



# **IDTech Windows SDK Guide for SpectrumPro**

**#80140504-001**

**Rev. A**



## Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;SpectrumPro;SDK;Win	04/06/2016
A	Updated UWP Implementation Information	9/22/2016

# Contents

<b>1</b>	<b>IDTech Windows SDK Reference Guide for SpectrumPro</b>	<b>1</b>
<b>2</b>	<b>Important Security Notice</b>	<b>3</b>
2.1	Applicability . . . . .	3
2.2	What Does PA-DSS Mean to You? . . . . .	3
2.3	Third Party Applications . . . . .	4
2.4	PA-DSS Guidelines . . . . .	4
2.5	More Information . . . . .	9
<b>3</b>	<b>Main Transaction Commands</b>	<b>11</b>
3.1	EMV Methods . . . . .	11
3.2	MSR Methods . . . . .	12
<b>4</b>	<b>Connecting to SpectrumPro</b>	<b>13</b>
4.1	Connect with USB: . . . . .	13
4.2	Connect with Serial Interface (COM) . . . . .	13
<b>5</b>	<b>Core Implementation: WinForms</b>	<b>14</b>
5.1	Integrating with IDTechSDK.dll . . . . .	14
5.2	Import the necessary libraries . . . . .	14
5.3	Add using statements to utilize library . . . . .	15
5.4	Implement the callback function . . . . .	15
5.5	Initialize SpectrumPro: . . . . .	16
5.6	Sample Project Tutorial . . . . .	17
5.6.1	Step 1: Create New Project . . . . .	17
5.6.2	Step 2: Import Libraries . . . . .	18
5.6.3	Step 3: Design Interface . . . . .	18
5.6.4	Step 4: Configure the project file . . . . .	19
5.6.5	Step 5: Configure callback to receive important SDK data (messages,log info and transaction results) . . . . .	22
<b>6</b>	<b>Core Implementation: UWP</b>	<b>27</b>
6.1	Integrating with IDTechSDK_UWP.dll . . . . .	27

6.2	Import the Necessary Libraries . . . . .	27
6.3	Add Device Details to the Application's Manifest . . . . .	28
6.4	Add Using Statements to Utilize Library . . . . .	30
6.5	Implement the Callback Function . . . . .	30
6.6	Initialize SpectrumPro . . . . .	31
6.7	Sample Project Tutorial . . . . .	32
6.7.1	Step 1: Create New Project . . . . .	32
6.7.2	Step 2: Import Libraries . . . . .	33
6.7.3	Step 3: Add Device Details . . . . .	33
6.7.4	Step 4: Design Interface . . . . .	33
6.7.5	Step 5: Configure the Project File . . . . .	34
6.7.6	Step 6: Configure Callback . . . . .	38
<b>7</b>	<b>LCD Foreign Language Mapping Table</b>	<b>43</b>
<b>8</b>	<b>Error Code Reference</b>	<b>45</b>
<b>9</b>	<b>Enumeration Reference</b>	<b>50</b>
<b>10</b>	<b>EMV Callback</b>	<b>52</b>
<b>11</b>	<b>EMV Tag Reference</b>	<b>53</b>
<b>12</b>	<b>Namespace Index</b>	<b>61</b>
12.1	Packages . . . . .	61
<b>13</b>	<b>Class Index</b>	<b>62</b>
13.1	Class List . . . . .	62
<b>14</b>	<b>Namespace Documentation</b>	<b>63</b>
14.1	IDTechSDK Namespace Reference . . . . .	63
14.1.1	Enumeration Type Documentation . . . . .	64
14.1.1.1	CTLS_APPLICATION . . . . .	64
14.1.1.2	EMV_CALLBACK_TYPE . . . . .	64
14.1.1.3	EMV_LCD_DISPLAY_MODE . . . . .	65
14.1.1.4	EMV_PIN_MODE . . . . .	65
14.1.1.5	EMV_RESULT_CODE . . . . .	65
<b>15</b>	<b>Class Documentation</b>	<b>66</b>
15.1	IDTechSDK.EMV_Callback Class Reference . . . . .	66
15.1.1	Detailed Description . . . . .	66
15.1.2	Member Data Documentation . . . . .	66
15.1.2.1	callbackType . . . . .	66
15.1.2.2	language . . . . .	66

15.1.2.3	lcd_backlightTimeout	67
15.1.2.4	lcd_displayMode	67
15.1.2.5	lcd_entryTimeout	67
15.1.2.6	lcd_entryTimeoutMinor	67
15.1.2.7	lcd_messages	67
15.1.2.8	maskEntry	67
15.1.2.9	msr_displayMessage	68
15.1.2.10	msr_swipeTimeout	68
15.1.2.11	pin_entryInterval	68
15.1.2.12	pin_entryStartTimeout	68
15.1.2.13	pin_KSN	68
15.1.2.14	pin_pinMode	68
15.1.2.15	pin_truncatedPAN	68
15.2	IDTechSDK.IDT_SpectrumPro Class Reference	68
15.2.1	Detailed Description	70
15.2.2	Member Function Documentation	70
15.2.2.1	config_getModelNumber(ref string response)	70
15.2.2.2	config_getSerialNumber(ref string response)	70
15.2.2.3	device_controlUserInterface(byte[] values)	70
15.2.2.4	device_getFirmwareVersion(ref string response)	71
15.2.2.5	device_getResponseCodeString(RETURN_CODE eCode)	71
15.2.2.6	device_getSpectrumProKSN(int type, ref byte[] KSN)	81
15.2.2.7	device_pollCardReader(ref byte[] status)	82
15.2.2.8	device_rebootDevice()	83
15.2.2.9	device_sendDataCommand(string cmd, bool calcLRC, ref byte[] response)	83
15.2.2.10	device_sendMacDataCommand(byte taskID, byte[] functionID, byte[] data, bool macData, ref byte[] response)	83
15.2.2.11	device_setSpectrumProBDK(string BDK)	84
15.2.2.12	device_setUID(string UID, int keyType, ref SpectrumInfo info)	84
15.2.2.13	device_startRKI()	84
15.2.2.14	device_updateFirmware(byte[] firmwareData, string firmwareName, int encryptionType, byte[] keyBlob)	85
15.2.2.15	emv_allowFallback(bool allow)	85
15.2.2.16	emv_authenticateTransaction(byte[] updatedTLV)	86
15.2.2.17	emv_autoAuthenticate(bool authenticate)	86
15.2.2.18	emv_autoAuthenticate(bool authenticate, byte[] tags)	86
15.2.2.19	emv_callbackResponseLCD(EMV_LCD_DISPLAY_MODE type, byte selection)	86
15.2.2.20	emv_callbackResponseMSR(byte[] MSR)	87
15.2.2.21	emv_callbackResponsePIN(EMV_PIN_MODE type, byte[] KSN, byte[] PIN)	87
15.2.2.22	emv_cancelTransaction()	88

15.2.2.23 emv_clearTransactionLog()	88
15.2.2.24 emv_completeTransaction(bool commError, byte[] authCode, byte[] iad, byte[] tlvScripts, byte[] tlv)	88
15.2.2.25 emv_getEMVConfigurationCheckValue(ref string response)	88
15.2.2.26 emv_getEMVKernelCheckValue(ref string response)	89
15.2.2.27 emv_getEMVKernelVersion(ref string response)	89
15.2.2.28 emv_getTerminalID(ref string response)	89
15.2.2.29 emv_getTerminalMajorConfiguration(ref int configuration)	89
15.2.2.30 emv_removeAllApplicationData()	90
15.2.2.31 emv_removeAllCAPK()	90
15.2.2.32 emv_removeAllCRL()	90
15.2.2.33 emv_removeApplicationData(byte[] AID)	90
15.2.2.34 emv_removeCAPK(byte[] capk)	91
15.2.2.35 emv_removeCRL(byte[] crlList)	91
15.2.2.36 emv_removeTerminalData()	91
15.2.2.37 emv_retrieveAIDList(ref byte[][] response)	91
15.2.2.38 emv_retrieveApplicationData(byte[] AID, ref byte[] tlv)	92
15.2.2.39 emv_retrieveCAPK(byte[] capk, ref byte[] key)	92
15.2.2.40 emv_retrieveCAPKList(ref byte[] keys)	92
15.2.2.41 emv_retrieveCRLList(ref byte[] list)	93
15.2.2.42 emv_retrieveTerminalData(ref byte[] tlv)	93
15.2.2.43 emv_retrieveTransactionResult(byte[] tags, ref IDTTransactionData tlv)	93
15.2.2.44 emv_setApplicationData(byte[] name, byte[] tlv)	94
15.2.2.45 emv_setCAPK(byte[] key)	94
15.2.2.46 emv_setCRL(byte[] list)	94
15.2.2.47 emv_setTerminalData(byte[] tlv)	95
15.2.2.48 emv_setTerminalID(string terminalID)	95
15.2.2.49 emv_setTerminalMajorConfiguration(int configuration)	95
15.2.2.50 emv_startTransaction(double amount, double amtOther, int exponent, int type, int timeout, byte[] tags, bool forceOnline)	96
15.2.2.51 icc_getICCReaderStatus(ref byte status)	96
15.2.2.52 icc_getICCStatus(ref byte[] status, ref byte[] atr)	96
15.2.2.53 icc_powerOffICC()	97
15.2.2.54 icc_powerOnICC(ref byte[] ATR)	97
15.2.2.55 msr_cancelMSRSwipe()	98
15.2.2.56 msr_clearMSRData()	98
15.2.2.57 msr_getMSRData(ref IDTTransactionData card)	98
15.2.2.58 msr_startMSRSwipe(int timeout)	98
15.2.2.59 pin_cancelPINEntry()	98
15.2.2.60 pin_getPIN(int mode, int PANSource, string iccPAN, int startTimeout, int entryTimeout, string language)	99

15.2.2.61	pin_passThroughMode(bool enable)	99
15.2.2.62	SDK_Version()	99
15.2.2.63	setCallback(CallBack my_Callback)	100
15.2.2.64	setCallback(IntPtr my_Callback, SynchronizationContext context)	100
15.2.2.65	useSerialPort(int port, bool isSRED)	100
15.2.2.66	useSerialPort(int port, int baud, bool isSRED)	100
15.2.2.67	useUSB()	101
15.2.3	Property Documentation	101
15.2.3.1	SharedController	101
15.3	IDTechSDK.IDTTTransactionData Class Reference	101
15.3.1	Detailed Description	102
15.3.2	Member Function Documentation	102
15.3.2.1	decryptData(bool isAES, byte[] ksn, string BDK, byte[] encodedData)	102
15.3.2.2	decryptData(bool isAES, byte[] ksn, string BDK1, string BDK2, byte[] encodedData)	103
15.3.3	Member Data Documentation	103
15.3.3.1	captureCardType	103
15.3.3.2	captureEncryptType	103
15.3.3.3	captureEncryptTypeEMV	103
15.3.3.4	ctlsApplication	103
15.3.3.5	device_RSN	104
15.3.3.6	emv_appErrorFn	104
15.3.3.7	emv_appErrorState	104
15.3.3.8	emv_clearingRecord	104
15.3.3.9	emv_encryptedTags	104
15.3.3.10	emv_ESC	104
15.3.3.11	emv_hasAdvise	104
15.3.3.12	emv_hasReversal	104
15.3.3.13	emv_maskedTags	104
15.3.3.14	emv_resultCode	104
15.3.3.15	emv_RF_State	104
15.3.3.16	emv_rfStateCode	104
15.3.3.17	emv_transaction_Error_Code	105
15.3.3.18	emv_unencryptedTags	105
15.3.3.19	Event	105
15.3.3.20	iccPresent	105
15.3.3.21	isCTLS	105
15.3.3.22	mac	105
15.3.3.23	macKSN	105
15.3.3.24	msr_captureEncodeStatus	105
15.3.3.25	msr_cardType	105

15.3.3.26 msr_encTrack1 . . . . .	106
15.3.3.27 msr_encTrack2 . . . . .	106
15.3.3.28 msr_encTrack3 . . . . .	106
15.3.3.29 msr_errorCode . . . . .	106
15.3.3.30 msr_extendedField . . . . .	106
15.3.3.31 msr_hashTrack1 . . . . .	106
15.3.3.32 msr_hashTrack2 . . . . .	106
15.3.3.33 msr_hashTrack3 . . . . .	106
15.3.3.34 msr_keyVariantType . . . . .	106
15.3.3.35 msr_KSN . . . . .	106
15.3.3.36 msr_rawData . . . . .	107
15.3.3.37 msr_sessionID . . . . .	107
15.3.3.38 msr_track1 . . . . .	107
15.3.3.39 msr_track1Length . . . . .	107
15.3.3.40 msr_track2 . . . . .	107
15.3.3.41 msr_track2Length . . . . .	107
15.3.3.42 msr_track3 . . . . .	107
15.3.3.43 msr_track3Length . . . . .	107
15.3.3.44 Notification . . . . .	107
15.3.3.45 pin_KeyEntry . . . . .	107
15.3.3.46 pin_KSN . . . . .	107
15.3.3.47 pin_pinblock . . . . .	108
15.3.3.48 SW1 . . . . .	108
15.3.3.49 SW2 . . . . .	108



## Chapter 1

# IDTech Windows SDK Reference Guide for SpectrumPro



IDTechSDK.dll and IDTechSDK\_UWP.dll are Windows dynamic link libraries that will be provided by IDTech as the main interface between Windows Form (WinForms) and Universal Windows Platform (UWP) applications, respectively, the SpectrumPro and payment processing solutions.

The purpose of this document is to describe the requirements of the API as well as the interface definitions and requirements needed for a WinForms or UWP application wishing to deploy with the payment application.

- [Connecting to SpectrumPro](#)
- [Core Implementation: WinForms](#)
- [Core Implementation: UWP](#)
- [Important Security Notice](#)

- [Main Transaction Commands](#)
- [EMV Callback](#)
- [EMV Tag Reference](#)
- [Enumeration Reference](#)
- [Error Code Reference](#)
- [LCD Foreign Language Mapping Table](#)

## Chapter 2

# Important Security Notice

The Payment Card Industry Payment Application Data Security Standard (PCI PA-DSS) is comprised of fourteen requirements that support the Payment Card Industry Data Security Standard (PCI DSS). The PCI Security Standards Council (PCI SSC), which was founded by the major card brands in June 2005, set these requirements in order to protect cardholder payment information. The standards set by the council are enforced by the payment card companies who established the Council: American Express, Discover Financial Services, JCB International, MasterCard Worldwide, and Visa, Inc.

PCI PA-DSS is an evolution of Visas Payment Application Best Practices (PABP), which was based on the Visa Cardholder Information Security Program (CISP). In addition to Visa CISP, PCI DSS combines American Express Data Security Operating Policy (DSOP), Discover Networks Information Security and Compliance (DISC), and MasterCard Site Data Protection (SDP) into a single comprehensive set of security standards. The transition to PCI PA-DSS was announced in April 2008. In early October 2008, PCI PA-DSS Version 1.2 was released to align with the PCI DSS Version 1.2, which was released on October 1, 2008. On January 1, 2011, PCI PA-DSS Version 2.0 was released. This extends the PCI DSS Version 1.2, which was released on October 1, 2008 and is effective as of January 1, 2011.

## 2.1 Applicability

The PCI PA-DSS applies to any payment application that stores, processes, or transmits cardholder data as part of authorization or settlement, unless the application would fall under the merchant's PCI DSS validation. It is important to note that PA-DSS validated payment applications alone do not guarantee PCI DSS compliance for the merchant. The validated payment application must be implemented in a PCI DSS compliant environment. If your application runs on Windows XP, you are required to turn off Windows XP System Restore Points.

## 2.2 What Does PA-DSS Mean to You?

The following table provides opening points to cover in any discussion with merchants on data storage.

	Data Element	Storage Permitted	Protection Required	PCI DSS Req. 3, 4
Cardholder Data	Primary Account Number	Yes	Yes	Yes
	Cardholder Name <sup>1</sup>	Yes	Yes <sup>1</sup>	No
	Service Code <sup>1</sup>	Yes	Yes <sup>1</sup>	No
	Expiration Date <sup>1</sup>	Yes	Yes <sup>1</sup>	No
Sensitive Authentication Data <sup>2</sup>	Full Magnetic Stripe Data <sup>3</sup>	No	N/A	N/A
	CAV2/CID/CVC2/CVV2	No	N/A	N/A
	PIN/PIN Block	No	N/A	N/A

<sup>1</sup> These data elements must be protected if stored in conjunction with the PAN. This protection should be per PCI DSS requirements for general protection of the cardholder environment. Additionally, other legislation (for example, related to consumer personal data protection, privacy, identity theft, or data security) may require specific protection of this data, or proper disclosure of a company's practices if consumer-related personal data is being collected during the course of business. PCI DSS, however, does not apply if PANs are not stored, processed, or transmitted.

<sup>2</sup> Do not store sensitive authentication data after authorization (even if encrypted).

<sup>3</sup> Full track data from the magnetic stripe, magnetic-stripe image on the chip, or elsewhere.

## 2.3 Third Party Applications

The end-to-end transaction process, beginning with entry into the third party application until the response from the payment engine is returned, must meet the same level of compliance. In order to claim the third party application is end-to-end compliant, the application would need to be submitted to a QSA for a full PA-DSS audit.

The end user and/or P.O.S. developer can integrate and be compliant in the processing portion of a payment transaction. A brief review (given below) of the PA-DSS environmental variables that impact the end user merchant can help the end user merchant obtain and/or maintain PA-DSS compliance. Environmental variables that could prevent passing an audit include without limitation issues involving a secure network connection(s), end user setup location security, users, logging and assigned rights. Remove all testing configurations, samples, and data prior to going into production on your application.

## 2.4 PA-DSS Guidelines

The following PA-DSS Guidelines are being provided by IDTech as a convenience to its customers. Customers should not rely on these PA-DSS Guidelines, but should instead always refer to the most recent PCI DSS Program Guide published by PCI SSC.

### 1. Sensitive Data Storage Guidelines

Do not retain full magnetic stripe, card validation code or value (CAV2, CID, CVC2, CVV2), or PIN block data.

1.1 Do not store sensitive authentication data after authorization (even if encrypted): Sensitive authentication data includes the data as cited in the following Requirements 1.1.1 through 1.1.3. PCI Data Security Standard Requirement 3.2

Note: By prohibiting storage of sensitive authentication data after authorization, the assumption is that the transaction has completed the authorization process and the customer has received the final transaction approval. After authorization has completed, this sensitive authentication data cannot be stored.

1.1.1 After authorization, do not store the full contents of any track from the magnetic stripe (located on the back

of a card, contained in a chip, or elsewhere). This data is alternatively called full track, track, track 1, track 2, and magnetic-stripe data.

In the normal course of business, the following data elements from the magnetic stripe may need to be retained:

- The accountholders name,
- Primary account number (PAN),
- Expiration date, and
- Service code
- To minimize risk, store only those data elements needed for business.

Note: See PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms for additional information. PCI Data Security Standard Requirement 3.2.1

1.1.2 After authorization, do not store the card-validation value or code (three-digit or four-digit number printed on the front or back of a payment card) used to verify card-not-present transactions. Note: See PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms for additional information. PCI Data Security Standard Requirement 3.2.2

1.1.3 After authorization, do not store the personal identification number (PIN) or the encrypted PIN block.

Note: See PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms for additional information. PCI Data Security Standard Requirement 3.2.3

1.1.4 Securely delete any magnetic stripe data, card validation values or codes, and PINs or PIN block data stored by previous versions of the payment application, in accordance with industry-accepted standards for secure deletion, as defined, for example by the list of approved products maintained by the National Security Agency, or by other State or National standards or regulations. PCI Data Security Standard Requirement 3.2

Note: This requirement only applies if previous versions of the payment application stored sensitive authentication data.

1.1.5 Securely delete any sensitive authentication data (pre-authorization data) used for debugging or troubleshooting purposes from log files, debugging files, and other data sources received from customers, to ensure that magnetic stripe data, card validation codes or values, and PINs or PIN block data are not stored on software vendor systems. These data sources must be collected in limited amounts and only when necessary to resolve a problem, encrypted while stored, and deleted immediately after use. PCI Data Security Standard Requirement 3.2

## 2. Protect stored cardholder data

2.1 Software vendor must provide guidance to customers regarding purging of cardholder data after expiration of customer-defined retention period. PCI Data Security Standard Requirement 3.1

2.2 Mask PAN when displayed (the first six and last four digits are the maximum number of digits to be displayed).

Notes:

- This requirement does not apply to those employees and other parties with a legitimate business need to see full PAN;
- This requirement does not supersede stricter requirements in place for displays of cardholder data for example, for point-of-sale (POS) receipts. PCI Data Security Standard Requirement 3.3

2.3 Render PAN, at a minimum, unreadable anywhere it is stored, (including data on portable digital media, backup media, and in logs) by using any of the following approaches:

- One-way hashes based on strong cryptography with associated key management processes and procedures
- Truncation

- Index tokens and pads (pads must be securely stored)
- Strong cryptography with associated key management processes and procedures. The MINIMUM account information that must be rendered unreadable is the PAN. PCI Data Security Standard Requirement 3.4

The PAN must be rendered unreadable anywhere it is stored, even outside the payment application. Note: Strong cryptography is defined in the PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms.

