::printBitmap() |
| | | | | | |
| 26 | | 00 | | Plugin Interface | libsdi::SDI::pluginCommand() |
| | | | | | |
| 28 | 00 | 00 | 00 | VCL Registart SRED Request | libsdi::SDI::vclRegistartSRED() |
| 28 | 01 | 00 | 00 | VCL Status Request | libsdi::SDI::vclStatusRequest() |

| | | | | | |
|---|---|---|---|---|---|
| 28 | 02 | 00 | 00 | VCL Advance DDK | libsdi::SDI::vclAdvanceDDK() |
| 28 | 03 | 00 | 00 | VCL Request eParms | libsdi::SDI::vclRequestEParms() |
| 28 | 06 | 00 | 00 | VCL Request Diag Query | libsdi::SDI::vclGetDiagnosticData() |
| 28 | 07 | 00 | 00 | VCL Request Derive Query | libsdi::SDI::vclGetKeyDerivationMode() |
| 28 | 08 | 00 | 00 | VCL Override Message Query | libsdi::SDI::vclOverrideMessageQuery() |
| 28 | 09 | 00 | 00 | VCL_KSN_Request | libsdi::SDI::vclKsnRequest() |
| 28 | 0A | 00 | 00 | VCL KMAILIN Request | libsdi::SDI::vclKmailinRequest() |
| | | | | | |
| 29 | 00 | 00 | 00 | getEncData | libsdi::SdiCrypt::getEncData() |
| 29 | 01 | 00 | 00 | getEncMsgData | libsdi::SdiCrypt::getEncMsgData() |
| 29 | 02 | 00 | 00 | fetchTrxTags | SDI_fetchTxnTags() |
| 29 | 03 | 00 | 00 | clearDataStore | libsdi::SDI::clearDataStore() |
| 29 | 04 | 00 | 00 | getMsgSignature | libsdi::SdiCrypt::getMsgSignature() |
| 29 | 05 | 00 | 00 | performValidationChecks | libsdi::SDI::performValidationChecks() |
| 29 | 06 | 00 | 00 | getValidationInfo | libsdi::SDI::getValidationInfo() |
| 29 | 07 | 00 | 00 | getEncTrxData | libsdi::SdiCrypt::getEncTrxData() |
| 29 | 08 | 00 | 00 | setEncTrxData | libsdi::SdiCrypt::setEncTrxData() |

| | | | | | |
|---|---|---|---|---|---|
| 29 | 09 | 00 | 00 | endEncTrxData | libsdi::SdiCrypt::endEncTrxData() |
| 29 | 0A | 00 | 00 | getEmvInfo | *not available*, use EMV API |
| | | | | | |
| 31 | 00 | 00 | 00 | NFC Ping | libsdi::NFC_Ping(), NFC_Ping() |
| 31 | 01 | 00 | 00 | NFC Pass Through Open | libsdi::NFC_PT_Open(), NFC_PT_Open() |
| 31 | 02 | 00 | 00 | NFC Pass Through Close | libsdi::NFC_PT_Close(), NFC_PT_Close() |
| 31 | 03 | 00 | 00 | NFC Pass Through Field On | libsdi::NFC_PT_FieldOn(), NFC_PT_FieldOn() |
| 31 | 04 | 00 | 00 | NFC Pass Through Field Off | libsdi::NFC_PT_FieldOff(), NFC_PT_FieldOff() |
| 31 | 05 | 00 | 00 | NFC Pass Through Field Polling | libsdi::NFC_PT_Polling(), NFC_PT_Polling() |
| 31 | 07 | 00 | 00 | NFC Pass Through Card Activation | libsdi::NFC_PT_Activation(), NFC_PT_Activation() |
| 31 | 08 | 00 | 00 | NFC Pass Through RxTx | *blocked*: libsdi::NFC_PT_TxRx(), NFC_PT_TxRx() |
| 31 | 09 | 00 | 00 | NFC Pass Through Ftech Baud | libsdi::NFC_PT_FtechBaud(), NFC_PT_FtechBaud() |
| 31 | 0A | 00 | 00 | NFC Mifare Authenticate | libsdi::NFC_Mifare_Authenticate(), NFC_Mifare_Authenticate() |
| 31 | 0B | 00 | 00 | NFC Mifare Read | libsdi::NFC_Mifare_Read(), NFC_Mifare_Read() |