2.4 If disk encryption is used (rather than file- or column-level database encryption), logical access must be managed independently of native operating system access control mechanisms (for example, by not using local user account databases). Decryption keys must not be tied to user accounts. PCI Data Security Standard Requirement 3.4.2

2.5 Payment application must protect cryptographic keys used for encryption of cardholder data against disclosure and misuse. PCI Data Security Standard Requirement 3.5

2.6 Payment application must implement key management processes and procedures for cryptographic keys used for encryption of cardholder data. PCI Data Security Standard Requirement 3.6

2.7 Securely delete any cryptographic key material or cryptogram stored by previous versions of the payment application, in accordance with industry-accepted standards for secure deletion, as defined, for example the list of approved products maintained by the National Security Agency, or by other State or National standards or regulations. These are cryptographic keys used to encrypt or verify cardholder data. PCI Data Security Standard Requirement 3.6

Note: This requirement only applies if previous versions of the payment application used cryptographic key materials or cryptograms to encrypt cardholder data.

### 3. Provide secure authentication features

3.1 The payment application must support and enforce unique user IDs and secure authentication for all administrative access and for all access to cardholder data. Secure authentication must be enforced to all accounts, generated or managed by the application by the completion of installation and for subsequent changes after the "out of the box" installation (defined at PCI DSS Requirements 8.1, 8.2, and 8.5.88.5.15) for all administrative access and for all access to cardholder data. PCI Data Security Standard Requirements 8.1, 8.2, and 8.5.88.5.15

Note: These password controls are not intended to apply to employees who only have access to one card number at a time to facilitate a single transaction. These controls are applicable for access by employees with administrative capabilities, for access to servers with cardholder data, and for access controlled by the payment application. This requirement applies to the payment application and all associated tools used to view or access cardholder data.

3.1.10 If a payment application session has been idle for more than 15 minutes, the application requires the user to re-authenticate. PCI Data Security Standard Requirement 8.5.15.

3.2 Software vendors must provide guidance to customers that all access to PCs, servers, and databases with payment applications must require a unique user ID and secure authentication. PCI Data Security Standard Requirements 8.1 and 8.2

3.3 Render payment application passwords unreadable during transmission and storage, using strong cryptography based on approved standards

Note: Strong cryptography is defined in PCI DSS and PA-DSS Glossary of Terms, Abbreviations, and Acronyms. PCI Data Security Standard Requirement 8.4

### 4. Log payment application activity

4.1 At the completion of the installation process, the out of the box default installation of the payment application must log all user access (especially users with administrative privileges), and be able to link all activities to individual users. PCI Data Security Standard Requirement 10.1

4.2 Payment application must implement an automated audit trail to track and monitor access. PCI Data Security Standard Requirements 10.2 and 10.3

### 5. Develop secure payment applications

5.1 Develop all payment applications in accordance with PCI DSS (for example, secure authentication and logging) and based on industry best practices and incorporate information security throughout the software development life cycle. These processes must include the following: PCI Data Security Standard Requirement 6.3

5.1.1 Live PANS are not used for testing or development. PCI Data Security Standard Requirement 6.4.4.

- Validation of all input (to prevent cross-site scripting, injection flaws, malicious file execution, etc.)
- Validation of proper error handling
- Validation of secure cryptographic storage
- Validation of secure communications
- Validation of proper role-based access control (RBAC)

5.1.2 Separate development/test, and production environments

5.1.3 Removal of test data and accounts before production systems become active development. PCI Data Security Standard Requirement 6.4.4

5.1.4 Review of payment application code prior to release to customers after any significant change, to identify any potential coding vulnerability. Removal of custom payment application accounts, user IDs, and passwords before payment applications are released to customers

Note: This requirement for code reviews applies to all payment application components (both internal and public-facing web applications), as part of the system development life cycle required by PA-DSS Requirement 5.1 and PCI DSS Requirement 6.3. Code reviews can be conducted by knowledgeable internal personnel or third parties.

5.2 Develop all web payment applications (internal and external, and including web administrative access to product) based on secure coding guidelines such as the Open Web Application Security Project Guide. Cover prevention of common coding vulnerabilities in software development processes, to include:

- Injection flaws, with particular emphasis on SQL injection, Cross-site scripting (XSS) OS Command Injection, LDAP and Xpath injection flaws, as well as other injection flaws.
- Buffer Overflow.
- Insecure cryptographic storage.
- Insecure communications.
- Improper error handling.
- All HIGH vulnerabilities as identified in the vulnerability identification process at PA-DSS Requirement 7.1.
- Cross-site scripting (XSS)
- Improper access control such as insecure direct object references, failure to restrict URL access and directory traversal.
- Cross-site request forgery (CSRF)

Note: The vulnerabilities listed in PA-DSS Requirements 5.2.1 through 5.2.9 and in PCI DSS at 6.5.1 through 6.5.9 were current in the OWASP guide when PCI DSS v1.2 / PCI DSS v2.0 (01/01/10) were published. However, if and when the OWASP guide is updated, the current version must be used for these requirements.

5.3 Software vendor must follow change control procedures for all product software configuration changes. PCI Data Security Standard Requirement 6.4. 5.The procedures must include the following:

- Documentation of impact
- Management sign-off by appropriate parties
- Testing functionality to verify the new change(s) does not adversely impact the security of the system. Remove all testing configurations, samples, and data before finalizing the product for production.

- Back-out or product de-installation procedures

5.4 The payment application must not use or require use of unnecessary and insecure services and protocols (for example, NetBIOS, file-sharing, Telnet, unencrypted FTP must be secured via SSH, S-FTP, SSL, IPsec and other technology to implement end to end security). PCI Data Security Standard Requirement 2.2.2

## 6. Protect wireless transmissions

6.1 For payment applications using wireless technology, the wireless technology must be implemented securely. Payment applications using wireless technology must facilitate use of industry best practices (for example, IEEE 802.11i) to implement strong encryption for authentication and transmission. Controls must be in place to protect the implemented wireless network from unknown wireless access points and clients. This includes testing the end users wireless deployment on a quarterly basis to detect unauthorized access points within the system. Change wireless vendor defaults, including but not limited to default wireless encryption keys, passwords, and SSID community strings. Maintain a detailed updated hardware list. The end to end wireless implementation must be end to end secure. The use of WEP as a security control was prohibited as of 30 June 2010. PCI Data Security Standard Requirements 1.2.3, 2.1.1, 4.1.1, 6.2, 11.1a-e and 11.4a-c.

## 7. Test payment applications to address vulnerabilities

7.1 Software vendors must establish a process to identify newly discovered security vulnerabilities (for example, subscribe to alert services freely available on the Internet) and to test their payment applications for vulnerabilities. Any underlying software or systems that are provided with or required by the payment application (for example, web servers, third-party libraries and programs) must be included in this process. Remove all test configurations, samples, and data after testing and before promoting the changes to production. PCI Data Security Standard Requirement 6.2

7.2 Software vendors must establish a process for timely development and deployment of security patches and upgrades, which includes delivery of updates and patches in a secure manner with a known chain-of-trust, and maintenance of the integrity of patch and update code during delivery and deployment.

## 8. Facilitate secure network implementation

8.1 The payment application must be able to be implemented into a secure network environment. Application must not interfere with use of devices, applications, or configurations required for PCI DSS compliance (for example, payment application cannot interfere with anti-virus protection, firewall configurations, or any other device, application, or configuration required for PCI DSS compliance). PCI Data Security Standard Requirements 1, 3, 4, 5, and 6.

## 9. Cardholder data must never be stored on a server connected to the Internet

9.1 The payment application must be developed such that the database server and web server are not required to be on the same server, nor is the database server required to be in the DMZ with the web server. PCI Data Security Standard Requirement 1.3.7

## 10. Facilitate secure remote software updates

10.1 If payment application updates are delivered securely via remote access into customers systems, software vendors must tell customers to turn on remote-access technologies only when needed for downloads from vendor



and to turn off immediately after download completes. Alternatively, if delivered via VPN or other high-speed connection, software vendors must advise customers to properly configure a firewall or a personal firewall product to secure authentication using a two factor authentication mechanism. PCI Data Security Standard Requirement 8.3

10.2 If payment application may be accessed remotely, remote access to the payment application must be authenticated using a two factor authentication mechanism. PCI Data Security Standard Requirement 8.3

10.3 Any remote access into the payment application must be done securely. If vendors, resellers/integrators, or customers can access customers payment applications remotely, the remote access must be implemented securely. PCI Data Security Standard Requirements 1, 8.3 and 12.3.9

#### 11. Encrypt sensitive traffic over public networks

11.1 If the payment application sends, or facilitates sending, cardholder data over public networks, the payment application must support use of strong cryptography and security protocols such as SSL/TLS and Internet protocol security (IPSEC) to safeguard sensitive cardholder data during transmission over open, public networks. Examples of open, public networks that are in scope of the PCI DSS are: The Internet Wireless technologies Global System for Mobile Communications (GSM) General Packet Radio Service (GPRS) PCI Data Security Standard Requirement 4.1

11.2 The payment application must never send unencrypted PANs by end-user messaging technologies (for example, e-mail, instant messaging, and chat). PCI Data Security Standard Requirement 4.2

#### 12. Encrypt all non-console administrative access

12.1 Instruct customers to encrypt all non-console administrative access using technologies such as SSH, VPN, or SSL/TLS for web-based management and other non-console administrative access. Telnet or remote login must never be used for administrative access. PCI Data Security Standard Requirement 2.3

#### 13. Maintain instructional documentation and training programs for customers, resellers, and integrators

13.1 Develop, maintain, and disseminate a PA-DSS Implementation Guide(s) for customers, resellers, and integrators that accomplishes the following:

- Addresses all requirements in this document wherever the PA-DSS Implementation Guide is referenced.
- Includes a review at least annually and updates to keep the documentation current with all major and minor software changes as well as with changes to the requirements in this document.

13.2 Develop and implement training and communication programs to ensure payment application resellers and integrators know how to implement the payment application and related systems and networks according to the PA-DSS Implementation Guide and in a PCI DSS-compliant manner.

- Update the training materials on an annual basis and whenever new payment application versions are released.

## 2.5 More Information

IDTech Systems, Inc. highly recommends that merchants contact the card association(s) or their processing company and find out exactly what they mandate and/or recommend. Doing so may help merchants protect themselves from fines and fraud.

For more information related to security, visit:

- <http://www.pcisecuritystandards.org>
- <http://www.visa.com/cisp>
- <http://www.sans.org/resources>
- <http://www.microsoft.com/security/default.asp>
- <https://sdp.mastercardintl.com/>
- <http://www.americanexpress.com/merchantspecs>

CAPN questions: [capninfocenter@aexp.com](mailto:capninfocenter@aexp.com)

## Chapter 3

# Main Transaction Commands

The methods below are provided as a reference to the main commands needed to execute an EMV transaction.

### 3.1 EMV Methods

#### Start EMV Transaction

```
IDTechSDK::IDT_SpectrumPro::emv_startTransaction()
```

Begins an amount authorization request with the ICC. Returns authorization decision (approved, denied, or go online) in callback method.

#### Authenticate EMV Transaction

```
IDTechSDK::IDT_SpectrumPro::emv_authenticateTransaction()
```

By default, auto-authenticate is ON and this step does not need to be performed. If auto-authenticate is OFF ([IDTechSDK::IDT\\_SpectrumPro::emv\\_autoAuthenticate\(\)](#)), when the results to [IDTechSDK::IDT\\_SpectrumPro::emv\\_startTransaction\(\)](#) come back as EMV\_RESULT\_CODE.EMV\_RESULT\_CODE\_AUTHENTICATE\_TRANSACTION, this method must be called to continue the EMV transaction.

#### Complete Online EMV Transaction

```
IDTechSDK::IDT_SpectrumPro::emv_completeTransaction()
```

After receiving a host response, pass host tags (minimum 8A Authorization Response Code) as a parameter.

If there was a communication error with host, you must still finish the EMV transaction by passing "TRUE" for commError.

#### Terminal Configuration

```
IDTechSDK::IDT_SpectrumPro::emv_retrieveTerminalData()
IDTechSDK::IDT_SpectrumPro::emv_removeTerminalData()
IDTechSDK::IDT_SpectrumPro::emv_setTerminalData()
```

Methods for terminal configuration. When setting the terminal data, you pass the tags in TLV format.

#### AID Management

```
IDTechSDK::IDT_SpectrumPro::emv_retrieveApplicationData:response()
IDTechSDK::IDT_SpectrumPro::emv_removeApplicationData()
```

```
IDTechSDK::IDT_SpectrumPro::emv_removeAllApplicationData()  
IDTechSDK::IDT_SpectrumPro::emv_setApplicationData()  
IDTechSDK::IDT_SpectrumPro::emv_retrieveAIDList()
```

Methods for AID management. When setting the AID, you pass tags in TLV format. When retrieving AID, you can receive the results as tags in TLV format. When retrieving the AID list, the list of AID Names/length can be retrieved from the 2 dimensional byte array

#### CAPK Management

```
IDTechSDK::IDT_SpectrumPro::emv_retrieveCAPK()  
IDTechSDK::IDT_SpectrumPro::emv_removeCAPK()  
IDTechSDK::IDT_SpectrumPro::emv_removeAllCAPK()  
IDTechSDK::IDT_SpectrumPro::emv_setCAPK()  
IDTechSDK::IDT_SpectrumPro::emv_retrieveCAPKList()
```

Methods for Certificate Authority Public Key management. When setting the CAPK, you populate and pass the key as a sequence of ordered bytes. When specifying a CAPK to retrieve or remove, you populate the name in the byte[] parameter. When retrieving the CAPK list, the list of RID/Index can be retrieved from the ordered byte stream, 6 bytes each, bytes 1-5 RID, byte 6 index

#### CRL Management

```
IDTechSDK::IDT_SpectrumPro::emv_removeCRL()  
IDTechSDK::IDT_SpectrumPro::emv_removeAllCRL()  
IDTechSDK::IDT_SpectrumPro::emv_retrieveCRLList()  
IDTechSDK::IDT_SpectrumPro::emv_setCRL:()
```

Methods for Certificate Revocation List management.

#### Kernel Version

```
IDTechSDK::IDT_SpectrumPro::emv_getEMVKernelVersion()
```

Method to retrieve kernel version. Valued returned in IDTResult.data

## 3.2 MSR Methods

#### Start MSR Swipe

```
IDTechSDK::IDT_SpectrumPro::msr_startMSRSwipe()
```

Starts a swipe request. Returns card data in callback method.

#### Cancel MSR Swipe

```
IDTechSDK::IDT_SpectrumPro::msr_cancelMSRSwipe()
```

Cancels a swipe request.

## Chapter 4

# Connecting to SpectrumPro

The SpectrumPro connects through either USB or Serial Interface (COM).

### 4.1 Connect with USB:

The SpectrumPro uses USB-HID (Human Interface Device) and does not need any vendor-specific drivers. Simply by plugging into an available USB port makes it available for SDK connectivity. To inform the SDK you are using the USB interface of the SpectrumPro, you would execute the following command during program initialization to establish a connection:

```
IDT_SpectrumPro.useUSB();
```

### 4.2 Connect with Serial Interface (COM)

The SpectrumPro can connect via Serial Interface. The default serial port settings are as follows:

- Speed: 115,200
- Bits: 8
- Stop Bit: 1
- Parity: None

To inform the SDK you are using the Serial Interface of the SpectrumPro, you would execute the following command during program initialization to establish a connection by passing the correct COM port number. In the following example, we are connecting to COM1 non-SRED device:

```
IDT_SpectrumPro.useSerialPort(1, false);
```

If you change the default baud rate, then there is an overloaded method to also include the baud rate:

```
IDT_SpectrumPro.useSerialPort(1, 115200, false);
```

## Chapter 5

# Core Implementation: WinForms

IDTechSDK.dll includes class libraries to interface with the SpectrumPro. This guide assume a fair understanding of Visual Studio 2013+, C# and general Windows programming knowledge.

### 5.1 Integrating with IDTechSDK.dll

- [Import the necessary libraries](#)
- [Add using statements to utilize library](#)
- [Implement the callback function](#)
- [Initialize SpectrumPro](#)
- [Sample Project Tutorial](#)

### 5.2 Import the necessary libraries

Communicating with IDTech Devices requires the following library to be referenced by the project:

- IDTechSDK.dll

Add the reference as you would any .NET managed library reference. The most direct method would be right-click on the "References" in the Solution Explorer for the project, select "Add Reference...", click "Browse..." and locate IDTechSDK.dll.

IDTechSDK.dll has a dependency of Microsoft .NET 4.5 or greater. Please make sure your final application installer checks for this dependency and installs it if not on the destination system.

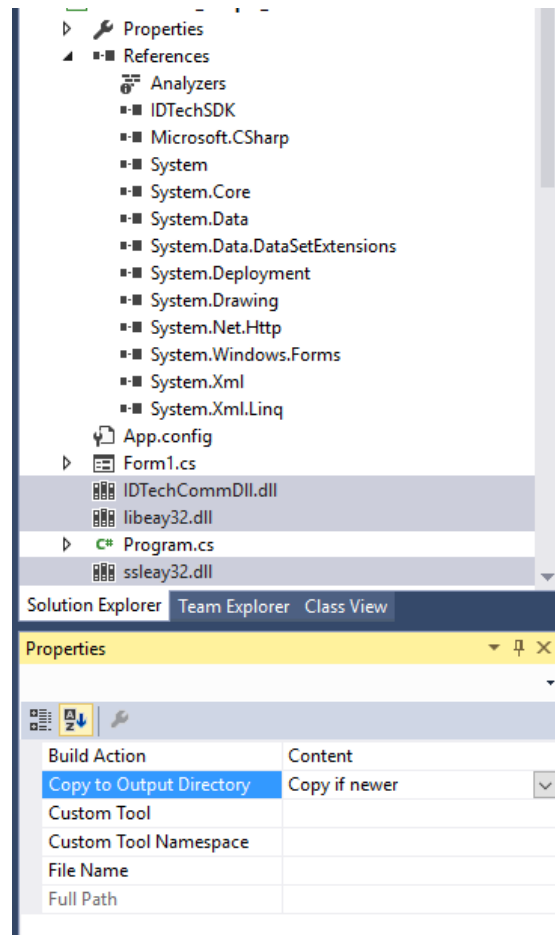
In addition, the following libraries need to be added to project folder and included in with the application distribution:

- IDTechCommDll.dll
- libeay32.dll
- ssleay.32.dll
- SpectrumPro\_config.dll
- SpectrumPro\_device.dll
- SpectrumPro\_emv.dll
- SpectrumPro\_icc.dll

- SpectrumPro\_msr.dll
- SpectrumPro\_parse.dll
- SpectrumPro\_pin.dll

You can add these libraries by right-clicking on your project name in the solution Explorer and select "Add->Existing Item..." or keyboard shortcut Shift-Alt-A.

Once all three items are added, set their properties to "Copy if newer" so they will be included in the final applications destination folder.



## 5.3 Add using statements to utilize library

Add a line of code to the class that will utilize IDTechSDK.dll at the start of the file:

```
using IDTechSDK;
```

## 5.4 Implement the callback function

There is a single method that will receive all callback information from the SDK. It uses DeviceState to determine which action to take.

```

private void MessageCallBack(IDTechSDK.IDT_DEVICE_Types type, DeviceState state, byte[] data,
    IDTTransactionData cardData, EMV_Callback emvCallback, RETURN_CODE transactionResultCode)
{
    switch (state)
    {
        case DeviceState.ToConnect:
            //A connection attempt is starting for IDT_DEVICE_TYPES type
            break;
        case DeviceState.DefaultDeviceTypeChange:
            //The SDK is changing the default device to IDT_DEVICE_TYPES type
            break;
        case DeviceState.Connected:
            //A connection has been made to IDT_DEVICE_TYPES type
            break;
        case DeviceState.Disconnected:
            //A disconnection has occurred with IDT_DEVICE_TYPES type
            break;
        case DeviceState.ConnectionFailed:
            //A connection attempt has failed for IDT_DEVICE_TYPES type
            break;
        case DeviceState.TransactionData:
            //Transaction data is being returned in IDTTransactionData cardData
            break;
        case DeviceState.DataReceived:
            //Low-level data received for IDT_DEVICE_TYPES type
            break;
        case DeviceState.DataSent:
            //Low-level data sent for IDT_DEVICE_TYPES type
            break;
        case DeviceState.CommandTimeout:
            //Command timeout has occurred for IDT_DEVICE_TYPES type
            break;
        case DeviceState.ToSwipe:
            //Awaiting a swipe for IDT_DEVICE_TYPES type
            break;
        case DeviceState.MSRDecodeError:
            //Awaiting a swipe for IDT_DEVICE_TYPES type
            break;
        case DeviceState.ToTap:
            //Awaiting a contactless tap for IDT_DEVICE_TYPES type
            break;
        case DeviceState.SwipeTimeout:
            //Waiting for swipe timed out
            break;
        case DeviceState.TransactionCancelled:
            //Transaction has been cancelled
            break;
        case DeviceState.DeviceTimeout:
            //Waiting for EMV transaction to complete timed out
            break;
        case DeviceState.TransactionFailed:
            //Transaction failed to complete
            break;
        case DeviceState.EMVCallback:
            //Callback during Device transaction retrieved from EMV_Callback emvCallback
            break;
        case DeviceState.FirmwareUpdate:
            //Callback during Firmware Update
            break;
        default:
            break;
    }
}
}

```

## 5.5 Initialize SpectrumPro:

A Singleton instance has been established in the IDT\_SpectrumPro class. Establish the callback, and then access the SpectrumPro through USB via the SharedController instance.

```

public SpectrumPro_Simple_Demo()
{
    InitializeComponent();
    IDT_SpectrumPro.setCallback(MessageCallBack);
}

```



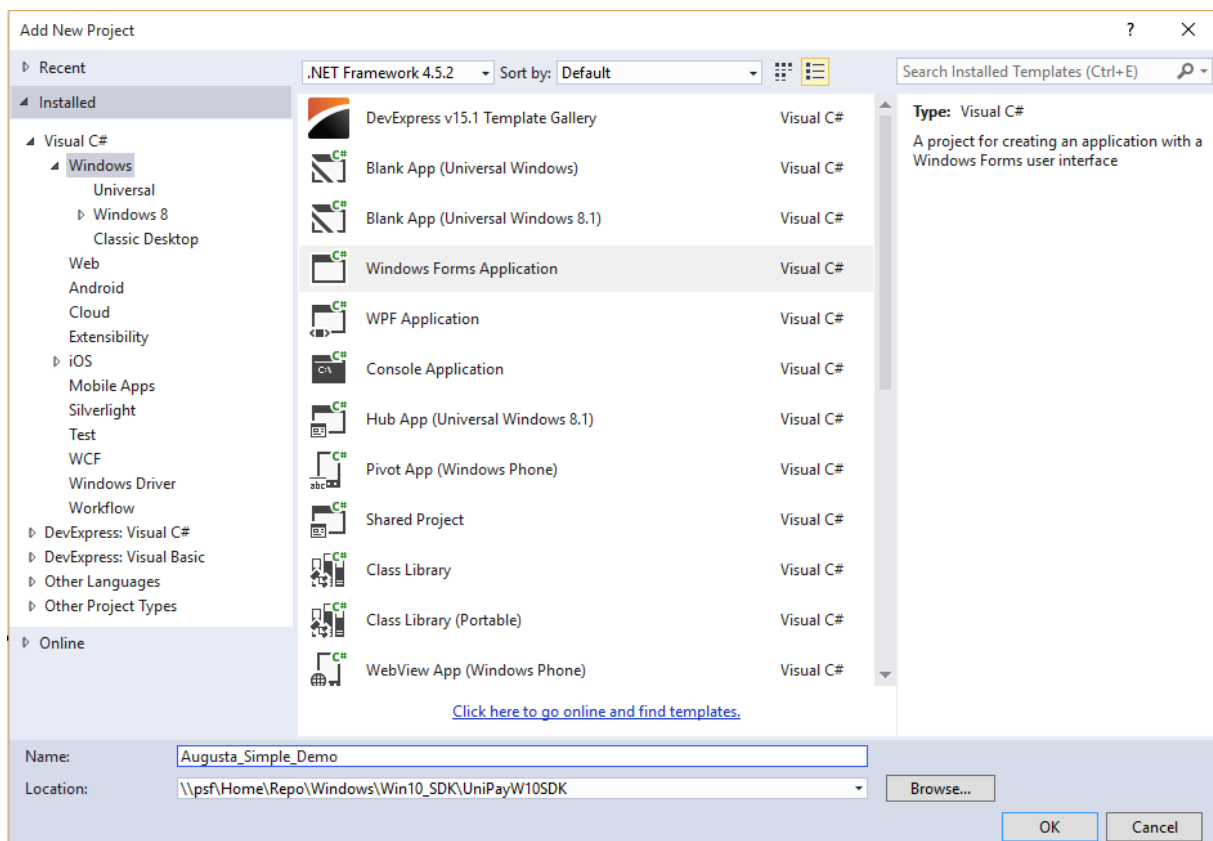
## 5.6 Sample Project Tutorial

Using Visual Studio 2015, we will create a C# sample project that will interface with the SpectrumPro and will perform the following activities:

- Select USB or COM interface
- Set UID and report device info
- Display ICC Status (card inserted/removed, ICC power on/off)
- Turn on/off ICC reader
- Get/Set terminal data file
- Get/Set AID for Visa
- Get/Set CAPKs for Visa
- Perform EMV Transaction
- Start/Stop a Swipe Request
- Show log of all data going to/from SpectrumPro

### 5.6.1 Step 1: Create New Project

Create a new Windows Form Application in Visual Studio. In our example, we use project name SpectrumPro\_↔ Simple\_Demo



### 5.6.2 Step 2: Import Libraries

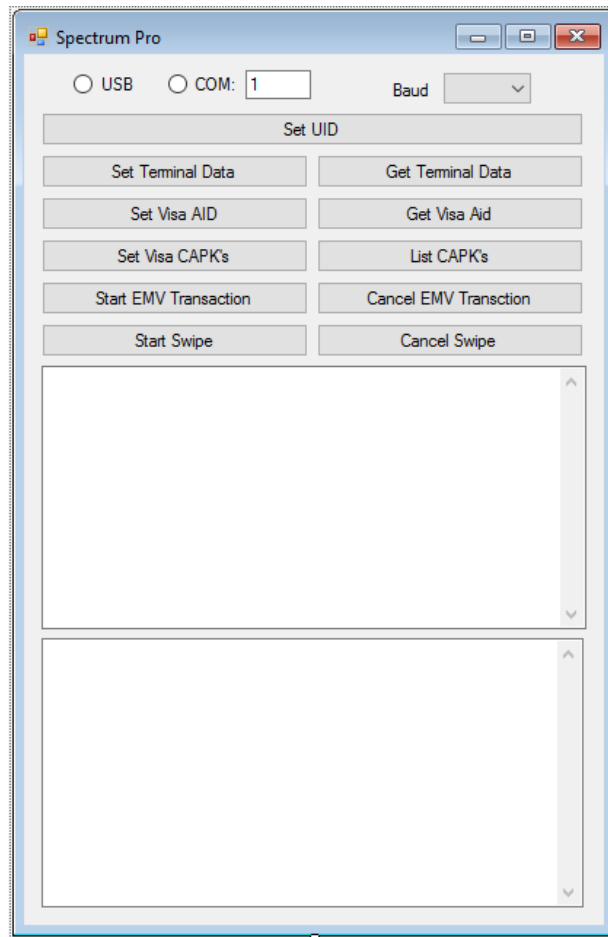
[Import the necessary libraries](#)

### 5.6.3 Step 3: Design Interface

Use the Form Designer to layout buttons/fields

Open your form designer and add items so it contains the following buttons/fields:

- Radio buttons for USB and COM selection. For COM selection, add a text box to specify COM port
  
- Add buttons to execute the following functions:
  - Set UID
  - Show ICC Status
  - Turn ON ICC Reader
  - Turn OFF ICC Reader
  - Set Terminal Data
  - Get Terminal Data
  - Set Visa AID
  - Get Visa AID
  - Set Visa CAPKs
  - List All CAPKs
  - Start EMV Transaction
  - Cancel EMV Transaction
  - Start Swipe
  - Cancel Swipe
  
- Add a text box to communicate data from the SpectrumPro.
  
- Add a text box for the log of SpectrumPro.



#### 5.6.4 Step 4: Configure the project file

In the start of the class file, perform the following:

- [Add using statements to utilize library](#)
- set callback method and initialize SpectrumPro singleton object in the class initializer. Reference: [Initialize SpectrumPro](#)
- Implement the button methods Click handlers for button press code execution

```
private void btnFirmwareVersion_Click(object sender, EventArgs e)
{
    SpectrumInfo info = new SpectrumInfo();
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.device_setUID(IDT_Device.Spectrum_UID, 2, ref info);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("UID set Successful\r\nFirmware: " + info.firmwareVersion + "\r\nApplication: "
            + info.applicationVersion + "\r\nMSR: " + info.msrHead + "\r\nHardware: " + info.hardwareVersion + "\r\n"
            + "Serial Number: " + info.serialNumber + "\r\nKey Mode: " + info.keyType.ToString() + "\r\n");
    }
    else
    {
        tbOutput.AppendText("UID set failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + ": "
            + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
    }
}

private void btnStartEMVTransaction_Click(object sender, EventArgs e)
{
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_startTransaction(1.00, 0, 2, 0, 30, null, false);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Start EMV Successful\r\n");
    }
    else
    {
    }
}
```

```

        {
            tbOutput.AppendText("Start EMV failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + "
            : " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        }
    }

private void btnCancelEMVTransaction_Click(object sender, EventArgs e)
{
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_cancelTransaction();
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Cancel Transaction Successful\r\n");
        System.Diagnostics.Debug.WriteLine("Cancel Transaction Successful");
    }
    else
    {
        tbOutput.AppendText("Cancel Transaction failed Error Code: " + "0x" + String.Format("{0:X}", (
        ushort)rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Cancel Transaction failed Error Code: " + "0x" + String.
        Format("{0:X}", (ushort)rt));
    }
}

private void btnSetTerminalData_Click(object sender, EventArgs e)
{
    byte[] term = Common.getByteArray("
    5f3601029f1a0208409f3501229f330360f8c89f4005f00000a0019f1e085465726d6
    96e616c9f150212349f160f303030303030303030303030303030309f1c0838373635343332319f4e2231303732312057616c6b6572205
    3742e20437970726573732c204341202c5553412edf260101df1008656e667265737a68df110101df270100dfee150101dfee160100d
    fee170107dfee180180dfee1e08f0dc3cf0c29e9400dfee1f0180dfee1b083030303135313030dfee20013cdf21010adfee2203323c3c");
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_setTerminalData(term);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Set Terminal Successful:" + " \r\n");
    }
    else
    {
        tbOutput.AppendText("Save Terminal failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
        + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
    }
}

private void btnGetTerminalData_Click(object sender, EventArgs e)
{
    byte[] tlv = null;
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_retrieveTerminalData(ref tlv);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Retrieve Terminal Successful- TLV: " + " \r\n" + Common.getHexStringFromBytes
        (tlv) + "\r\n");
    }
    else
    {
        tbOutput.AppendText("Retrieve Terminal failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)
        rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
    }
}

private void btnSetVisaAid_Click(object sender, EventArgs e)
{
    RETURN_CODE rt;
    byte[] name = Common.getByteArray("a0000000031010");
    byte[] aid = Common.getByteArray("
    9f01065649534130305f5701005f2a0208409f090200965f3601029f1b0400003a98df
    25039f3704df28039f0802dfee150101df13050000000000df14050000000000df15050000000000df180100df170400002710df190100");
    rt = IDT_SpectrumPro.SharedController.emv_setApplicationData(name, aid);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Default AID Successful\r\n");
    }
    else
    {
        tbOutput.AppendText("Default AID failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) +
        ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
    }
}

private void btnGetVisaAid_Click(object sender, EventArgs e)
{
    byte[] tlv = null;
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_retrieveApplicationData(Common.getByteArray("
    a0000000031010"), ref tlv);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {

```

```

        tbOutput.AppendText("Retrieve AID Successful- TLV: " + " \r\n" + Common.getHexStringFromBytes(tlv)
+ " \r\n");
    }
    else
    {
        tbOutput.AppendText("Retrieve AID failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
+ ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Retrieve AID failed Error Code: " + "0x" + String.Format(
"{0:X}", (ushort)rt));
    }
}

private void btnSetVisaCAPKs_Click(object sender, EventArgs e)
{
    byte[] capk = Common.getByteArray("
a000000003500101b769775668cacb5d22a647d1d993141edab7237b000100018000d
11197590057b84196c2f4d11a8f3c05408f422a35d702f90106ea5b019bb28ae607aa9cdebcd0d81a38d48c7ebb0062d287369ec0c42
124246ac30d80cd602ab7238d51084ded4698162c59d25eac1e66255b4db235252ef0982c3b8ad3d1cce85b01db5788e75e09f44be736136
RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
capk = Common.getByteArray("
a000000003510101b9d248075a3f23b522fe45573e04374dc4995d71000000039000db5fa29d
1fda8c1634b04dccff148abee63c772035c79851d3512107586e02a917f7c7e885e7c4a7d529710a145334ce67dc412cb1597b77aa25
43b98d19cf2cb80c522bdbbea0f1b113fa2c86216c8c610a2d58f29cf3355ceb1bd3ef410d1edd1f7ae0f16897979d28c6ef293e0a1928bd
rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
capk = Common.getByteArray("
a000000003530101ac213a2e0d2c0ca35ad0201323536d58097e4e5700000003f800bcd83721
be52cccc4b6457321f22a7dc769f54eb8025913be804d9eabbfa19b3d7c5d3ca658d768caf57067eec83c7e6e9f81d0586703ed9ddda
dd20675d63424980b10eb364e81eb37db40ed100344c928886ff4ccc37203ee6106d5b59d1ac102e2cd2d7ac17f4d96c398e5fd993ec
b4ffdf79b17547ff9fa2aa8eefd6cbda124cbb17a0f8528146387135e226b005a474b9062ff264d2ff8efa36814aa2950065b1b04c0a
1ae9b2f69d4a4aa979d6ce95fee9485ed0a03aee9bd953e81cfd1ef6e814dfd32ce37aefa38c1f9877371e91d6a5eb59fddedf75d3325fa3c
rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
capk = Common.getByteArray("
a0000000039601017616e9ac8be014af88call1a8fb17967b7394030e000000038000b74586d1
9a207be6627c5b0aafbc44a2ecf5a2942d3a26ce19c4ffaee920521868922e893e7838225a3947a2614796fb2c0628ce8c11e3825a5
6d3b1bbaef783a5c6a81f36f8625395126fa983c5216d3166d48acde8a431212ff763a7f79d9edb7fed76b485de45beb829a3d4730848a366
rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
capk = Common.getByteArray("
a000000003570101251a5f5de61cf28b5c6e2b5807c0644a01d46ff5000100016000942b7f2b
a5ea307312b63df77c5243618acc2002bd7ecb74d821fe7bdc78bf28f49f74190ad9b23b9713b140ffec1fb429d93f56bdc7ade4ac075d755
rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
capk = Common.getByteArray("
a000000003580101753ed0aa23e4cd5abd69eae7904b684a34a57c2200010001c80099552c4a
1ecd68a0260157fc4151b5992837445d3fc57365ca5692c87be358cdcdf2c92fb6837522842a48eb11cdf2fd91770c7221e4af6207
c2de4004c7deeb16276dc62d52a87d2cd01fbf2dc4065db52824d2a2167a06d19e6a0f781071cdb2dd314cb94441d8dc0e936317b77b
f06f5177f6c5aba3a3bc6aa30209c97260b7a1ad3a192c9b8cd1d153570afccc87c3cd681d13e997fe33b3963a0a1c79772acf991033e1b839
rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
capk = Common.getByteArray("
a00000000354010106960618791a86d387301edd4a3baf2d34fef1b400010001f800c6ddc0b7
645f7f16286ab7e4116655f56dd0c944766040dc68664dd973bd3bfd4c525bcb95272b6b3ad9ba8860303ad08d9e8cc344a4070f4cf
b9eeaf29c8a3460850c264cda39bbe3a7e7d08a69c31b5c8dd9f94ddbc9265758c0e7399adcf4362caee458d414c52b498274881b196
dacca7273f687f2a65faeb809d4b2ac1d3d1efb4f6490322318bd296d153b307a3283ab4e5be6ebd910359a8565eb9c4360d24baaca3
dbfe393f3d6c830d603c6fc1e83409dfcd80d3a33ba243813bbb4ceaf9cbab6b74b00116f72ab278a88a011d70071e06cab140646438d986d
rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
{
    tbOutput.AppendText("Load Visa CAPK Successful:" + " \r\n");
}
else
{
    tbOutput.AppendText("Load Visa CAPK failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
+ ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
}
}

private void btnListCAPKs_Click(object sender, EventArgs e)
{
    byte[] capk = null;
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_retrieveCAPKList(ref capk);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("List CAPK Successful \r\n");
        if (capk.Length > 0)
        {
            for (int x = 0; x < capk.Length; x = x + 6)
            {
                byte[] thecapk = new byte[] { capk[x], capk[x + 1], capk[x + 2], capk[x + 3], capk[x + 4],
                capk[x + 5] };
                tbOutput.AppendText(Common.getHexStringFromBytes(thecapk) + " \r\n");
            }
        }
        else
        {
            tbOutput.AppendText("List CAPK failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + "

```

```

        : " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
    }
}

private void btnStartSwipe_Click(object sender, EventArgs e)
{
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.msr_startMSRSwipe(60);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("MSR Turned On successfully; Ready to swipe\r\n");
        tbOutput.ReadOnly = false;
        tbOutput.Focus();
    }
    else
    {
        tbOutput.AppendText("MSR Turned On failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
            + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
    }
}

private void btnCancelSwipe_Click(object sender, EventArgs e)
{
    RETURN_CODE rt = IDT_SpectrumPro.SharedController.msr_cancelMSRSwipe();
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("MSR Turned Off successfully\r\n");
        System.Diagnostics.Debug.WriteLine("MSR Turned Off successfully");
    }
    else
    {
        tbOutput.AppendText("MSR Turned Off failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
            + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        System.Diagnostics.Debug.WriteLine("MSR Turned Off failed Error Code: " + "0x" + String.
            Format("{0:X}", (ushort)rt));
    }
}
}

```

### 5.6.5 Step 5: Configure callback to receive important SDK data (messages, log info and transaction results)

```

private void MessageCallback(IDTechSDK.IDT_DEVICE_Types type, DeviceState state, byte[] data,
    IDTTransactionData cardData, EMV_Callback emvCallback, RETURN_CODE transactionResultCode)
{
    switch (state)
    {
        case DeviceState.ToConnect:
            //A connection attempt is starting for IDT_DEVICE_TYPES type
            SetOutputText("Callback:ToConnect\n");
            break;
        case DeviceState.DefaultDeviceTypeChange:
            //The SDK is changing the default device to IDT_DEVICE_TYPES type
            SetOutputText("Callback:DefaultDeviceTypeChange\n");
            break;
        case DeviceState.Connected:
            //A connection has been made to IDT_DEVICE_TYPES type
            SetOutputText("Callback:Connected\n");
            break;
        case DeviceState.Disconnected:
            //A disconnection has occurred with IDT_DEVICE_TYPES type
            SetOutputText("Callback:Disconnected\n");
            break;
        case DeviceState.ConnectionFailed:
            //A connection attempt has failed for IDT_DEVICE_TYPES type
            SetOutputText("Callback:ConnectionFailed\n");
            break;
        case DeviceState.TransactionData:
            //Transaction data is beign returned in IDTTransactionData cardData
            SetOutputText("Callback:TransactionData\n");
            if (cardData.emv_resultCode == EMV_RESULT_CODE.EMV_RESULT_CODE_GO_ONLINE)
            {
                //auto complete. Normally, a host response is required here
                SetOutputText("Online request. Auto Complete EMV Transaction.\n");
                byte[] responseCode = new byte[] { 0x30, 0x30 };
                byte[] iad = new byte[] { 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x30, 0x30 };
            }
            RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_completeTransaction(false,
                responseCode, iad, null, null);
            return;
        }
        displayCardData(cardData);
        break;
        case DeviceState.DataReceived:
            //Low-level data received for IDT_DEVICE_TYPES type
            SetOutputTextLog(GetTimestamp() + " IN: " + Common.getHexStringFromBytes(data));
    }
}

```

```

        break;
    case DeviceState.DataSent:
        //Low-level data sent for IDT_DEVICE_TYPES type
        SetOutputTextLog(GetTimestamp() + " OUT: " + Common.getHexStringFromBytes(data));
        break;
    case DeviceState.CommandTimeout:
        SetOutputText("Callback:CommandTimeout\n");
        //Command timeout has occurred for IDT_DEVICE_TYPES type
        break;
    case DeviceState.ToSwipe:
        //Awaiting a swipe for IDT_DEVICE_TYPES type
        SetOutputText("Callback:ToSwipe\n");
        break;
    case DeviceState.MSRDecodeError:
        //Awaiting a swipe for IDT_DEVICE_TYPES type
        SetOutputText("Callback:MSRDecodeError\n");
        break;
    case DeviceState.ToTap:
        //Awaiting a contactless tap for IDT_DEVICE_TYPES type
        SetOutputText("Callback:ToTap\n");
        break;
    case DeviceState.SwipeTimeout:
        //Waiting for swipe timed out
        SetOutputText("Callback:SwipeTimeout\n");
        break;
    case DeviceState.TransactionCancelled:
        //Transaction has been cancelled
        SetOutputText("Callback:TransactionCancelled\n");
        break;
    case DeviceState.DeviceTimeout:
        //Waiting for device timeout
        SetOutputText("Callback:DeviceTimeout\n");
        break;
    case DeviceState.TransactionFailed:
        //Transaction failed to complete
        SetOutputText("Callback:TransactionFailed\n");
        break;
    case DeviceState.EMVCallback:
        //Callback during EMV transaction retrieved from EMV_Callback emvCallback
        SetOutputText("Callback:EMVCallback\n");
        if (emvCallback.callbackType == EMV_CALLBACK_TYPE.EMV_CALLBACK_TYPE_LCD
    ) //LCD Callback Type
        {
            if (emvCallback.lcd_displayMode == EMV_LCD_DISPLAY_MODE.
EMV_LCD_DISPLAY_MODE_CLEAR_SCREEN)
            {
                SetOutputText("LCD Callback:Clear LCD Screen\n");
                return;
            }
            else if (emvCallback.lcd_displayMode ==
EMV_LCD_DISPLAY_MODE.EMV_LCD_DISPLAY_MODE_MESSAGE)
            {
                // A message is requested to be displayed. See other test app included with SDK
                for a complete example.
                SetOutputText("LCD Callback:Display Message\n");
            }
            else
            {
                //Display message with menu/language or prompt, and return result to
                emv_callbackResponseLCD
                //Kernel will not proceed until this step is complete
                //seeing to default value of 1. See other test app included with SDK for a
                complete example
                SetOutputText("LCD Callback:Menu Display Request. Sending result 1\n");
                IDT_SpectrumPro.SharedController.emv_callbackResponseLCD(emvCallback.
                lcd_displayMode, 1);
            }
        }
        break;
    default:
        break;
}

}

private void displayCardData(IDTTransactionData cardData)
{
    string text = "";

    if (cardData.Event == EVENT_TRANSACTION_DATA.Types.EVENT_TRANSACTION_PIN_DATA)
    {