| | | | | | |
|---|---|---|---|---|---|
| 31 | 0C | 00 | 00 | NFC Mifare Write | libsdi::NFC_Mifare_Write(), NFC_Mifare_Write() |
| 31 | 0D | 00 | 00 | NFC Mifare Increment | libsdi::NFC_Mifare_Increment(), NFC_Mifare_Increment() |
| 31 | 0E | 00 | 00 | NFC Mifare Decrement | libsdi::NFC_Mifare_Decrement(), NFC_Mifare_Decrement() |
| 31 | 0F | 00 | 00 | NFC Felica Exchange | libsdi::NFC_Felica_Exchange(), NFC_Felica_Exchange() |
| 31 | 10 | 00 | 00 | NFC Client Init | libsdi::NFC_Client_Init(), NFC_Client_Init() |
| 31 | 11 | 00 | 00 | NFC Get Version | libsdi::NFC_Get_Version(), NFC_Get_Version() |
| 31 | 13 | 00 | 00 | NFC Pass Through Field Polling Full | libsdi::NFC_PT_PollingFull(), NFC_PT_PollingFull() |
| 31 | 14 | 00 | 00 | NFC Felica Polling | libsdi::NFC_Felica_Polling(), NFC_Felica_Polling() |
| 31 | 15 | 00 | 00 | NFC Mifare Increment Only | libsdi::NFC_Mifare_Increment_Only(), NFC_Mifare_Increment_Only() |
| 31 | 16 | 00 | 00 | NFC Mifare Decrement Only | libsdi::NFC_Mifare_Decrement_Only(), NFC_Mifare_Decrement_Only() |
| 31 | 17 | 00 | 00 | NFC Mifare Transfer | libsdi::NFC_Mifare_Transfer(), NFC_Mifare_Transfer() |

| | | | | | |
|---|---|---|---|---|---|
| 31 | 18 | 00 | 00 | NFC Mifare Restore | libsdi::NFC_Mifare_Restore(), NFC_Mifare_Restore() |
| 31 | 1C | 00 | 00 | NFC APDU Exchange | *blocked*, libsdi::NFC_APDU_Exchange(), NFC_APDU_Exchange() |
| | | | | | |
| 32 | 00 | 00 | 00 | VAS Terminal Config | libsdi::NFC_Terminal_Config(), NFC_Terminal_Config() |
| 32 | 01 | 00 | 00 | VAS Terminal Read Config | libsdi::NFC_TERMINAL_ReadConfig(), NFC_TERMINAL_ReadConfig() |
| 32 | 02 | 00 | 00 | VAS Read Config | libsdi::NFC_VAS_ReadConfig(), NFC_VAS_ReadConfig() |
| 32 | 03 | 00 | 00 | VAS Activate | libsdi::NFC_VAS_Activate(), NFC_VAS_Activate() |
| - | - | - | - | VAS Cancel | *not available*, libsdi::NFC_VAS_Cancel, NFC_VAS_Cancel() |
| 32 | 05 | 00 | 00 | VAS Update Config | libsdi::NFC_VAS_UpdateConfig(), NFC_VAS_UpdateConfig() |
| 32 | 06 | 00 | 00 | VAS Cancel Config | libsdi::NFC_VAS_CancelConfig(), NFC_VAS_CancelConfig() |