```

```

        SetOutputText("PIN Data received:\r\nKSN: " + cardData.pin_KSN + "\r\nPINBLOCK: " + cardData
        .pin_pinblock + "\r\n");
        return;
    }

    if (cardData.Event == EVENT_TRANSACTION_DATA_Types.EVENT_TRANSACTION_DATA_CARD_DATA)
    {
        SetOutputText("Data received: (Length [" + cardData.msr_rawData.Length.ToString() + "])\n" +
        string.Concat(cardData.msr_rawData.ToArray().Select(b => b.ToString("X2")).ToArray()) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Data received: (Length [" + cardData.msr_rawData.
        Length.ToString() + "])\n" + string.Concat(cardData.msr_rawData.ToArray().Select(b => b.ToString("X2")).
        ToArray()));
    }

    if (cardData.device_RSN != null && cardData.device_RSN.Length > 0)
        text += "Serial Number: " + cardData.device_RSN + "\r\n";

    if (cardData.msr_track1Length > 0)
        text += "Track 1: " + cardData.msr_track1 + "\r\n";

    if (cardData.msr_encTrack1 != null)
        text += "Track 1 Encrypted: " + Common.getHexStringFromBytes(cardData.msr_encTrack1) + "\r\n";
";

    if (cardData.msr_hashTrack1 != null)
        text += "Track 1 Hash: " + Common.getHexStringFromBytes(cardData.msr_hashTrack1) + "\r\n";

    if (cardData.msr_track2Length > 0)
        text += "Track 2: " + cardData.msr_track2 + "\r\n";

    if (cardData.msr_encTrack2 != null)
        text += "Track 2 Encrypted: " + Common.getHexStringFromBytes(cardData.msr_encTrack2) + "\r\n";
";

    if (cardData.msr_hashTrack2 != null)
        text += "Track 2 Hash: " + Common.getHexStringFromBytes(cardData.msr_hashTrack2) + "\r\n";

    if (cardData.msr_track3Length > 0)
        text += "Track 3: " + cardData.msr_track3 + "\r\n";

    if (cardData.msr_encTrack3 != null)
        text += "Track 3 Encrypted: " + Common.getHexStringFromBytes(cardData.msr_encTrack3) + "\r\n";
";

    if (cardData.msr_hashTrack3 != null)
        text += "Track 3 Hash: " + Common.getHexStringFromBytes(cardData.msr_hashTrack3) + "\r\n";

    if (cardData.msr_KSN != null)
        text += "KSN: " + Common.getHexStringFromBytes(cardData.msr_KSN) + "\r\n";

    if (cardData.emv_clearingRecord != null)
    {
        if (cardData.emv_clearingRecord.Length > 0)
        {
            text += "\r\nCTLS Clearing Record: \r\n";
            text += Common.getHexStringFromBytes(cardData.emv_clearingRecord) + "\r\n";
            Dictionary<string, string> dict = Common.processTLVUnencrypted(cardData.
            emv_clearingRecord);
            foreach (KeyValuePair<string, string> kvp in dict) text += kvp.Key + ": " + kvp.Value +
            "\r\n";
            text += "\r\n\r\n";
        }
    }

    if (cardData.emv_unencryptedTags != null)
    {
        if (cardData.emv_unencryptedTags.Length > 0)
        {
            text += "\r\n===== \r\n";
            text += "\r\nUnencrypted Tags: \r\n";

```



```

        text += Common.getHexStringFromBytes(cardData.emv_unencryptedTags) + "\r\n\r\n";
        text += tlvToValues(cardData.emv_unencryptedTags);
        text += "\r\n===== \r\n";
    }

}

if (cardData.emv_encryptedTags != null)
{
    if (cardData.emv_encryptedTags.Length > 0)
    {
        text += "\r\n===== \r\n";
        text += "\r\nEncrypted Tags: \r\n";
        text += Common.getHexStringFromBytes(cardData.emv_encryptedTags) + "\r\n\r\n";
        text += tlvToValues(cardData.emv_encryptedTags);
        text += "\r\n===== \r\n";
    }
}

if (cardData.emv_maskedTags != null)
{
    if (cardData.emv_maskedTags.Length > 0)
    {
        text += "\r\n===== \r\n";
        text += "\r\nMasked Tags: \r\n";
        text += Common.getHexStringFromBytes(cardData.emv_maskedTags) + "\r\n\r\n";
        text += tlvToValues(cardData.emv_maskedTags);
        text += "\r\n===== \r\n";
    }
}

text += "ICC Present: ";
text += (cardData.iccPresent ? "TRUE" : "FALSE") + "\r\n";
text += "is CTLS: ";
text += (cardData.isCTLS ? "TRUE" : "FALSE") + "\r\n";

if (cardData.Event == EVENT_TRANSACTION_DATA_Types.EVENT_TRANSACTION_DATA_EMV_DATA)
{
    if (!cardData.isCTLS) text += "Capture Encrypt Type: " + ((cardData.emv_encryptionMode ==
EMV_ENCRYPTION_MODE.EMV_ENCRYPTION_MODE_TDES) ? "TDES" : "AES") + "\r\n";
    switch (cardData.emv_resultCode)
    {
        case EMV_RESULT_CODE.EMV_RESULT_CODE_APPROVED:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_APPROVED" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_APPROVED_OFFLINE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_APPROVED_OFFLINE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_DECLINED_OFFLINE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_DECLINED_OFFLINE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_DECLINED:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_DECLINED" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_GO_ONLINE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_GO_ONLINE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CALL_YOUR_BANK:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CALL_YOUR_BANK" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_NOT_ACCEPTED:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_NOT_ACCEPTED" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_FALLBACK_TO_MSR:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_FALLBACK_TO_MSR" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_TIMEOUT:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_TIMEOUT" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_AUTHENTICATE_TRANSACTION:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_AUTHENTICATE_TRANSACTION" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_SWIPE_NON_ICC:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_SWIPE_NON_ICC" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CTLS_TWO_CARDS:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CTLS_TWO_CARDS" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CTLS_TERMINATE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CTLS_TERMINATE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER" + "\r\n");
            break;
    }
}

```

```

    }
}

SetOutputText(text);
}

private string tlvToValues(byte[] tlv)
{
    string text = "";
    Dictionary<string, string> dict = Common.processTLVUnencrypted(tlv);
    foreach (KeyValuePair<string, string> kvp in dict) text += kvp.Key + ": " + kvp.Value + "\r\n";
    return text;
}

delegate void SetTextCallback(string text);

private void SetOutputText(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (tbOutput.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetOutputText);
        Invoke(d, new object[] { text });
    }
    else
    {
        try { tbOutput.AppendText(text + "\r\n"); } catch (Exception ex) { }
    }
}

private void SetOutputTextLog(string text)
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (logOutput.InvokeRequired)
    {
        SetTextCallback d = new SetTextCallback(SetOutputTextLog);
        Invoke(d, new object[] { text });
    }
    else
    {
        try { logOutput.AppendText(text + "\r\n"); }
        catch (Exception ex)
        {
            System.Diagnostics.Debug.WriteLine("Exception: " + ex);
        }
    }
}

public static String GetTimestamp()
{
    DateTime value = DateTime.Now;
    return value.ToString("HH:mm:ss.fff");
}

```

## Chapter 6

# Core Implementation: UWP

IDTechSDK\_UWP.dll includes class libraries to interface with the SpectrumPro. This guide assumes a fair understanding of Visual Studio 2015+, C# and general Windows programming knowledge.

### 6.1 Integrating with IDTechSDK\_UWP.dll

- [Import the Necessary Libraries](#)
- [Add Device Details to the Application's Manifest](#)
- [Add Using Statements to Utilize Library](#)
- [Implement the Callback Function](#)
- [Initialize SpectrumPro](#)
- [Sample Project Tutorial](#)

### 6.2 Import the Necessary Libraries

Communicating with IDTech Devices requires the following library to be referenced by the project:

- IDTechSDK\_UWP.dll

Add the reference as you would any .NET managed library reference. The most direct method would be right-click on the "References" in the Solution Explorer for the project, select "Add Reference...", click "Browse..." and locate IDTechSDK\_UWP.dll.

IDTechSDK\_UWP.dll has a dependency of Microsoft .NET Core. Please make sure your final application installer checks for this dependency and installs it if it is not on the destination system.

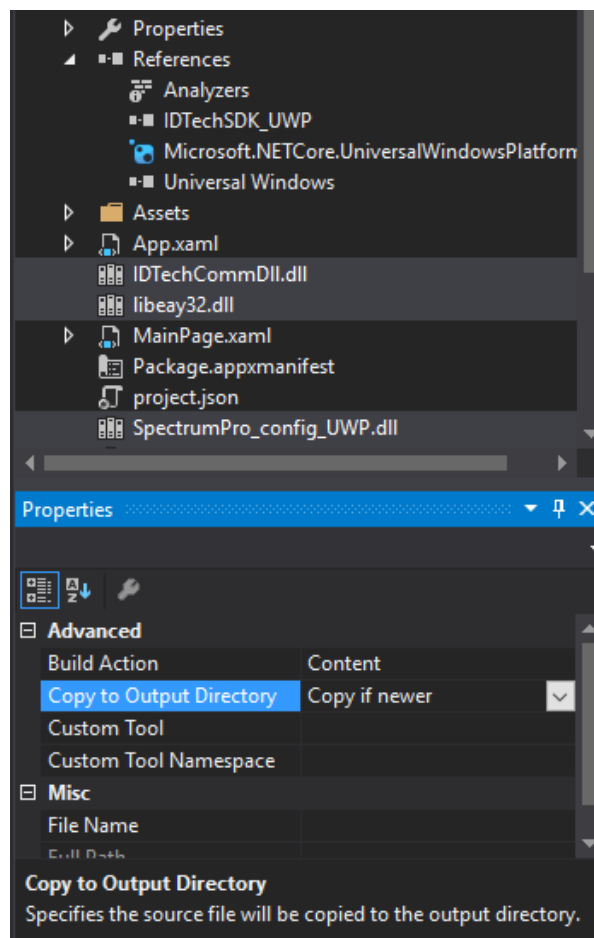
In addition, the following libraries need to be added to project folder and included in with the application distribution:

- IDTechCommDll.dll
- libeay32.dll
- ssleay.32.dll
- SpectrumPro\_config\_UWP.dll
- SpectrumPro\_device\_UWP.dll
- SpectrumPro\_emv\_UWP.dll

- SpectrumPro\_icc\_UWP.dll
- SpectrumPro\_msr\_UWP.dll
- SpectrumPro\_parse\_UWP.dll
- SpectrumPro\_pin\_UWP.dll

You can add these libraries by right-clicking on your project name in the solution Explorer and select "Add->Existing Item..." or keyboard shortcut Shift-Alt-A.

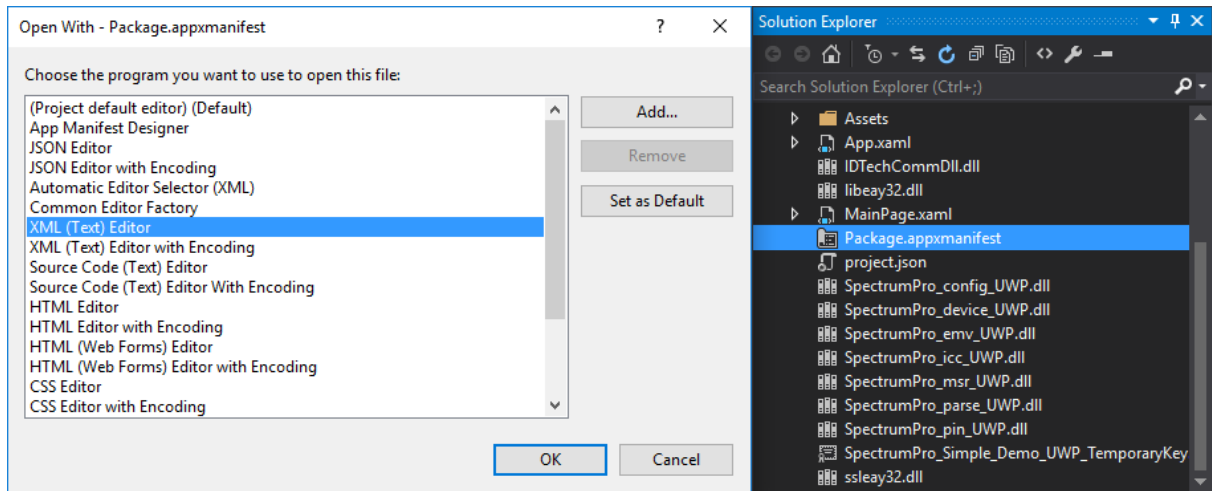
Once all three items are added, set their properties to "Copy if newer" so they will be included in the final applications destination folder.



### 6.3 Add Device Details to the Application's Manifest

In order for your application to recognize the device, the device's details must be added to the application's manifest file.

Open up your project's Package.appxmanifest file by right-clicking on it in the Solution Explorer, selecting "Open With...", and then choosing "XML (Text) Editor".



Search for the "Capabilities" tag and add the following code in between:

```
<!-- HID Devices -->
<DeviceCapability Name="humaninterfacedevice">
  <!-- SPECTRUM_PRO -->
  <Device Id="vidpid:0ACD 4010 usb">
    <Function Type="usage:0002 0026"/>
  </Device>
</DeviceCapability>

<!-- Serial Devices -->
<DeviceCapability Name="serialcommunication">
  <!-- Any serial device -->
  <Device Id="any">
    <Function Type="name:serialPort"/>
  </Device>
</DeviceCapability>
```

The aforementioned code contains the possible combinations of vendor IDs (VID), product IDs (PID), and connection types that a SpectrumPro device can have depending on its configuration.

The Package.appxmanifest file should now look similar to the following:

```
<?xml version="1.0" encoding="utf-8"?>

<Package
  xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10"
  xmlns:mp="http://schemas.microsoft.com/appx/2014/phone/manifest"
  xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10"
  IgnorableNamespaces="uap mp">

  <Identity
    Name="52cf1d7a-5c4c-4e36-b137-09eb02547462"
    Publisher="CN=daviddt"
    Version="1.0.0.0" />

  <mp:PhoneIdentity PhoneProductId="52cf1d7a-5c4c-4e36-b137-09eb02547462" PhonePublisherId="
    00000000-0000-0000-0000-000000000000"/>

  <Properties>
    <DisplayName>SpectrumPro_Simple_Demo_UWP</DisplayName>
    <PublisherDisplayName>daviddt</PublisherDisplayName>
    <Logo>Assets\StoreLogo.png</Logo>
  </Properties>

  <Dependencies>
    <TargetDeviceFamily Name="Windows.Universal" MinVersion="10.0.0.0" MaxVersionTested="10.0.0.0" />
  </Dependencies>

  <Resources>
    <Resource Language="x-generate"/>
```

```

</Resources>

<Applications>
  <Application Id="App"
    Executable="$targetnametoken$.exe"
    EntryPoint="SpectrumPro_Simple_Demo_UWP.App">
    <uap:VisualElements
      DisplayName="SpectrumPro_Simple_Demo_UWP"
      Square150x150Logo="Assets\Square150x150Logo.png"
      Square44x44Logo="Assets\Square44x44Logo.png"
      Description="SpectrumPro_Simple_Demo_UWP"
      BackgroundColor="transparent">
      <uap:DefaultTile Wide310x150Logo="Assets\Wide310x150Logo.png"/>
      <uap:SplashScreen Image="Assets\SplashScreen.png" />
    </uap:VisualElements>
  </Application>
</Applications>

<Capabilities>
  <Capability Name="internetClient" />

  <!-- HID Devices -->
  <DeviceCapability Name="humaninterfacedevice">
    <!-- SPECTRUM_PRO -->
    <Device Id="vidpid:0ACD 4010 usb">
      <Function Type="usage:0002 0026"/>
    </Device>
  </DeviceCapability>

  <!-- Serial Devices -->
  <DeviceCapability Name="serialcommunication">
    <!-- Any serial device -->
    <Device Id="any">
      <Function Type="name:serialPort"/>
    </Device>
  </DeviceCapability>
</Capabilities>
</Package>

```

## 6.4 Add Using Statements to Utilize Library

Add a line of code to the class that will utilize IDTechSDK\_UWP.dll at the start of the file:

```
using IDTechSDK;
```

## 6.5 Implement the Callback Function

There is a single method that will receive all callback information from the SDK. It uses DeviceState to determine which action to take.

```

private void MessageCallback(IDTechSDK.IDT_DEVICE_Types type, DeviceState state, byte[] data,
    IDTTransactionData cardData, EMV_Callback emvCallback, RETURN_CODE transactionResultCode)
{
    switch (state)
    {
        case DeviceState.ToConnect:
            //A connection attempt is starting for IDT_DEVICE_Types type
            break;
        case DeviceState.DefaultDeviceTypeChange:
            //The SDK is changing the default device to IDT_DEVICE_Types type
            break;
        case DeviceState.Connected:
            //A connection has been made to IDT_DEVICE_Types type

```

```

        break;
    case DeviceState.Disconnected:
        //A disconnection has occurred with IDT_DEVICE_TYPES type
        break;
    case DeviceState.ConnectionFailed:
        //A connection attempt has failed for IDT_DEVICE_TYPES type
        break;
    case DeviceState.TransactionData:
        //Transaction data is being returned in IDTTransactionData cardData
        break;
    case DeviceState.DataReceived:
        //Low-level data received for IDT_DEVICE_TYPES type
        break;
    case DeviceState.DataSent:
        //Low-level data sent for IDT_DEVICE_TYPES type
        break;
    case DeviceState.CommandTimeout:
        //Command timeout has occurred for IDT_DEVICE_TYPES type
        break;
    case DeviceState.ToSwipe:
        //Awaiting a swipe for IDT_DEVICE_TYPES type
        break;
    case DeviceState.MSRDecodeError:
        //Awaiting a swipe for IDT_DEVICE_TYPES type
        break;
    case DeviceState.ToTap:
        //Awaiting a contactless tap for IDT_DEVICE_TYPES type
        break;
    case DeviceState.SwipeTimeout:
        //Waiting for swipe timed out
        break;
    case DeviceState.TransactionCancelled:
        //Transaction has been cancelled
        break;
    case DeviceState.DeviceTimeout:
        //Waiting for EMV transaction to complete timed out
        break;
    case DeviceState.TransactionFailed:
        //Transaction failed to complete
        break;
    case DeviceState.DeviceCallback:
        //Callback during Device transaction retrieved from EMV_Callback emvCallback
        break;
    default:
        break;
}
}
}

```

## 6.6 Initialize SpectrumPro

A Singleton instance has been established in the IDT\_SpectrumPro class. Establish the callback, and then set a connection to SpectrumPro through either USB or Serial.

```

public MainPage()
{
    this.InitializeComponent();
    IDT_SpectrumPro.SetCallback(MessageCallback);

    //USB Connection:
    Task.Run(() => IDT_SpectrumPro.useUSB());

    //Serial Connection:
    //Task.Run(() => IDT_SpectrumPro.useSerialPort(1, false));
}