| | | | | | |
|---|---|---|---|---|---|
| 32 | 07 | 00 | 00 | VAS Preload | libsdi::NFC_VAS_PreLoad(), NFC_VAS_PreLoad() |
| 32 | 08 | 00 | 00 | VAS Cancel Preload | libsdi::NFC_VAS_CancelPreLoad(), NFC_VAS_CancelPreLoad() |
| 32 | 09 | 00 | 00 | VAS Decrypt | libsdi::NFC_VAS_Decrypt(), NFC_VAS_Decrypt() |
| | | | | | |
| 39 | 00 | 00 | 00 | EMV_CT_Init_Framewerk | SDI_CT_Init_Framework() |
| 39 | 00 | 00 | 01 | EMV_CT_Exit_Framewerk | SDI_CT_Exit_Framework() |
| 39 | 01 | 00 | 00 | EMV_CT_SetTermData | SDI_CT_SetTermData() |
| 39 | 01 | 00 | 01 | EMV_CT_GetTermData | SDI_CT_GetTermData() |
| 39 | 02 | 00 | 00 | EMV_CT_SetAppliData | SDI_CT_SetAppliData() |
| 39 | 02 | 00 | 01 | EMV_CT_GetAppliData | SDI_CT_GetAppliData() |
| 39 | 03 | 00 | 00 | EMV_CT_StoreCAPKey | SDI_CT_StoreCAPKey() |
| 39 | 03 | 00 | 01 | EMV_CT_ReadCAPKeys | SDI_CT_ReadCAPKeys() |
| | | | | EMV_CT_GetCAPKeyInfo | SDI_CT_GetCAPKeyInfo() |
| 39 | 04 | 00 | 00 | EMV_CT_ApplyConfiguration | SDI_CT_ApplyConfiguration() |
| 39 | 05 | 00 | 00 | EMV_CT_FRAMEWORK_GetVersion | SDI_CTLS_FRAMEWORK_GetVersion() |
| - | - | - | - | EMV_CT_CLIENT_GetVersion | SDI_CT_CLIENT_GetVersion() |

| | | | | | |
|---|---|---|---|---|---|
| 39 | 06 | 00 | 00 | EMV_CT_MapVirtualTerminal | SDI_CT_MapVirtualTerminal() |
| 39 | 10 | 00 | 00 | EMV_CT_StartTransaction | SDI_CT_StartTransaction() |
| 39 | 11 | 00 | 00 | EMV_CT_ContinueOffline | SDI_CT_ContinueOffline() |
| 39 | 12 | 00 | 00 | EMV_CT_ContinueOnline | SDI_CT_ContinueOnline() |
| 39 | 13 | 00 | 00 | EMV_CT_updateTxnTags | SDI_CT_updateTxnTags() |
| 39 | 14 | 00 | 00 | EMV_CT_fetchTxnTags | *blocked*, use SDI_fetchTxnTags() |
| 39 | 15 | 00 | 00 | EMV_CT_EndTransaction | SDI_CT_EndTransaction(), libsdi::SDI::clearDataStore() |
| 39 | 16 | 00 | 00 | EMV_CT_GetCandidateData | SDI_CT_GetCandidateData() |
| 39 | 17 | 00 | 00 | EMV_CT_CheckSupportedAID | SDI_CT_CheckSupportedAID() |
| 39 | 18 | 00 | 00 | EMV_CT_fetchTxnDOL | *not available* |
| | | | | | |
| 40 | 00 | 00 | 00 | EMV_CTLS_Init_Framewerk | SDI_CTLS_Init_Framework() |
| 40 | 00 | 00 | 01 | EMV_CTLS_Exit_Framewerk | SDI_CTLS_Exit_Framework() |
| | | | | EMV_CTLS_Exit_Framewerk_extended | SDI_CTLS_Exit_Framework_extended() |
| 40 | 01 | 00 | 00 | EMV_CTLS_SetTermData | SDI_CTLS_SetTermData() |
| 40 | 01 | 00 | 01 | EMV_CTLS_GetTermData | SDI_CTLS_GetTermData() |