```

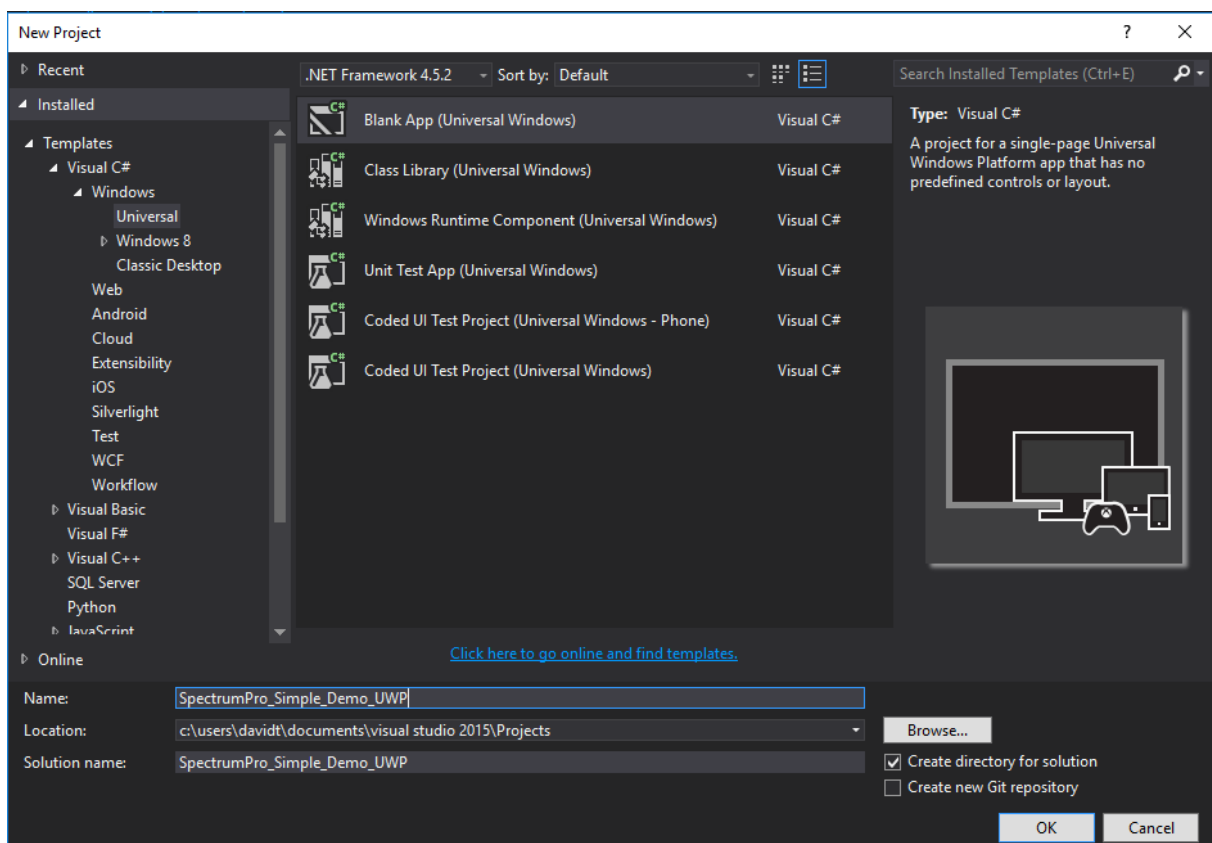
## 6.7 Sample Project Tutorial

Using Visual Studio 2015, we will create a C# sample project that will interface with the SpectrumPro and will perform the following activities:

- Select USB or COM interface
- Set UID and report device info
- Display ICC Status (card inserted/removed, ICC power on/off)
- Turn on/off ICC reader
- Get/Set terminal data file
- Get/Set AID for Visa
- Get/Set CAPKs for Visa
- Perform EMV Transaction
- Start/Stop a Swipe Request
- Show log of all data going to/from SpectrumPro

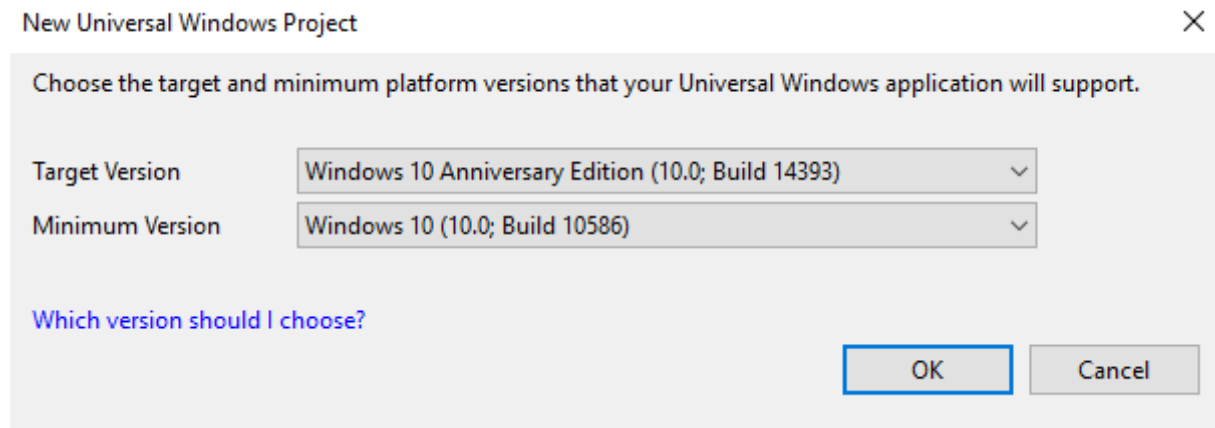
### 6.7.1 Step 1: Create New Project

Create a new Blank App (Universal Windows) in Visual Studio. In our example, we use project name SpectrumPro\_Simple\_Demo\_UWP.





On the following popup, choose the target and minimum versions that your application will support. We have chosen the following for our demo application:



### 6.7.2 Step 2: Import Libraries

[Import the Necessary Libraries](#)

### 6.7.3 Step 3: Add Device Details

[Add Device Details to the Application's Manifest](#)

### 6.7.4 Step 4: Design Interface

Use the Designer to layout buttons/fields

Open your designer and add items so it contains the following buttons/fields:

- Radio buttons for USB and COM selection. For COM selection, add a text box to specify COM port
- Add buttons to execute the following functions:
  - Set UID
  - Show ICC Status
  - Turn ON ICC Reader
  - Turn OFF ICC Reader
  - Set Terminal Data
  - Get Terminal Data
  - Set Visa AID
  - Get Visa AID
  - Set Visa CAPKs
  - List All CAPKs
  - Start EMV Transaction

- Cancel EMV Transaction
- Start Swipe
- Cancel Swipe
- Add a text box to communicate data from the SpectrumPro.
- Add a text box for the log of SpectrumPro.
- Add a text box to display the connection status.



### 6.7.5 Step 5: Configure the Project File

In the start of the class file, perform the following:

- [Add Using Statements to Utilize Library](#)
- Set callback method and initialize SpectrumPro singleton object in the class initializer. Reference: [Initialize SpectrumPro](#)
- Implement the radio button methods Click handlers to set the proper interface for SDK communications

```
private void rbUSB_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() => IDT_SpectrumPro.useUSB());
}

private void rbSerial_Click(object sender, RoutedEventArgs e)
{
    Int32 portNumber = Convert.ToInt32(tbCOMPort.Text);

    if (comboBoxBaudRate.SelectedIndex == 0)
    {
        Task.Run(() => IDT_SpectrumPro.useSerialPort(portNumber, false));
    }
    else
    {
        Int32 baudRate = Convert.ToInt32(comboBoxBaudRate.SelectedItem);

        Task.Run(() => IDT_SpectrumPro.useSerialPort(portNumber, baudRate, false));
    }
}
```

- Implement the button methods Click handlers for button press code execution

```
private void btnSetUID_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        SpectrumInfo info = new SpectrumInfo();
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.device_setUID(IDT_Device.Spectrum_UID, 2, ref
info);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("UID set Successful\r\nFirmware: " + info.firmwareVersion + "\r\nApplication: " +
info.applicationVersion + "\r\nMSR: " + info.msrHead + "\r\nHardware: " + info.hardwareVersion + "\r\n
Serial Number: " + info.serialNumber + "\r\nKey Mode: " + info.keyType.ToString() + "\r\n");
            System.Diagnostics.Debug.WriteLine("UID set Successful\r\nFirmware: " + info.
firmwareVersion + "\r\nApplication: " + info.applicationVersion + "\r\nMSR: " + info.msrHead + "\r\nHardware: " +
hardwareVersion + "\r\nSerial Number: " + info.serialNumber + "\r\nKey Mode: " + info.keyType.ToString());
        }
        else
        {
            SetOutputText("UID set failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + ": "
+ IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("UID set failed Error Code: " + "0x" + String.Format("
{0:X}", (ushort)rt));
        }
    });
}

private void btnSetTerminalData_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        byte[] term = Common.getByteArray("
5f3601029f1a0208409f3501229f330360f8c89f4005f00000a0019f1e0854657
26d696e616c9f150212349f160f3030303030303030303030303030309f1c0838373635343332319f4e2231303732312057616c6b657
22053742e20437970726573732c204341202c5553412edf260101df1008656e667265737a68df110101df270100dfee150101dfee160
100dfee170107dfee180180dfee1e08f0dc3cf0c29e9400dfee1f0180dfee1b083030303135313030dfee20013cdfee21010adfee2203323c
RETURN_CODE rt = IDT_SpectrumPro.SharedController.emw_setTerminalData(term);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Set Terminal Successful:" + " \r\n");
            System.Diagnostics.Debug.WriteLine("Set Terminal Successful");
        }
        else
        {
            SetOutputText("Save Terminal failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) +
": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Save Terminal failed Error Code: " + "0x" + String.
Format("{0:X}", (ushort)rt));
        }
    });
}

private void btnGetTerminalData_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        byte[] tlv = null;
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.emw_retrieveTerminalData(ref tlv);
```



```

a0000000039601017616e9ac8be014af88calla8fb17967b7394030e000000038000b745
86d19a207be6627c5b0aafbc44a2ecf5a2942d3a26ce19c4ffae920521868922e893e7838225a3947a2614796fb2c0628ce8c11e38
25a56d3b1bbaef783a5c6a81f36f8625395126fa983c5216d3166d48acde8a431212ff763a7f79d9edb7fed76b485de45beb829a3d4730848a
    rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
    capk = Common.getBytesArray("
a000000003570101251a5f5de61cf28b5c6e2b5807c0644a01d46ff5000100016000942b
7f2ba5ea307312b63df77c5243618acc2002bd7ecb74d821fe7bdc78bf28f49f74190ad9b23b9713b140ffec1fb429d93f56bdc7ade4ac075
    rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
    capk = Common.getBytesArray("
a000000003580101753ed0aa23e4cd5abd69eae7904b684a34a57c2200010001c8009955
2c4a1ecd68a0260157fc4151b5992837445d3fc57365ca5692c87be358cdcdf2c92fb6837522842a48eb11cdf2fd91770c7221e4af
6207c2de4004c7deeb6276dc62d52a87d2cd01fbf2dc4065db52824d2a2167a06d19e6a0f781071cdb2dd314cb94441d8dc0e936317
b77bf06f5177f6c5aba3a3bc6aa30209c97260b7a1ad3a192c9b8cd1d153570afcc87c3cd681d13e997fe33b3963a0a1c79772acf991033e1
    rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
    capk = Common.getBytesArray("
a00000000354010106960618791a86d387301edd4a3baf2d34fef1b400010001f800c6dd
c0b7645f7f16286ab7e4116655f56dd0c944766040dc68664dd973bd3bfd4c525bcbb95272b6b3ad9ba8860303ad08d9e8cc344a4070
f4c9fb9eeaf29c8a3460850c264cda39bbe3a7e7d08a69c31b5c8dd9f94ddbc9265758c0e7399adcf4362caee458d414c52b498274881
b196dacca7273f687f2a65faeb809d4b2ac1d3d1efb4f6490322318bd296d153b307a3283ab4e5be6ebd910359a8565eb9c4360d24ba
aca3dbfe393f3d6c830d603c6fc1e83409dfcd80d3a33ba243813bbb4ceaf9cbab6b74b00116f72ab278a88a011d70071e06cab140646438d
    rt = IDT_SpectrumPro.SharedController.emv_setCAPK(capk);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        SetOutputText("Load Visa CAPK Successful:" + " \r\n");
        System.Diagnostics.Debug.WriteLine("Load Visa CAPK Successful");
    }
    else
    {
        SetOutputText("Load Visa CAPK failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
+ " " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Load Visa CAPK failed Error Code: " + "0x" + String.
Format("{0:X}", (ushort)rt));
    }
});
}

private void btnListCAPKs_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        byte[] capk = null;
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_retrieveCAPKList(ref capk);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("List CAPK Successful \r\n");
            System.Diagnostics.Debug.WriteLine("List CAPK Successful");
            if (capk.Length > 0)
            {
                for (int x = 0; x < capk.Length; x = x + 6)
                {
                    byte[] thecapk = new byte[] { capk[x], capk[x + 1], capk[x + 2], capk[x + 3], capk[x +
4], capk[x + 5] };
                    SetOutputText(Common.getHexStringFromBytes(thecapk) + " \r\n");
                }
            }
        }
        else
        {
            SetOutputText("List CAPK failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + ":
" + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("List CAPK failed Error Code: " + "0x" + String.Format
("{0:X}", (ushort)rt));
        }
    });
}

private void btnStartEMVTransaction_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_startTransaction(1.00, 0, 2, 0, 30, null,
false);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Start EMV Successful\r\n");
            System.Diagnostics.Debug.WriteLine("Start EMV Successful");
        }
        else
        {
            SetOutputText("Start EMV failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + ":
" + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Start EMV failed Error Code: " + "0x" + String.Format
("{0:X}", (ushort)rt));
        }
    });
}
}

```

```

private void btnCancelEMVTransaction_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.emv_cancelTransaction();
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Cancel Transaction Successful\r\n");
            System.Diagnostics.Debug.WriteLine("Cancel Transaction Successful");
        }
        else
        {
            SetOutputText("Cancel Transaction failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)
rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Cancel Transaction failed Error Code: " + "0x" +
String.Format("{0:X}", (ushort)rt));
        }
    });
}

private void btnStartSwipe_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.msr_startMSRSwipe(60);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("MSR Turned On successfully; Ready to swipe\r\n");
            System.Diagnostics.Debug.WriteLine("MSR Turned On successfully; Ready to swipe");
        }
        else
        {
            SetOutputText("MSR Turned On failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) +
": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("MSR Turned On failed Error Code: " + "0x" + String.
Format("{0:X}", (ushort)rt));
        }
    });
}

private void btnCancelSwipe_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_SpectrumPro.SharedController.msr_cancelMSRSwipe();
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("MSR Turned Off successfully\r\n");
            System.Diagnostics.Debug.WriteLine("MSR Turned Off successfully");
        }
        else
        {
            SetOutputText("MSR Turned Off failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
+ ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("MSR Turned Off failed Error Code: " + "0x" + String.
Format("{0:X}", (ushort)rt));
        }
    });
}

```

Note that all the the click handlers' code are ran asynchronously. At least the device command needs to be executed asynchronously or it will block the UI thread. The UWP API that is used by the IDTechSDK\_UWP library for communication with HID devices is asynchronous.

## 6.7.6 Step 6: Configure Callback

The callback is used to receive important SDK data (messages, log info and transaction results). Reference: [Implement the Callback Function](#)

```

private void MessageCallBack(IDTechSDK.IDT_DEVICE_Types type, DeviceState state, byte[] data,
    IDTTransactionData cardData, EMV_Callback emvCallback, RETURN_CODE transactionResultCode)
{
    switch (state)
    {
        case DeviceState.ToConnect:
            //A connection attempt is starting for IDT_DEVICE_TYPES type
            SetOutputText("Callback:ToConnect\n");
    }
}

```

```

        break;
    case DeviceState.DefaultDeviceTypeChange:
        //The SDK is changing the default device to IDT_DEVICE_TYPES type
        SetOutputText("Callback:DefaultDeviceTypeChange\n");
        connectionStatus.Text = "SpectrumPro connected.";
        connectionStatus.Background = new SolidColorBrush(Windows.UI.Colors.Green);
        break;
    case DeviceState.Connected:
        //A connection has been made to IDT_DEVICE_TYPES type
        SetOutputText("Callback:Connected\n");
        connectionStatus.Text = "SpectrumPro connected.";
        connectionStatus.Background = new SolidColorBrush(Windows.UI.Colors.Green);
        break;
    case DeviceState.Disconnected:
        //A disconnection has occurred with IDT_DEVICE_TYPES type
        SetOutputText("Callback:Disconnected\n");
        connectionStatus.Text = "SpectrumPro not connected.";
        connectionStatus.Background = new SolidColorBrush(Windows.UI.Colors.Red);
        break;
    case DeviceState.ConnectionFailed:
        //A connection attempt has failed for IDT_DEVICE_TYPES type
        SetOutputText("Callback:ConnectionFailed\n");
        break;
    case DeviceState.TransactionData:
        //Transaction data is being returned in IDTTransactionData cardData
        SetOutputText("Callback:TransactionData\n");
        if (cardData.emv_resultCode == EMV_RESULT_CODE.EMV_RESULT_CODE_GO_ONLINE)
        {
            //auto complete. Normally, a host response is required here
            SetOutputText("Online request. Auto Complete EMV Transaction.\n");
            byte[] responseCode = new byte[] { 0x30, 0x30 };
            byte[] iad = new byte[] { 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x30, 0x30 };
            Task.Run(() => IDT_SpectrumPro.SharedController.emv_completeTransaction(false, responseCode
, iad, null, null));
            return;
        }
        displayCardData(cardData);
        break;
    case DeviceState.DataReceived:
        //Low-level data received for IDT_DEVICE_TYPES type
        SetOutputTextLog(GetTimestamp() + " IN: " + Common.getHexStringFromBytes(data));
        break;
    case DeviceState.DataSent:
        //Low-level data sent for IDT_DEVICE_TYPES type
        SetOutputTextLog(GetTimestamp() + " OUT: " + Common.getHexStringFromBytes(data));
        break;
    case DeviceState.CommandTimeout:
        SetOutputText("Callback:CommandTimeout\n");
        //Command timeout has occurred for IDT_DEVICE_TYPES type
        break;
    case DeviceState.ToSwipe:
        //Awaiting a swipe for IDT_DEVICE_TYPES type
        SetOutputText("Callback:ToSwipe\n");
        break;
    case DeviceState.MSRDecodeError:
        //Awaiting a swipe for IDT_DEVICE_TYPES type
        SetOutputText("Callback:MSRDecodeError\n");
        break;
    case DeviceState.ToTap:
        //Awaiting a contactless tap for IDT_DEVICE_TYPES type
        SetOutputText("Callback:ToTap\n");
        break;
    case DeviceState.SwipeTimeout:
        //Waiting for swipe timed out
        SetOutputText("Callback:SwipeTimeout\n");
        break;
    case DeviceState.TransactionCancelled:
        //Transaction has been cancelled
        SetOutputText("Callback:TransactionCancelled\n");
        break;
    case DeviceState.DeviceTimeout:
        //Waiting for EMV transaction to complete timed out
        SetOutputText("Callback:DeviceTimeout\n");
        break;
    case DeviceState.TransactionFailed:
        //Transaction failed to complete
        SetOutputText("Callback:TransactionFailed\n");
        break;
    case DeviceState.EMVCallback:
        //Callback during EMV transaction retrieved from EMV_Callback emvCallback
        SetOutputText("Callback:EMVCallback\n");
        if (emvCallback.callbackType == EMV_CALLBACK_TYPE.EMV_CALLBACK_TYPE_LCD) //LCD
        Callback Type
        {
            if (emvCallback.lcd_displayMode == EMV_LCD_DISPLAY_MODE.
EMV_LCD_DISPLAY_MODE_CLEAR_SCREEN)
            {

```

```

        SetOutputText("LCD Callback:Clear LCD Screen\n");
        return;
    }
    else if (emvCallback.lcd_displayMode == EMV_LCD_DISPLAY_MODE.
EMV_LCD_DISPLAY_MODE_MESSAGE)
    {
        // A message is requested to be displayed. See other test app included with SDK for a
        complete example.
        SetOutputText("LCD Callback:Display Message\n");
    }
    else
    {
        //Display message with menu/language or prompt, and return result to
        emv_callbackResponseLCD
        //Kernel will not proceed until this step is complete
        //seeing to default value of 1. See other test app included with SDK for a complete
        example
        SetOutputText("LCD Callback:Menu Display Request. Sending result 1\n");
        IDT_SpectrumPro.SharedController.emv_callbackResponseLCD(emvCallback.lcd_displayMode, 1
    );
    }
    }
    break;
    default:
        break;
}

public static String GetTimestamp()
{
    DateTime value = DateTime.Now;
    return value.ToString("HH:mm:ss.fff");
}

private void displayCardData(IDTTransactionData cardData)
{
    string text = "";
    if (cardData.Event == EVENT_TRANSACTION_DATA_Types.EVENT_TRANSACTION_PIN_DATA)
    {
        SetOutputText("PIN Data received:\r\nKSN: " + cardData.pin_KSN + "\r\nPINBLOCK: " + cardData.
        pin_pinblock + "\r\n");
        return;
    }
    if (cardData.Event == EVENT_TRANSACTION_DATA_Types.EVENT_TRANSACTION_DATA_CARD_DATA)
    {
        SetOutputText("Data received: (Length [" + cardData.msr_rawData.Length.ToString() + "])\n" + string
        .Concat(cardData.msr_rawData.ToArray().Select(b => b.ToString("X2")).ToArray()) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Data received: (Length [" + cardData.msr_rawData.Length.
        ToString() + "])\n" + string.Concat(cardData.msr_rawData.ToArray().Select(b => b.ToString("X2")).ToArray()));
    }
    if (cardData.device_RSN != null && cardData.device_RSN.Length > 0)
        text += "Serial Number: " + cardData.device_RSN + "\r\n";
    if (cardData.msr_track1Length > 0)
        text += "Track 1: " + cardData.msr_track1 + "\r\n";
    if (cardData.msr_encTrack1 != null)
        text += "Track 1 Encrypted: " + Common.getHexStringFromBytes(cardData.msr_encTrack1) + "\r\n";
    if (cardData.msr_hashTrack1 != null)
        text += "Track 1 Hash: " + Common.getHexStringFromBytes(cardData.msr_hashTrack1) + "\r\n";
    if (cardData.msr_track2Length > 0)
        text += "Track 2: " + cardData.msr_track2 + "\r\n";
    if (cardData.msr_encTrack2 != null)
        text += "Track 2 Encrypted: " + Common.getHexStringFromBytes(cardData.msr_encTrack2) + "\r\n";
    if (cardData.msr_hashTrack2 != null)
        text += "Track 2 Hash: " + Common.getHexStringFromBytes(cardData.msr_hashTrack2) + "\r\n";
    if (cardData.msr_track3Length > 0)
        text += "Track 3: " + cardData.msr_track3 + "\r\n";
    if (cardData.msr_encTrack3 != null)
        text += "Track 3 Encrypted: " + Common.getHexStringFromBytes(cardData.msr_encTrack3) + "\r\n";
    if (cardData.msr_hashTrack3 != null)
        text += "Track 3 Hash: " + Common.getHexStringFromBytes(cardData.msr_hashTrack3) + "\r\n";
    if (cardData.msr_KSN != null)
        text += "KSN: " + Common.getHexStringFromBytes(cardData.msr_KSN) + "\r\n";
    if (cardData.emv_clearingRecord != null)
    {
        if (cardData.emv_clearingRecord.Length > 0)
        {
            text += "\r\nCTLS Clearing Record: \r\n";
            text += Common.getHexStringFromBytes(cardData.emv_clearingRecord) + "\r\n";
            Dictionary<string, string> dict = Common.processTLVUnencrypted(cardData.emv_clearingRecord);
            foreach (KeyValuePair<string, string> kvp in dict) text += kvp.Key + ": " + kvp.Value + "\r\n";
            text += "\r\n\r\n";
        }
    }
    if (cardData.emv_unencryptedTags != null)
    {
        if (cardData.emv_unencryptedTags.Length > 0)
        {

```



```

        text += "\r\n===== \r\n";
        text += "\r\nUnencrypted Tags: \r\n";
        text += Common.getHexStringFromBytes(cardData.emv_unencryptedTags) + "\r\n\r\n";
        text += tlvToValues(cardData.emv_unencryptedTags);
        text += "\r\n===== \r\n";
    }
}
if (cardData.emv_encryptedTags != null)
{
    if (cardData.emv_encryptedTags.Length > 0)
    {
        text += "\r\n===== \r\n";
        text += "\r\nEncrypted Tags: \r\n";
        text += Common.getHexStringFromBytes(cardData.emv_encryptedTags) + "\r\n\r\n";
        text += tlvToValues(cardData.emv_encryptedTags);
        text += "\r\n===== \r\n";
    }
}
if (cardData.emv_maskedTags != null)
{
    if (cardData.emv_maskedTags.Length > 0)
    {
        text += "\r\n===== \r\n";
        text += "\r\nMasked Tags: \r\n";
        text += Common.getHexStringFromBytes(cardData.emv_maskedTags) + "\r\n\r\n";
        text += tlvToValues(cardData.emv_maskedTags);
        text += "\r\n===== \r\n";
    }
}
text += "ICC Present: ";
text += (cardData.iccPresent == 1 ? "TRUE" : "FALSE") + "\r\n";
text += "is CTLS: ";
text += (cardData.isCTLS == 1 ? "TRUE" : "FALSE") + "\r\n";
if (cardData.Event == EVENT_TRANSACTION_DATA.Types.EVENT_TRANSACTION_DATA_EMV_DATA)
{
    if (cardData.isCTLS == 2) text += "Capture Encrypt Type: " + ((cardData.emv_encryptionMode ==
EMV_ENCRYPTION_MODE.EMV_ENCRYPTION_MODE_TDES) ? "TDES" : "AES") + "\r\n";
    switch (cardData.emv_resultCode)
    {
        case EMV_RESULT_CODE.EMV_RESULT_CODE_APPROVED:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_APPROVED" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_APPROVED_OFFLINE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_APPROVED_OFFLINE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_DECLINED_OFFLINE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_DECLINED_OFFLINE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_DECLINED:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_DECLINED" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_GO_ONLINE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_GO_ONLINE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CALL_YOUR_BANK:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CALL_YOUR_BANK" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_NOT_ACCEPTED:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_NOT_ACCEPTED" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_FALLBACK_TO_MSR:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_FALLBACK_TO_MSR" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_TIMEOUT:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_TIMEOUT" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_AUTHENTICATE_TRANSACTION:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_AUTHENTICATE_TRANSACTION" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_SWIPE_NON_ICC:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_SWIPE_NON_ICC" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CTLS_TWO_CARDS:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CTLS_TWO_CARDS" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CTLS_TERMINATE:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CTLS_TERMINATE" + "\r\n");
            break;
        case EMV_RESULT_CODE.EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER:
            text += ("EMV RESULT: " + "EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER" + "\r\n");
            break;
    }
}
SetOutputText(text);
}

private string tlvToValues(byte[] tlv)

```

```
{
    string text = "";
    Dictionary<string, string> dict = Common.processTLVUnencrypted(tlv);
    foreach (KeyValuePair<string, string> kvp in dict) text += kvp.Key + ": " + kvp.Value + "\r\n";
    return text;
}

delegate void SetTextCallback(string text);

private async void SetOutputText(string text)
{
    await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, () =>
    {
        tbOutput.Text = text + "\r\n" + tbOutput.Text;
    });
}

private void SetOutputTextLog(string text)
{
    logOutput.Text = text + "\r\n" + logOutput.Text;
}
```

## Chapter 7

# LCD Foreign Language Mapping Table

ID	Message ID	English	French	Spanish	Chinese
0	MSG_NULL	-	-	-	-
1	MSG_AMOUNT	AMOUNT	MONTANT	CANTIDAD	金
2	MSG_AMOUNT_↔ OK	AMOUNT OK?	MONTANT OK	MONTO CORRE↔ CTO?	确定金
3	MSG_APPROVED	APPROVED	APPROUVE	APROVADO	通
4	MSG_CALL_YO↔ UR_BANK	CALL YOUR BANK	APPE VOTRE B↔ ANQE	LLAME A SU BA↔ NCO	系您的行
5	MSG_CANCEL_↔ OR_ENTER	CANCEL OR EN↔ TER	ANNULE OU EN↔ TRER	CANCEL O ENT↔ RAR	取消或确定
6	MSG_CARD_ER↔ ROR	CARD ERROR	ERREUR CARTE	ERROR DE TAR↔ JETA	卡
7	MSG_DECLINED	DECLINED	REFUSE	DECLINADO	卡被拒
8	MSG_ENTER_A↔ MOUNT	ENTER AMOUNT	ENTRER MONT↔ ANT	INGRESE MONTO	入金
9	MSG_ENTER_PIN	ENTER PIN:	ENTRER PIN:	ENTRAR NPI:	入密
10	MSG_INCORRE↔ CT_PIN	INCORRECT PIN	NIP INCORRECT	NPI INCORRECTO	密
11	MSG_ICC_MSR1	SWIPE OR INSE↔ RT	PASSER OU INS↔ ERT	MOVER O INSERT	刷卡或插卡
12	MSG_ICC_MSR2	CARD	CARTE	TARJETA	卡
13	MSG_INSERT_↔ CARD	INSERT CARD	INSERT CARTE	INSERTAR TAR↔ JETA	插卡
14	MSG_USE_CHI↔ P_READER	USE CHIP READ↔ ER UTI	LECTEUR CHIP	USO CHIP LECT↔ OR	使用芯片卡
15	MSG_NOT_ACC↔ EPTED	NOT ACCEPTED	PAS ACCEPTE	DENEGADO	法接受
16	MSG_PIN_OK	GET PIN OK	-	-	密正确
17	MSG_PLEASE_↔ WAIT	PLEASE WAIT...	ATTENDRE...	POR FAVOR ES↔ PERE	等候中
18	MSG_PROCESS↔ ING_ERROR	PROCESSING E↔ RROR	ERREUR DE TR↔ AITE	ERROR PROCE↔ SANDO	理
19	MSG_USE_MAG↔ STRIPE	USE MAGSTRIPE	USAGE MAGST↔ RIPE	USO DE MAGST↔ RIPE	使用磁卡
20	MSG_TRY_AGAIN	TRY AGAIN	REESSAYER	VUELV INTENTA↔ RLO	重
21	MSG_ONLINE	GO ONLINE	GO LIGNE	GO LINEA	在

ID	Message ID	English	French	Spanish	Chinese
22	MSG_TRANSACTION_ERROR↔	TRANSACTION ERR	ERREUR DE TRAN↔	ERROR DE TRA↔NSAC	交易
23	MSG_TERMINATE	TERMINATE	RESILIER	TERMINAR	止
24	MSG_ADVICE	ADVICE	CONSEILS	CONSEJOS	建
25	MSG_TIMEOUT	TIME OUT	TIMEOUT	TIEMPO DE ESP↔ERA	超
26	MSG_PROCESSING↔	PROCESSING...	PROCESSUS...	PROCESANDO...	理中。。。
27	MSG_PIN_TRY_EX↔	PIN TRY LIMIT EX	PIN TRY DEPAS↔SE	TRY PIN SUPER↔ADA	密次多
28	MSG_ISSUER_AUTH_FAIL↔	ISSUER AUTH FAIL	EMETTEUR FAIL	EMISOR FALLA	与卡机构
29	MSG_CONTINUE_PROCESS↔	CONTINUE PROCESS	CONTINUER LA	CONTINUAR PR↔OCES	理
30	MSG_GET_PIN_ERROR↔	GET PIN ERROR	GET PIN ERROR	OBTENER PIN E↔RR	密
31	MSG_GET_PIN_FAIL↔	GET PIN FAIL	GET PIN FAIL	OBTENER PIN F↔ALL	取密
32	MSG_NOKEY_GET_PIN↔	NO KEY GET PIN	NO KEY GET PIN	NO CLAVE GET PIN	法入密
33	MSG_CANCELLED↔	CANCELLED	ANNULE	CANCELADO	取消
34	MSG_LAST_PIN_TRY↔	LAST PIN TRY	-	-	最后一次入密
35	MSG_REMOVE_CARD↔	REMOVE CARD	RETIRER LA CA↔RTE	RETIRE LA TAR↔JETA	取卡

## Chapter 8

# Error Code Reference

```
public static string getErrorString(RETURN_CODE code)
{
    switch ((int)code)
    {
        case 0: return "no error, beginning task";
        case 1: return "no response from reader";
        case 2: return "invalid response data";
        case 3: return "time out for task or CMD";
        case 4: return "wrong parameter";
        case 5: return "SDK is doing MSR or ICC task";
        case 6: return "SDK is doing PINPad task";
        case 7: return "SDK is doing CTLS task";
        case 8: return "SDK is doing EMV task";
        case 9: return "SDK is doing Other task";
        case 10: return "err response or data";
        case 11: return "no reader attached";
        case 12: return "mono audio is enabled";
        case 13: return "did connection";
        case 14: return "audio volume is too low";
        case 15: return "task or CMD be canceled";
        case 16: return "UF wrong string format";
        case 17: return "UF file not found";
        case 18: return "UF wrong file format";
        case 19: return "Attempt to contact online host failed";
        case 20: return "Attempt to perform RKI failed";
        case 0x300: return "Key Type(TDES) of Session Key is not same as the related Master Key.";
        case 0x400: return "Related Key was not loaded.";
        case 0x500: return "Key Same.";
        case 0x501: return "Key is all zero";
        case 0x502: return "TR-31 format error";
        case 0x702: return "PAN is Error Key.";
        case 0x705: return "No Internal MSR PAN (or Internal MSR PAN is erased timeout)";
        case 0X0C01: return "Incorrect Frame Tag";
        case 0X0C02: return "Incorrect Frame Type";
        case 0X0C03: return "Unknown Frame Type";
        case 0X0C04: return "Unknown Command";
        case 0X0C05: return "Unknown Sub-Command";
        case 0X0C06: return "CRC Error";
        case 0X0C07: return "Failed";
        case 0X0C08: return "Timeout";
        case 0X0C0A: return "Incorrect Parameter";
        case 0X0C0B: return "Command Not Supported";
        case 0X0C0C: return "Sub-Command Not Supported";
        case 0X0C0D: return "Parameter Not Supported / Status Abort Command";
        case 0X0C0F: return "Sub-Command Not Allowed";
        case 0X0D01: return "Incorrect Header Tag";
        case 0X0D02: return "Unknown Command";
        case 0X0D03: return "Unknown Sub-Command";
        case 0X0D04: return "CRC Error in Frame";
        case 0X0D05: return "Incorrect Parameter";
        case 0X0D06: return "Parameter Not Supported";
        case 0X0D07: return "Mal-formatted Data";
        case 0X0D08: return "Timeout";
        case 0X0D0A: return "Failed / NACK";
        case 0X0D0B: return "Command not Allowed";
        case 0X0D0C: return "Sub-Command not Allowed";
        case 0X0D0D: return "Buffer Overflow (Data Length too large for reader buffer)";
        case 0X0D0E: return "User Interface Event";
        case 0X0D11: return "Communication type not supported, VT-1, burst, etc.";
    }
}
```

```

    case 0X0D12: return "Secure interface is not functional or is in an intermediate state.";
    case 0X0D13: return "Data field is not mod 8";
    case 0X0D14: return "Pad 0x80 not found where expected";
    case 0X0D15: return "Specified key type is invalid";
    case 0X0D1: return "Could not retrieve key from the SAM(InitSecureComm)";
    case 0X0D17: return "Hash code problem";
    case 0X0D18: return "Could not store the key into the SAM(InstallKey)";
    case 0X0D19: return "Frame is too large";
    case 0X0D1A: return "Unit powered up in authentication state but POS must resend the
InitSecureComm command";
    case 0X0D1B: return "The EEPROM may not be initialized because SecCommInterface does not
make sense";
    case 0X0D1C: return "Problem encoding APDU";
    case 0X0D20: return "Unsupported Index(ILM) SAM Transceiver error - problem communicating
with the SAM(Key Mgr)";
    case 0X0D2: return "Unexpected Sequence Counter in multiple frames for single bitmap(ILM)
Length error in data returned from the SAM(Key Mgr)";
    case 0X0D22: return "Improper bit map(ILM)";
    case 0X0D23: return "Request Online Authorization";
    case 0X0D24: return "ViVOCard3 raw data read successful";
    case 0X0D25: return "Message index not available(ILM) ViVOComm activate transaction card
type (ViVOComm)";
    case 0X0D26: return "Version Information Mismatch(ILM)";
    case 0X0D27: return "Not sending commands in correct index message index(ILM)";
    case 0X0D28: return "Time out or next expected message not received(ILM)";
    case 0X0D29: return "ILM languages not available for viewing(ILM)";
    case 0X0D2A: return "Other language not supported(ILM)";
    case 0X0D41: return "Unknown Error from SAM";
    case 0X0D42: return "Invalid data detected by SAM";
    case 0X0D43: return "Incomplete data detected by SAM";
    case 0X0D44: return "Reserved";
    case 0X0D45: return "Invalid key hash algorithm";
    case 0X0D46: return "Invalid key encryption algorithm";
    case 0X0D47: return "Invalid modulus length";
    case 0X0D48: return "Invalid exponent";
    case 0X0D49: return "Key already exists";
    case 0X0D4A: return "No space for new RID";
    case 0X0D4B: return "Key not found";
    case 0X0D4C: return "Crypto not responding";
    case 0X0D4D: return "Crypto communication error";
    case 0X0D4E: return "Module-specific error for Key Manager";
    case 0X0D4F: return "All key slots are full (maximum number of keys has been installed)";
    case 0X0D50: return "Auto-Switch OK";
    case 0X0D51: return "Auto-Switch failed";
    case 0X0D90: return "Account DUKPT Key not exist";
    case 0X0D91: return "Account DUKPT Key KSN exhausted";
    case 0x0D00: return "This Key had been loaded.";
    case 0x0E00: return "Base Time was loaded.";
    case 0x0F00: return "Encryption Or Decryption Failed.";
    case 0x1000: return "Battery Low Warning (It is High Priority Response while Battery is
Low.)";
    case 0x1800: return "Send \"Cancel Command\" after send \"Get Encrypted PIN\" & \"Get Numeric \"&
\"Get Amount\"";
    case 0x1900: return "Press \"Cancel\" key after send \"Get Encrypted PIN\" & \"Get Numeric \"&
\"Get Amount\"";
    case 0x30FF: return "Security Chip is not connect";
    case 0x3000: return "Security Chip is deactivation & Device is In Removal Legally State.";
    case 0x3101: return "Security Chip is activation & Device is In Removal Legally State.";
    case 0x5500: return "No Admin DUKPT Key.";
    case 0x5501: return "Admin DUKPT Key STOP.";
    case 0x5502: return "Admin DUKPT Key KSN is Error.";
    case 0x5503: return "Get Authentication Code1 Failed.";
    case 0x5504: return "Validate Authentication Code Error.";
    case 0x5505: return "Encrypt or Decrypt data failed.";
    case 0x5506: return "Not Support the New Key Type.";
    case 0x5507: return "New Key Index is Error.";
    case 0x5508: return "Step Error.";
    case 0x5509: return "KSN Error";
    case 0x550A: return "MAC Error.";
    case 0x550B: return "Key Usage Error.";
    case 0x550C: return "Mode Of Use Error.";
    case 0x550F: return "Other Error.";
    case 0x6000: return "Save or Config Failed / Or Read Config Error.";
    case 0x6200: return "No Serial Number.";
    case 0x6900: return "Invalid Command - Protocol is right, but task ID is invalid.";
    case 0x6A01: return "Unsupported Command - Protocol and task ID are right, but command is
invalid - In this State";
    case 0x6A00: return "Unsupported Command - Protocol and task ID are right, but command is
invalid.";
    case 0x6B00: return "Unknown parameter in command - Protocol task ID and command are right,
but parameter
is invalid.";
    case 0x6C00: return "Unknown parameter in command - Protocol task ID and command are right,
but length is
out of the requirement.";
    case 0x7200: return "Device is suspend (MKSK suspend or press password suspend).";
    case 0x7300: return "PIN DUKPT is STOP (21 bit 1).";
    case 0x7400: return "Device is Busy.";
    case 0xE100: return "Can not enter sleep mode";

```

```

case 0xE200: return "File has existed";
case 0xE300: return "File has not existed";
case 0xE313: return "IO line low -- Card error after session start";
case 0xE400: return "Open File Error";
case 0xE500: return "SmartCard Error";
case 0xE600: return "Get MSR Card data is error";
case 0xE700: return "Command time out";
case 0xE800: return "File read or write is error";
case 0xE900: return "Active 1850 error!";
case 0xEA00: return "Load bootloader error";
case 0xEF00: return "Protocol Error- STX or ETX or check error.";
case 0xEB00: return "Picture is not exist";
case 0x2C02: return "No Microprocessor ICC seated";
case 0x2C06: return "no card seated to request ATR";
case 0x2D01: return "Card Not Supported,";
case 0x2D03: return "Card Not Supported, wants CRC";
case 0x690D: return "Command not supported on reader without ICC support";
case 0x8100: return "ICC error time out on power-up";
case 0x8200: return "invalid TS character received - Wrong operation step";
case 0x8300: return "Decode MSR Error";
case 0x8400: return "TriMagII no Response";
case 0x8500: return "No Swipe MSR Card";
case 0x8510: return "No Financial Card";
case 0x8600: return "Unsupported F, D, or combination of F and D";
case 0x8700: return "protocol not supported EMV TD1 out of range";
case 0x8800: return "power not at proper level";
case 0x8900: return "ATR length too long";
case 0x8B01: return "EMV invalid TA1 byte value";
case 0x8B02: return "EMV TB1 required";
case 0x8B03: return "EMV Unsupported TB1 only 00 allowed";
case 0x8B04: return "EMV Card Error, invalid BWI or CWI";
case 0x8B06: return "EMV TB2 not allowed in ATR";
case 0x8B07: return "EMV TC2 out of range";
case 0x8B08: return "EMV TC2 out of range";
case 0x8B09: return "per EMV96 TA3 must be > 0xF";
case 0x8B10: return "ICC error on power-up";
case 0x8B11: return "EMV T=1 then TB3 required";
case 0x8B12: return "Card Error, invalid BWI or CWI";
case 0x8B13: return "Card Error, invalid BWI or CWI";
case 0x8B17: return "EMV TC1/TB3 conflict*";
case 0x8B20: return "EMV TD2 out of range must be T=1";
case 0x8C00: return "TCK error";
case 0xA304: return "connector has no voltage setting";
case 0xA305: return "ICC error on power-up invalid (SBLK(IFSD) exchange";
case 0xE301: return "ICC error after session start";
case 0xFF00: return "Request to go online";
case 0xFF01: return "EMV: Accept the offline transaction";
case 0xFF02: return "EMV: Decline the offline transaction";
case 0xFF03: return "EMV: Accept the online transaction";
case 0xFF04: return "EMV: Decline the online transaction";
case 0xFF05: return "EMV: Application may fallback to magstripe technology";
case 0xFF06: return "EMV: ICC detected that the conditions of use are not satisfied";
case 0xFF07: return "EMV: ICC didn't accept transaction";
case 0xFF08: return "EMV: Transaction was cancelled";
case 0xFF09: return "EMV: Application was not selected by kernel or ICC format error or ICC
missing data error";
case 0xFF0A: return "EMV: Transaction is terminated";
case 0xFF0B: return "EMV: Other EMV Error";
case 0xFFFF: return "NO RESPONSE";
case 0xF002: return "ICC communication timeout";
case 0xF003: return "ICC communication Error";
case 0xF00F: return "ICC Card Seated and Highest Priority, disable MSR work request";
case 0xF200: return "AID List / Application Data is not exist";
case 0xF201: return "Terminal Data is not exist";
case 0xF202: return "TLV format is error";
case 0xF203: return "AID List is full";
case 0xF204: return "Any CA Key is not exist";
case 0xF205: return "CA Key RID is not exist";
case 0xF206: return "CA Key Index it not exist";
case 0xF207: return "CA Key is full";
case 0xF208: return "CA Key Hash Value is Error";
case 0xF209: return "Transaction format error";
case 0xF20A: return "The command will not be processing";
case 0xF20B: return "CRL is not exist";
case 0xF20C: return "CRL number exceed max number";
case 0xF20D: return "Amount, Other Amount, Transaction Type are missing";
case 0xF20E: return "The Identification of algorithm is mistake";
case 0xF20F: return "No Financial Card";
case 0xF210: return "In Encrypt Result state, TLV total Length is greater than Max Length";
case 0x1001: return "INVALID ARG";
case 0x1002: return "FILE_OPEN_FAILED";
case 0x1003: return "FILE_OPERATION_FAILED";
case 0x2001: return "MEMORY_NOT_ENOUGH";
case 0x3002: return "SMARTCARD_FAIL";
case 0x3003: return "SMARTCARD_INIT_FAILED";
case 0x3004: return "FALLBACK_SITUATION";
case 0x3005: return "SMARTCARD_ABSENT";