| | | | | | |
|---|---|---|---|---|---|
| 40 | 02 | 00 | 00 | EMV_CTLS_SetAppliDataSchemeSpecific | SDI_CTLS_SetAppliDataSchemeSpecific() |
| 40 | 02 | 00 | 01 | EMV_CTLS_GetAppliDataSchemeSpecific | SDI_CTLS_GetAppliDataSchemeSpecific() |
| 40 | 03 | 00 | 00 | EMV_CTLS_StoreCAPKey | SDI_CTLS_StoreCAPKey() |
| 40 | 03 | 00 | 01 | EMV_CTLS_ReadCAPKeys | SDI_CTLS_ReadCAPKeys() |
| | | | | EMV_CTLS_GetCAPKeyInfo | SDI_CTLS_GetCAPKeyInfo() |
| 40 | 04 | 00 | 00 | EMV_CTLS_ApplyConfiguration | SDI_CTLS_ApplyConfiguration() |
| 40 | 05 | 00 | 00 | EMV_CTLS_FRAMEWORK_GetVersion | SDI_CTLS_FRAMEWORK_GetVersion() |
| - | - | - | - | EMV_CTLS_CLIENT_GetVersion | SDI_CTLS_CLIENT_GetVersion() |
| 40 | 06 | 00 | 00 | EMV_CTLS_MapVirtualTerminal | SDI_CTLS_MapVirtualTerminal() |
| 40 | 10 | 00 | 00 | EMV_CTLS_SetupTransaction | SDI_CTLS_SetupTransaction() |
| 40 | 11 | 00 | 00 | EMV_CTLS_ContinueOffline | SDI_CTLS_ContinueOffline() |
| | | | | EMV_CTLS_ContinueOfflineExt | SDI_CTLS_ContinueOfflineExt() |
| 40 | 12 | 00 | 00 | EMV_CTLS_ContinueOnline | SDI_CTLS_ContinueOnline() |
| 40 | 14 | 00 | 00 | EMV_CTLS_fetchTxnTags | *blocked*, use SDI_fetchTxnTags() |
| 40 | 15 | 00 | 00 | EMV_CTLS_EndTransaction | SDI_CTLS_EndTransaction(), libsdi::SDI::clearDataStore() |
| 40 | 16 | 00 | 00 | EMV_CTLS_GetCandidateData | SDI_CTLS_GetCandidateData() |

| | | | | | |
|---|---|---|---|---|---|
| 40 | 18 | 00 | 00 | EMV_CTLS_fetchTxnDOL | *not available* |
| 40 | 20 | 00 | 00 | EMV_CTLS_Break | SDI_CTLS_Break() |
| 41 | 00 | 00 | 00 | Smart Card Enable | *not available* |
| 41 | 01 | 00 | 00 | EMV_CT_SmartDetect | SDI_CT_SmartDetect() |
| 41 | 02 | 00 | 00 | EMV_CT_SmartReset | SDI_CT_SmartReset() |
| 41 | 03 | 00 | 00 | EMV_CT_SmartISO | *blocked* |
| 41 | 04 | 00 | 00 | EMV_CT_SmartPowerOff | SDI_CT_SmartPowerOff() |
| 41 | 05 | 00 | 00 | EMV_CT_Send_PIN_Offline | SDI_CT_Send_PIN_Offline() |
| 41 | 09 | 00 | 00 | EMV_CT_LED | SDI_CT_LED() |
| 42 | 02 | 00 | 00 | EMV_CTLS_SmartReset | SDI_CTLS_SmartReset() |
| 42 | 03 | 00 | 00 | EMV_CTLS_SmartISO | *blocked* |
| 42 | 04 | 00 | 00 | EMV_CTLS_SmartPowerOff | SDI_CTLS_SmartPowerOff() |
| 42 | 0A | 00 | 00 | EMV_CTLS_CardRemoval | SDI_CTLS_CardRemoval() |
| 43 | 03 | 00 | 00 | EMV_CTLS_LED | SDI_CTLS_LED() |
| 43 | 04 | 00 | 00 | EMV_CTLS_LED_SetMode | SDI_CTLS_LED_SetMode() |
| 70 | 00 | 00 | 00 | Crypto Open | libsdi::SdiCrypt::open() |
| 70 | 01 | 00 | 00 | Crypto Close | libsdi::SdiCrypt::close() |