```

```

case 0x3006: return "SMARTCARD_TIMEOUT";
case 0x5001: return "EMV_PARSING_TAGS_FAILED";
case 0x5002: return "EMV_DUPLICATE_CARD_DATA_ELEMENT";
case 0x5003: return "EMV_DATA_FORMAT_INCORRECT";
case 0x5004: return "EMV_NO_TERM_APP";
case 0x5005: return "EMV_NO_MATCHING_APP";
case 0x5006: return "EMV_MISSING_MANDATORY_OBJECT";
case 0x5007: return "EMV_APP_SELECTION_RETRY";
case 0x5008: return "EMV_GET_AMOUNT_ERROR";
case 0x5009: return "EMV_CARD_REJECTED";
case 0x5010: return "EMV_AIP_NOT_RECEIVED";
case 0x5011: return "EMV_AFL_NOT_RECEIVED";
case 0x5012: return "EMV_AFL_LEN_OUT_OF_RANGE";
case 0x5013: return "EMV_SFI_OUT_OF_RANGE";
case 0x5014: return "EMV_AFL_INCORRECT";
case 0x5015: return "EMV_EXP_DATE_INCORRECT";
case 0x5016: return "EMV_EFF_DATE_INCORRECT";
case 0x5017: return "EMV_ISS_COD_TBL_OUT_OF_RANGE";
case 0x5018: return "EMV_CRYPTOGAM_TYPE_INCORRECT";
case 0x5019: return "EMV_PSE_NOT_SUPPORTED_BY_CARD";
case 0x5020: return "EMV_USER_SELECTED_LANGUAGE";
case 0x5021: return "EMV_SERVICE_NOT_ALLOWED";
case 0x5022: return "EMV_NO_TAG_FOUND";
case 0x5023: return "EMV_CARD_BLOCKED";
case 0x5024: return "EMV_LEN_INCORRECT";
case 0x5025: return "CARD_COM_ERROR";
case 0x5026: return "EMV_TSC_NOT_INCREASED";
case 0x5027: return "EMV_HASH_INCORRECT";
case 0x5028: return "EMV_NO_ARC";
case 0x5029: return "EMV_INVALID_ARC";
case 0x5030: return "EMV_NO_ONLINE_COMM";
case 0x5031: return "TRAN_TYPE_INCORRECT";
case 0x5032: return "EMV_APP_NO_SUPPORT";
case 0x5033: return "EMV_APP_NOT_SELECT";
case 0x5034: return "EMV_LANG_NOT_SELECT";
case 0x5035: return "EMV_NO_TERM_DATA";
case 0x6001: return "CVM_TYPE_UNKNOWN";
case 0x6002: return "CVM_AIP_NOT_SUPPORTED";
case 0x6003: return "CVM_TAG_8E_MISSING";
case 0x6004: return "CVM_TAG_8E_FORMAT_ERROR";
case 0x6005: return "CVM_CODE_IS_NOT_SUPPORTED";
case 0x6006: return "CVM_COND_CODE_IS_NOT_SUPPORTED";
case 0x6007: return "NO_MORE_CVM";
case 0x6008: return "PIN_BYPASSED_BEFORE";
case 0x7001: return "PK_BUFFER_SIZE_TOO_BIG";
case 0x7002: return "PK_FILE_WRITE_ERROR";
case 0x7003: return "PK_HASH_ERROR";
case 0x8001: return "NO_CARD_HOLDER_CONFIRMATION";
case 0x8002: return "GET_ONLINE_PIN";
case 0xD000: return "Data not exist";
case 0xD001: return "Data access error";
case 0xD100: return "RID not exist";
case 0xD101: return "RID existed";
case 0xD102: return "Index not exist";
case 0xD200: return "Maximum exceeded";
case 0xD201: return "Hash error";
case 0xD205: return "System Busy";
case 0xE001: return "Unable to go online";
case 0xE002: return "Technical Issue";
case 0xE003: return "Declined";
case 0xE004: return "Issuer Referral transaction";
case 0xF001: return "Decline the online transaction";
case 0xF002: return "Request to go online";
case 0xF003: return "Transaction is terminated";
case 0xF005: return "Application was not selected by kernel or ICC format error or ICC
missing data error";
case 0xF007: return "ICC didn't accept transaction";
case 0xF00A: return "Application may fallback to magstripe technology";
case 0xF00C: return "Transaction was cancelled";
case 0xF00D: return "Timeout";
case 0xF00F: return "Other EMV Error";
case 0xF010: return "Accept the offline transaction";
case 0xF011: return "Decline the offline transaction";
case 0xF021: return "ICC detected tah the conditions of use are not satisfied";
case 0xF022: return "No app were found on card matching terminal configuration";
case 0xF023: return "Terminal file does not exist";
case 0xF024: return "CAPK file does not exist";
case 0xF025: return "CRL Entry does not exist";
case 0xFFFE: return "Return code when blocking is disabled";
case 0xFFFF: return "Return code when command is not applicable on the selected device";
case 0xF005: return "ICC Encrypted C-APDU Data Structure Length Error Or Format Error.";
case 0xBBE0: return "CM100 Success";
case 0xBBE1: return "CM100 Parameter Error";
case 0xBBE2: return "CM100 Low Output Buffer";
case 0xBBE3: return "CM100 Card Not Found";
case 0xBBE4: return "CM100 Collision Card Exists";
case 0xBBE5: return "CM100 Too Many Cards Exist";

```



---

```

    case 0xBBE6: return "CM100 Saved Data Does Not Exist";
    case 0xBBE8: return "CM100 No Data Available";
    case 0xBBE9: return "CM100 Invalid CID Returned";
    case 0xBBEA: return "CM100 Invalid Card Exists";
    case 0xBBEC: return "CM100 Command Unsupported";
    case 0xBBED: return "CM100 Error In Command Process";
    case 0xBBEE: return "CM100 Invalid Command";

    case 0X9031: return "Unknown command";
    case 0X9032: return "Wrong parameter (such as the length of the command is incorrect)";

    case 0X9038: return "Wait (the command couldn't be finished in BWT)";
    case 0X9039: return "Busy (a previously command has not been finished)";
    case 0X903A: return "Number of retries over limit";

    case 0X9040: return "Invalid Manufacturing system data";
    case 0X9041: return "Not authenticated";
    case 0X9042: return "Invalid Master DUKPT Key";
    case 0X9043: return "Invalid MAC Key";
    case 0X9044: return "Reserved for future use";
    case 0X9045: return "Reserved for future use";
    case 0X9046: return "Invalid DATA DUKPT Key";
    case 0X9047: return "Invalid PIN Pairing DUKPT Key";
    case 0X9048: return "Invalid DATA Pairing DUKPT Key";
    case 0X9049: return "No nonce generated";
    case 0X9949: return "No GUID available. Perform getVersion first.";
    case 0X9950: return "MAC Calculation unsuccessful. Check BDK value.";

    case 0X904A: return "Not ready";
    case 0X904B: return "Not MAC data";

    case 0X9050: return "Invalid Certificate";
    case 0X9051: return "Duplicate key detected";
    case 0X9052: return "AT checks failed";
    case 0X9053: return "TR34 checks failed";
    case 0X9054: return "TR31 checks failed";
    case 0X9055: return "MAC checks failed";
    case 0X9056: return "Firmware download failed";

    case 0X9060: return "Log is full";
    case 0X9061: return "Removal sensor unengaged";
    case 0X9062: return "Any hardware problems";

    case 0X9070: return "ICC communication timeout";
    case 0X9071: return "ICC data error (such check sum error)";
    case 0X9072: return "Smart Card not powered up";

}
return "";
}

```

## Chapter 9

# Enumeration Reference

### Common

```

public enum EVENT_TRANSACTION_DATA_Types
{
    EVENT_TRANSACTION_DATA_UNKNOWN, EVENT_TRANSACTION_DATA_CARD_DATA, EVENT_TRANSACTION_DATA_EMV_DATA,
    EVENT_TRANSACTION_DATA_MSR_CANCEL_KEY, EVENT_TRANSACTION_DATA_MSR_BACKSPACE_KEY,
    EVENT_TRANSACTION_DATA_MSR_ENTER_KEY, EVENT_TRANSACTION_DATA_MSR_DATA_ERROR, EVENT_TRANSACTION_PIN_DATA
}

public enum CAPTURE_ENCODE_TYPE
{
    CAPTURE_ENCODE_TYPE_ISOABA, CAPTURE_ENCODE_TYPE_AAMVA, CAPTURE_ENCODE_TYPE_Other,
    CAPTURE_ENCODE_TYPE_Raw, CAPTURE_ENCODE_TYPE_JisI_II
}

public enum CAPTURE_ENCRYPT_TYPE
{
    CAPTURE_ENCRYPT_TYPE_TDES, CAPTURE_ENCRYPT_TYPE_AES, CAPTURE_ENCRYPT_TYPE_NONE
}

public enum EMV_ENCRYPTION_MODE
{
    EMV_ENCRYPTION_MODE_TDES = 0, EMV_ENCRYPTION_MODE_AES = 1
}

public enum EMV_ENCRYPTION_MODE
{
    EMV_ENCRYPTION_MODE_TDES = 0, EMV_ENCRYPTION_MODE_AES = 1
}

public enum EMV_LCD_DISPLAY_MODE
{
    EMV_LCD_DISPLAY_MODE_CANCEL = 0, EMV_LCD_DISPLAY_MODE_MENU = 1, EMV_LCD_DISPLAY_MODE_PROMPT = 2,
    EMV_LCD_DISPLAY_MODE_MESSAGE = 3, EMV_LCD_DISPLAY_MODE_LANGUAGE_SELECT = 8, EMV_LCD_DISPLAY_MODE_CLEAR_SCREEN = 16
}

public enum EMV_RESULT_CODE
{
    EMV_RESULT_CODE_APPROVED_OFFLINE = 0,
    EMV_RESULT_CODE_DECLINED_OFFLINE = 1,
    EMV_RESULT_CODE_APPROVED = 2,
    EMV_RESULT_CODE_DECLINED = 3,
    EMV_RESULT_CODE_GO_ONLINE = 4,
    EMV_RESULT_CODE_CALL_YOUR_BANK = 5,
    EMV_RESULT_CODE_NOT_ACCEPTED = 6,
    EMV_RESULT_CODE_FALLBACK_TO_MSR = 7,
    EMV_RESULT_CODE_TIMEOUT = 8,
    EMV_RESULT_CODE_GO_ONLINE_CTLS = 9,
    EMV_RESULT_CODE_AUTHENTICATE_TRANSACTION = 0x0010,
    EMV_RESULT_CODE_SWIPE_NON_ICC = 17,
    EMV_RESULT_CODE_CTLS_TWO_CARDS = 0x7A,
    EMV_RESULT_CODE_CTLS_TERMINATE = 0x7E,
    EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER = 0x7D,
    EMV_RESULT_CODE_UNABLE_TO_REACH_HOST
}

```



## Chapter 10

# EMV Callback

During an EMV transaction, without a built-in LCD display on the SpectrumPro, LCD Display messages will be returned as an EMV Callback.

In the MessageCallback for the SDK, there is a DeviceState EMVCallback. When this DeviceState is received, a [IDTechSDK::EMV\\_Callback](#) class object will be returned. [IDTechSDK::EMV\\_Callback::callbackType](#) will specify the type of callback: EMV\_CALLBACK\_TYPE\_LCD, EMV\_CALLBACK\_TYPE\_PINPAD, EMV\_CALLBACK\_MSR. The SpectrumPro will not utilize the EMV\_CALLBACK\_TYPE\_PINPAD.

If MiniSmartII terminal settings specify MSR is part of the configuration, for cases of fallback, the type of callback will be EMV\_CALLBACK\_MSR. When this is received, MSR data must be collected and returned to [IDTechSDK::IDT\\_SpectrumPro::emv\\_callbackResponseMSR\(\)](#) to complete the transaction.

For LCD display messages, the callback type will be EMV\_CALLBACK\_TYPE\_LCD. To evaluate what kind of LCD message, you get EMV\_LCD\_DISPLAY\_MODE from [IDTechSDK::EMV\\_Callback::lcd\\_displayMode](#): 1- LCD\_DISPLAY\_MODE\_MENU: Menu selection, response required with selected menu index #, or 0 to cancel 2- LCD\_DISPLAY\_MODE\_PROMPT: Message Prompt, response required 'E' for Enter/Accept, or 'C' for cancel 3- LCD\_DISPLAY\_MODE\_MESSAGE: Display Message, no response required 8 - LCD\_DISPLAY\_MODE\_LANGUAGE\_SELECT: Language selection, response required with selected language index # 16 - LCD\_DISPLAY\_MODE\_CLEAR\_SCREEN: Request to clear LCD screen of information

If the mode is LCD\_DISPLAY\_MODE\_MESSAGE or LCD\_DISPLAY\_MODE\_CLEAR\_SCREEN, these do not pause the EMV transaction. These two modes are for displaying a message (no response required), or for clearing the screen.

If the mode is LCD\_DISPLAY\_MODE\_MENU, LCD\_DISPLAY\_MODE\_PROMPT, or LCD\_DISPLAY\_MODE\_LANGUAGE\_SELECT, the provided message must be displayed, and then the EMV transaction pauses until a response is sent to [IDTechSDK::IDT\\_SpectrumPro::emv\\_callbackResponseLCD\(\)](#).

The message to display is byte[] lcd\_messages. This contains either Message String, or a Message ID according to LCD Foreign Language Mapping Table ([Foreign Language Mapping Table](#)).

## Chapter 11

# EMV Tag Reference

Tag	Description
42	Issuer Identification Number (IIN)
4F	Application Identifier (ADF Name)
50	Application Label
52	Command to perform
56	Track 1 Data
57	Track 2 Equivalent Data
5A	Application Primary Account Number (PAN)
5D	Deleted (see 9D)
5F20	Cardholder Name
5F24	Application Expiration Date
5F25	Application Effective Date
5F28	Issuer Country Code
5F2A	Transaction Currency Code
5F2D	Language Preference
5F30	Service Code
5F34	Application Primary Account Number (PAN) Sequence Number (PSN)
5F36	Transaction Currency Exponent
5F3C	Transaction Reference Currency Code
5F3D	Transaction Reference Currency Exponent
5F50	Issuer URL
5F53	International Bank Account Number (IBAN)
5F54	Bank Identifier Code (BIC)
5F55	Issuer Country Code (alpha2 format)
5F56	Issuer Country Code (alpha3 format)
5F57	Account Type
61	Application Template
62	File Control Parameters (FCP) Template
6F	File Control Information (FCI) Template
70	READ RECORD Response Message Template
71	Issuer Script Template 1
72	Issuer Script Template 2
73	Directory Discretionary Template
77	Response Message Template Format 2
80	Response Message Template Format 1
81	Amount, Authorised (Binary)

Tag	Description
82	Application Interchange Profile (AIP)
83	Command Template
84	Dedicated File (DF) Name
86	Issuer Script Command
87	Application Priority Indicator
88	Short File Identifier (SFI)
89	Authorisation Code
8A	Authorisation Response Code (ARC)
8C	Card Risk Management Data Object List 1 (CDOL1)
8D	Card Risk Management Data Object List 2 (CDOL2)
8E	Cardholder Verification Method (CVM) List
8F	Certification Authority Public Key Index (PKI)
90	Issuer Public Key Certificate
91	Issuer Authentication Data
92	Issuer Public Key Remainder
93	Signed Application Data
94	Application File Locator (AFL)
95	Terminal Verification Results (TVR)
97	Transaction Certificate Data Object List (TDOL)
98	Transaction Certificate (TC) Hash Value
99	Transaction Personal Identification Number (PIN) Data
9A	Transaction Date
9B	Transaction Status Information
9C	Transaction Type
9D	Directory Definition File (DDF) Name
9F01	Acquirer Identifier
9F02	Amount, Authorised (Numeric)
9F03	Amount, Other (Numeric)
9F04	Amount, Other (Binary)
9F05	Application Discretionary Data
9F06	Application Identifier (AID) - terminal
9F07	Application Usage Control (AUC)
9F08	Application Version Number
9F09	Application Version Number
9F0B	Cardholder Name Extended
9F0D	Issuer Action Code - Default
9F0E	Issuer Action Code - Denial
9F0F	Issuer Action Code - Online
9F10	Issuer Application Data (IAD)
9F11	Issuer Code Table Index
9F12	Application Preferred Name
9F13	Last Online Application Transaction Counter (ATC) Register
9F14	Lower Consecutive Offline Limit
9F15	Merchant Category Code
9F16	Merchant Identifier
9F17	Personal Identification Number (PIN) Try Counter
9F18	Issuer Script Identifier
9F19	Deleted (see 9F49)
9F1A	Terminal Country Code

Tag	Description
9F1B	Terminal Floor Limit
9F1C	Terminal Identification
9F1D	Terminal Risk Management Data
9F1E	Interface Device (IFD) Serial Number
9F1F	Track 1 Discretionary Data
9F20	Track 2 Discretionary Data
9F21	Transaction Time
9F22	Certification Authority Public Key Index (PKI)
9F23	Upper Consecutive Offline Limit
9F26	Application Cryptogram (AC)
9F27	Cryptogram Information Data (CID)
9F29	Extended Selection
9F2A	Kernel Identifier
9F2D	Integrated Circuit Card (ICC) PIN Encipherment Public Key Certificate
9F2E	Integrated Circuit Card (ICC) PIN Encipherment Public Key Exponent
9F2F	Integrated Circuit Card (ICC) PIN Encipherment Public Key Remainder
9F32	Issuer Public Key Exponent
9F33	Terminal Capabilities
9F34	Cardholder Verification Method (CVM) Results
9F35	Terminal Type
9F36	Application Transaction Counter (ATC)
9F37	Unpredictable Number (UN)
9F37	Unpredictable Number (UN) (Reader/Terminal)
9F38	Processing Options Data Object List (PDOL)
9F39	Point-of-Service (POS) Entry Mode
9F3A	Amount, Reference Currency
9F3B	Application Reference Currency
9F3C	Transaction Reference Currency Code
9F3D	Transaction Reference Currency Exponent
9F40	Additional Terminal Capabilities
9F41	Transaction Sequence Counter
9F42	Application Currency Code
9F43	Application Reference Currency Exponent
9F44	Application Currency Exponent
9F45	Data Authentication Code
9F46	Integrated Circuit Card (ICC) Public Key Certificate
9F46	Application Public Key Certificate
9F47	Integrated Circuit Card (ICC) Public Key Exponent
9F47	Application Public Key Exponent
9F48	Integrated Circuit Card (ICC) Public Key Remainder
9F48	Application Public Key Remainder
9F49	Dynamic Data Authentication Data Object List (DDOL)
9F4A	Static Data Authentication Tag List (SDA)
9F4B	Signed Dynamic Application Data (SDAD)
9F4C	ICC Dynamic Number
9F4D	Log Entry
9F4E	Merchant Name and Location
9F4F	Log Format
9F50	Offline Accumulator Balance