| | | | | | |
|---|---|---|---|---|---|
| 70 | 02 | 00 | 00 | Crypto Encrypt | libsdi::SdiCrypt::encrypt() |
| 70 | 03 | 00 | 00 | Crypto Decrypt | libsdi::SdiCrypt::decrypt() |
| 70 | 04 | 00 | 00 | Crypto Sign | libsdi::SdiCrypt::sign() |
| 70 | 05 | 00 | 00 | Crypto Verify | libsdi::SdiCrypt::verify() |
| 70 | 06 | 00 | 00 | Crypto Updatekey | libsdi::SdiCrypt::updateKey() |
| 70 | 07 | 00 | 00 | Crypto Set Key Set | libsdi::SdiCrypt::setKeySetId() |
| 70 | 08 | 00 | 00 | Crypto Get encrypted Pin | libsdi::SdiCrypt::getEncryptedPin() |
| 70 | 09 | 00 | 00 | Crypto Get Key Inventory | libsdi::SdiCrypt::getKeyInventory() |
| 70 | 0A | 00 | 00 | Crypto Get Key Data | libsdi::SdiCrypt::getKeyData() |
| 70 | 0B | 00 | 00 | Crypto Get Status | libsdi::SdiCrypt::getStatus() |
| 70 | 0C | 00 | 00 | Crypto Get Versions | libsdi::SdiCrypt::getVersions() |
| 72 | 00 | 00 | 00 | Read Certificate | *not available* |
| 72 | 01 | 00 | 00 | RSA Private Key Calculate | *not available* |

### Callback Functions

Callbacks sent by SDI Server do not end a running command, instead the client API has to wait for the response. Meanwhile some types callback require a response which blocks the command execution. Therefore libsdiprotocol and libsdiclient support setting callback functions that are called by separate threads. This chapter lists up the SDI callbacks, the way a callback function can be registered and in which context it is invoked.

The Data Available callback is no SDI Server callback but sent once the SDI Server response is available on a command sent out by SDI_Send(). It is listed here for completeness.

For callback types that do not expect a response the output parameter *sizeOut* of SDI_CALLBACK_TYPE has to be set to zero by the callback function.

| SW1 | SW2 | SDI reference | Registration | Context |
|---|---|---|---|---|
| 91 | 01 | EMV Callback (contact) | SDI_CT_Init_Framework() | SDI_CT_StartTransaction() and SDI_CT_ContinueOffline() in callback mode |
| 91 | 01 | CTLS Query Callback | SDI_CTLS_Init_Framework() | libsdi::CardDetection::startSelection() with DETECTION_MODE_BLOCKING |
| 9F | | Status Callbacks | SDI_SetSdiCallback() | libsdi::PED::startPinInput(), libsdi::ManualEntry::start() |
| 9B | 01 | Control Callback | SDI_SetSdiCallback() | libsdi::SDI::performValidationChecks() |
| 9E | 01 | Notify Callback | SDI_CTLS_Init_Framework() | libsdi::CardDetection::startSelection() and out of transaction, e.g. LED blinking |
| 9E | 02 | Card Detected Callback | libsdi::CardDetection::setCallback() | libsdi::CardDetection::startSelection() with DETECTION_MODE_CALLBACK |
| 9E | 03 | CTLS Query Callback | SDI_CTLS_Init_Framework() | libsdi::CardDetection::startSelection() other than DETECTION_MODE_BLOCKING |
| 9E | 04 | Card Removal Callback | SDI_SetSdiCallback() | hybrid card reader only: libsdi::CardDetection::startSelection() (MSR only), libsdi::CardDetection::startMsrRead() |

| | | | | |
|---|---|---|---|---|
| 9D | | Navigator Callbacks | SDI_SetSdiCallback() | libsdi::PED::startPinInput(), libsdi::ManualEntry::start() |
| SW1 | SW2 | Data Available | SDI_SetDataAvailableCallback() | libsdi::PED::startPinInput(), libsdi::ManualEntry::start(), libsdi::CardDetection::startSelection() with DETECTION_MODE_BLOCKING |