Tag	Description
9F50	Cardholder Verification Status
9F51	Application Currency Code
9F51	DRDOL
9F52	Application Default Action (ADA)
9F52	Terminal Compatibility Indicator
9F53	Consecutive Transaction Counter International Limit (CTCIL)
9F53	Transaction Category Code
9F53	Terminal Interchange Profile (dynamic)
9F54	Cumulative Total Transaction Amount Limit (CTTAL)
9F54	DS ODS Card
9F55	Geographic Indicator
9F56	Issuer Authentication Indicator
9F57	Issuer Country Code
9F58	Consecutive Transaction Counter Limit (CTCL)
9F59	Consecutive Transaction Counter Upper Limit (CTCUL)
9F5A	Application Program Identifier (Program ID)
9F5B	Issuer Script Results
9F5B	DSDOL
9F5C	Cumulative Total Transaction Amount Upper Limit (CTTAUL)
9F5C	DS Requested Operator ID
9F5C	Magstripe Data Object List (MDOL)
9F5D	Available Offline Spending Amount (AOSA)
9F5D	Application Capabilities Information (ACI)
9F5E	Consecutive Transaction International Upper Limit (CTIUL)
9F5E	DS ID
9F5F	DS Slot Availability
9F5F	Offline Balance
9F60	CVC3 (Track1)
9F60	Issuer Update Parameter
9F60	P3 Generated 3DES KEYS
9F61	CVC3 (Track2)
9F62	PCVC3 (Track1)
9F62	Encrypted PIN - ISO 95641 Format 0 (Thales P3 Format 01)
9F63	Offline Counter Initial Value
9F63	PUNATC (Track1)
9F64	NATC (Track1)
9F65	PCVC3 (Track2)
9F66	Terminal Transaction Qualifiers (TTQ)
9F66	PUNATC (Track2)
9F67	MSD Offset
9F67	NATC (Track2)
9F68	Card Additional Processes
9F69	Card Authentication Related Data
9F69	UDOL
9F6A	Unpredictable Number (Numeric)
9F6B	Card CVM Limit
9F6B	Track 2 Data
9F6C	Card Transaction Qualifiers (CTQ)
9F6D	VLP Reset Threshold
9F6D	Mag-stripe Application Version Number (Reader)



Tag	Description
9F6D	Kernel 4 Reader Capabilities
9F6E	Third Party Data
9F6E	Form Factor Indicator (FFI)
9F6E	Terminal Transaction Capabilities
9F6F	DS Slot Management Control
9F70	Protected Data Envelope 1
9F70	Card Interface Capabilities
9F71	Protected Data Envelope 2
9F71	Mobile CVM Results
9F72	Protected Data Envelope 3
9F72	Consecutive Transaction Limit (International—Country)
9F73	Protected Data Envelope 4
9F73	Currency Conversion Parameters
9F74	Protected Data Envelope 5
9F74	VLP Issuer Authorisation Code
9F75	Unprotected Data Envelope 1
9F75	Cumulative Total Transaction Amount Limit-Dual Currency
9F76	Unprotected Data Envelope 2
9F76	Secondary Application Currency Code
9F77	Unprotected Data Envelope 3
9F78	Unprotected Data Envelope 4
9F79	Unprotected Data Envelope 5
9F77	VLP Funds Limit
9F78	VLP Single Transaction Limit
9F79	VLP Available Funds
9F7A	VLP Terminal Support Indicator
9F7B	VLP Terminal Transaction Limit
9F7C	Customer Exclusive Data (CED)
9F7C	Merchant Custom Data
9F7D	DS Summary 1
9F7D	VISA Applet Data
9F7E	Mobile Support Indicator
9F7E	Application life cycle data (8 first bytes)
9F7F	DS Unpredictable Number
9F7F	Card Production Life Cycle (CPLC) Data
A5	File Control Information (FCI) Proprietary Template
BF0C	File Control Information (FCI) Issuer Discretionary Data
BF50	Visa Fleet - CDO
BF60	Integrated Data Storage Record Update Template
C3	Card issuer action code -decline
C4	Card issuer action code -default
C5	Card issuer action code online
C6	PIN Try Limit
C7	CDOL 1 Related Data Length
C8	Card risk management country code
C9	Card risk management currency code
CA	Lower cumulative offline transaction amount
CB	Upper cumulative offline transaction amount
CD	Card Issuer Action Code (PayPass) – Default

Tag	Description
CE	Card Issuer Action Code (PayPass) – Online
CF	Card Issuer Action Code (PayPass) – Decline
D1	Currency conversion table
D2	Integrated Data Storage Directory (IDSD)
D3	Additional check table
D5	Application Control
D6	Default ARPC response code
D7	Application Control (PayPass)
D8	AIP (PayPass)
D9	AFL (PayPass)
DA	Static CVC3-TRACK1
DB	Static CVC3-TRACK2
DC	IVCVC3-TRACK1
DD	IVCVC3-TRACK2
DF70	Generic Name String
DF71	Value Added Tax 1
DF71	Generic Numeric
DF72	Value Added Tax 2
DF72	Generic Specification String
DF73	Merchant Category Code
DF73	Generic Implementation String
DF74	Discover Optional Features
DF75	Communications Error Message Delay
DF76	TVR from GenAC
DF77	ViVOpay MSR Custom Data Output Tag
DF78	MC Timing Performance Enable
DF79	Card Disable Mask
DF7A	Card Disable Interval
DF7B	Serial Port (UART) Inter-character Timeout Period
DF7C	Auto Switch Feature
DF7D	Track Formatting Feature
DF7F	Improved Collision Detection & Media Removal Feature
FF70	Serial Finite State Machine Version
FF71	Transaction Finite State Machine Version
FF72	System Information Suite
FF73	Serial Protocol Version
FF74	Serial Protocol Suite
FF75	L1 Paypass Version
FF76	L1 LCR Version
FF77	L2 Card App Version
FF78	L2 Card App Suite
FF79	GMEDs Data
FF79	User Experience Version
FF7A	User Experience Suite
FF7B	ViVOtech Proprietary Suite
FF7C	VIUDS Scheme IDs Supported
FF7D	VIUDS Scheme ID Selection Criteria
FFE0	Registered Application Provider Identifier (RID)
FFE1	Partial Selection Allowed
FFE2	Application Flow

Tag	Description
FFE3	Selection Features - GR 1.2.10
FFE4	Group Number / Fallback Group
FFE5	Max AID Length
FFE6	AID Disabled
FFE7	Interface Support
FFE8	Exclude from Processing
FFE9	Kernel ID Transaction Type Group List
FFEA	Default Kernel ID
FFF0	Specific Features Switch
FFF1	Terminal Contactless Transaction Limit
FFF2	Terminal IFD
FFF3	Application Capability
FFF4	Visa Reader Risk Flags
FFF5	CVM Required Limit
FFF6	Torn Transaction Log Clean Interval (minutes)
FFF7	Burst Mode
FFF8	UI Scheme
FFF9	LCD Font Size
FFFA	LCD delay Time
FFFB	Language Option for LCD
FFFC	Force MagStripe
FFFD	TAC - Online
FFFE	TAC - Default
FFFF	TAC - Denial
DF8123	Reader Contactless Floor Limit Data
DFDE04	MSR Encryption Option
DFEE12	KSN of Account DUKPT Key
DFEE15	Application Selection Indicator
DFEE16	DUKPT Key or MKSK Select for Online PIN Encrypted
DFEE17	ICC Terminal Entry Mode
DFEE18	MSR Terminal Entry Mode
DFEE19	Online DOL
DFEE1A	Output data element
DFEE1B	Authorization Request data elements
DFEE1E	Terminal Configuration
DFEE1F	Issuer Script Limit
DFEE20	ICC power on waiting time
DFEE21	ICC L1 data transaction waiting time
DFEE22	Driver (Menu, Get PIN, Get MSR) Timeout
DFEE23	MSR all track data
DFEE24	Force Acceptance
DFEE25	ICC Response Code
DFEE26	Encryption Status Information
DFEE27	MSR Control
FFEE01	ViVOpay TLV Group Tag
FFEE02	ViVOpay Pre-PPSE Special Flow Group Tag
FFEE03	ViVOpay Post-PPSE Special Flow Group Tag
FFEE04	M/Chip3 Intermediate Message Data
FFEE04	ViVOpay MChip Group Status

Tag	Description
FFEE05	M/Chip3 Intermediate Message Marker
FFEE06	ApplePay VAS Container
FFEE10	ViVOpay MChip Group Tag
FFEE11	ViVOpay Discover Group Tag
FFEE12	KSN of Account DUKPT Key
FFEE13	Track 1 Data
FFEE14	Track 2 Data
FFEE1C	Unpredictable Number Range
FFEE1D	Sensitive Data Mask

## Chapter 12

# Namespace Index

### 12.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">IDTechSDK</a> . . . . .	<a href="#">63</a>
-------------------------------------	--------------------

## Chapter 13

# Class Index

### 13.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">IDTechSDK.EMV_Callback</a>	66
<a href="#">IDTechSDK.IDT_SpectrumPro</a>	68
<a href="#">IDTechSDK.IDTTransactionData</a>	101

## Chapter 14

# Namespace Documentation

### 14.1 IDTechSDK Namespace Reference

#### Classes

- class [EMV\\_Callback](#)
- class [IDT\\_SpectrumPro](#)
- class [IDTTransactionData](#)

#### Enumerations

- enum [EMV\\_CALLBACK\\_TYPE](#) { [EMV\\_CALLBACK\\_TYPE\\_LCD](#) =1, [EMV\\_CALLBACK\\_TYPE\\_PINPAD](#) =2, [EMV\\_CALLBACK\\_MSR](#) =3 }
- enum [EMV\\_LCD\\_DISPLAY\\_MODE](#) { [EMV\\_LCD\\_DISPLAY\\_MODE\\_CANCEL](#) = 0, [EMV\\_LCD\\_DISPLAY\\_MODE\\_MENU](#) = 1, [EMV\\_LCD\\_DISPLAY\\_MODE\\_PROMPT](#) = 2, [EMV\\_LCD\\_DISPLAY\\_MODE\\_MESSAGE](#) = 3, [EMV\\_LCD\\_DISPLAY\\_MODE\\_LANGUAGE\\_SELECT](#) = 8, [EMV\\_LCD\\_DISPLAY\\_MODE\\_CLEAR\\_SCREEN](#) = 16 }
- enum [CTLS\\_APPLICATION](#) { [CTLS\\_APPLICATION\\_NONE](#) = 0, [CTLS\\_APPLICATION\\_MASTERCARD](#) = 1, [CTLS\\_APPLICATION\\_VISA](#) = 2, [CTLS\\_APPLICATION\\_AAMEX](#) = 3, [CTLS\\_APPLICATION\\_DISCOVER](#) = 4, [CTLS\\_APPLICATION\\_SPEEDPASS](#) = 5, [CTLS\\_APPLICATION\\_GIFT\\_CARD](#) = 6, [CTLS\\_APPLICATION\\_DINERS\\_CLUB](#) = 7, [CTLS\\_APPLICATION\\_EN\\_ROUTE](#) = 8, [CTLS\\_APPLICATION\\_JCB](#) = 9, [CTLS\\_APPLICATION\\_VIVO\\_DIAGNOSTIC](#) = 10, [CTLS\\_APPLICATION\\_HID](#) = 11, [CTLS\\_APPLICATION\\_MSR\\_SWIPE](#) = 12, [CTLS\\_APPLICATION\\_RESERVED](#) = 13, [CTLS\\_APPLICATION\\_ON\\_DES\\_FIRE\\_TRACK\\_DATA](#) = 14, [CTLS\\_APPLICATION\\_DES\\_FIRE\\_RAW\\_DATA](#) = 15, [CTLS\\_APPLICATION\\_RBS](#) = 17, [CTLS\\_APPLICATION\\_VIVO\\_COMM](#) = 20 }
- enum [EMV\\_PIN\\_MODE](#) { [EMV\\_PIN\\_MODE\\_CANCEL](#) =0, [EMV\\_PIN\\_MODE\\_ONLINE\\_DUKPT](#) =1, [EMV\\_PIN\\_MODE\\_ONLINE\\_MKSK](#) =2, [EMV\\_PIN\\_MODE\\_OFFLINE](#) =3 }
- enum [EMV\\_RESULT\\_CODE](#) { [EMV\\_RESULT\\_CODE\\_APPROVED\\_OFFLINE](#) = 0, [EMV\\_RESULT\\_CODE\\_DECLINED\\_OFFLINE](#) = 1, [EMV\\_RESULT\\_CODE\\_APPROVED](#) = 2, [EMV\\_RESULT\\_CODE\\_DECLINED](#) = 3, [EMV\\_RESULT\\_CODE\\_GO\\_ONLINE](#) = 4, [EMV\\_RESULT\\_CODE\\_CALL\\_YOUR\\_BANK](#) = 5, [EMV\\_RESULT\\_CODE\\_NOT\\_ACCEPTED](#) = 6, [EMV\\_RESULT\\_CODE\\_FALLBACK\\_TO\\_MSR](#) = 7, [EMV\\_RESULT\\_CODE\\_TIMEOUT](#) = 8, [EMV\\_RESULT\\_CODE\\_GO\\_ONLINE\\_CTLS](#) = 9, [EMV\\_RESULT\\_CODE\\_AUTHENTICATE\\_TRANSACTION](#) = 0x0010, [EMV\\_RESULT\\_CODE\\_TRANSACTION\\_CANCELLED](#) = 0x0012, [EMV\\_RESULT\\_CODE\\_SWIPE\\_NON\\_ICC](#) = 0x11, [EMV\\_RESULT\\_CODE\\_CTLS\\_TWO\\_CARDS](#) = 0x7A, [EMV\\_RESULT\\_CODE\\_CTLS\\_TERMINATE](#) = 0x7E, [EMV\\_RESULT\\_CODE\\_CTLS\\_TERMINATE\\_TRY](#) = 0x7F }

```

ANOTHER = 0x7D,
EMV_RESULT_CODE_MSR_SWIPE_CAPTURED = 0x80, EMV_RESULT_CODE_UNABLE_TO_REAC↵
H_HOST = 0xFF, EMV_RESULT_CODE_FILE_ARG_INVALID = 0x1001, EMV_RESULT_CODE_FILE_↵
OPEN_FAILED = 0x1002,
EMV_RESULT_CODE_FILE_OPERATION_FAILED = 0x1003, EMV_RESULT_CODE_MEMORY_NOT_↵
ENOUGH = 0x2001, EMV_RESULT_CODE_SMARTCARD_OK = 0x3001, EMV_RESULT_CODE_SMA↵
RTCARD_FAIL = 0x3002,
EMV_RESULT_CODE_SMARTCARD_INIT_FAILED = 0x3003, EMV_RESULT_CODE_FALLBACK_SI↵
TUATION = 0x3004, EMV_RESULT_CODE_SMARTCARD_ABSENT = 0x3005, EMV_RESULT_CODE_↵
SMARTCARD_TIMEOUT = 0x3006,
EMV_RESULT_CODE_MSR_CARD_ERROR = 0x3007, EMV_RESULT_CODE_PARSING_TAGS_FAIL↵
ED = 0x5001, EMV_RESULT_CODE_CARD_DATA_ELEMENT_DUPLICATE = 0x5002, EMV_RESULT_↵
_CODE_DATA_FORMAT_INCORRECT = 0x5003,
EMV_RESULT_CODE_APP_NO_TERM = 0x5004, EMV_RESULT_CODE_APP_NO_MATCHING =
0x5005, EMV_RESULT_CODE_MANDATORY_OBJECT_MISSING = 0x5006, EMV_RESULT_CODE_↵
_APP_SELECTION_RETRY = 0x5007,
EMV_RESULT_CODE_AMOUNT_ERROR_GET = 0x5008, EMV_RESULT_CODE_CARD_REJECTED =
0x5009, EMV_RESULT_CODE_AIP_NOT_RECEIVED = 0x5010, EMV_RESULT_CODE_AFL_NOT_RE↵
CEIVED = 0x5011,
EMV_RESULT_CODE_AFL_LEN_OUT_OF_RANGE = 0x5012, EMV_RESULT_CODE_SFI_OUT_OF_↵
RANGE = 0x5013, EMV_RESULT_CODE_AFL_INCORRECT = 0x5014, EMV_RESULT_CODE_EXP_D↵
ATE_INCORRECT = 0x5015,
EMV_RESULT_CODE_EFF_DATE_INCORRECT = 0x5016, EMV_RESULT_CODE_ISS_COD_TBL↵
_OUT_OF_RANGE = 0x5017, EMV_RESULT_CODE_CRYPTOGAM_TYPE_INCORRECT = 0x5018,
EMV_RESULT_CODE_PSE_BY_CARD_NOT_SUPPORTED = 0x5019,
EMV_RESULT_CODE_USER_LANGUAGE_SELECTED = 0x5020, EMV_RESULT_CODE_SERVICE_↵
NOT_ALLOWED = 0x5021, EMV_RESULT_CODE_NO_TAG_FOUND = 0x5022, EMV_RESULT_CODE_↵
_CARD_BLOCKED = 0x5023,
EMV_RESULT_CODE_LEN_INCORRECT = 0x5024, EMV_RESULT_CODE_CARD_COM_ERROR =
0x5025, EMV_RESULT_CODE_TSC_NOT_INCREASED = 0x5026, EMV_RESULT_CODE_HASH_INC↵
ORRECT = 0x5027,
EMV_RESULT_CODE_ARC_NOT_PRESENCE = 0x5028, EMV_RESULT_CODE_ARC_INVALID =
0x5029, EMV_RESULT_CODE_COMM_NO_ONLINE = 0x5030, EMV_RESULT_CODE_TRAN_TYPE_I↵
NCORRECT = 0x5031,
EMV_RESULT_CODE_APP_NO_SUPPORT = 0x5032, EMV_RESULT_CODE_APP_NOT_SELECT =
0x5033, EMV_RESULT_CODE_LANG_NOT_SELECT = 0x5034, EMV_RESULT_CODE_TERM_DATA_↵
NOT_PRESENCE = 0x5035,
EMV_RESULT_CODE_CVM_TYPE_UNKNOWN = 0x6001, EMV_RESULT_CODE_CVM_AIP_NOT_SU↵
PPORTED = 0x6002, EMV_RESULT_CODE_CVM_TAG_8E_MISSING = 0x6003, EMV_RESULT_COD↵
E_CVM_TAG_8E_FORMAT_ERROR = 0x6004,
EMV_RESULT_CODE_CVM_CODE_IS_NOT_SUPPORTED = 0x6005, EMV_RESULT_CODE_CVM↵
_COND_CODE_IS_NOT_SUPPORTED = 0x6006, EMV_RESULT_CODE_CVM_NO_MORE = 0x6007,
EMV_RESULT_CODE_PIN_BYPASSED_BEFORE = 0x6008,
EMV_RESULT_CODE_UNKONWN = 0xffff }

```

### 14.1.1 Enumeration Type Documentation

#### 14.1.1.1 enum IDTechSDK.CTLS\_APPLICATION [strong]

Define CTLS\_APPLICATION.

#### 14.1.1.2 enum IDTechSDK.EMV\_CALLBACK\_TYPE [strong]

Define EMV\_CALLBACK\_TYPES.



14.1.1.3 `enum IDTechSDK.EMV_LCD_DISPLAY_MODE` `[strong]`

Define EMV\_LCD\_DISPLAY\_MODE.

14.1.1.4 `enum IDTechSDK.EMV_PIN_MODE` `[strong]`

Define EMV\_PIN\_MODE.

14.1.1.5 `enum IDTechSDK.EMV_RESULT_CODE` `[strong]`

Define EMV\_PIN\_MODE.

## Chapter 15

# Class Documentation

### 15.1 IDTechSDK.EMV\_Callback Class Reference

#### Public Attributes

- int [msr\\_swipeTimeout](#)
- int [msr\\_displayMessage](#)
- [EMV\\_PIN\\_MODE](#) pin\_pinMode
- int [pin\\_entryStartTimeout](#)
- int [pin\\_entryInterval](#)
- byte[] [pin\\_KSN](#)
- byte[] [pin\\_truncatedPAN](#)
- [EMV\\_CALLBACK\\_TYPE](#) callbackType
- [EMV\\_LCD\\_DISPLAY\\_MODE](#) lcd\_displayMode
- int [lcd\\_entryTimeout](#)
- int [lcd\\_entryTimeoutMinor](#)
- byte[] [language](#)
- byte[] [lcd\\_messages](#)
- UInt16 [lcd\\_backlightTimeout](#)
- bool [maskEntry](#)

#### 15.1.1 Detailed Description

Class for LCD Message

#### 15.1.2 Member Data Documentation

##### 15.1.2.1 [EMV\\_CALLBACK\\_TYPE](#) IDTechSDK.EMV\_Callback.callbackType

Callback Type.

1- [EMV\\_CALLBACK\\_TYPE\\_LCD](#): LCD Display Hardware Event 2- [EMV\\_CALLBACK\\_TYPE\\_PINPAD](#): Pinpad Hardware Event 3- [EMV\\_CALLBACK\\_MSR](#): MSR Hardware Event

##### 15.1.2.2 [byte \[\]](#) IDTechSDK.EMV\_Callback.language

Message Language

2 Bytes

- EN - English (default)
- ES - Spanish
- ZH - Chinese
- FR – French

#### 15.1.2.3 UInt16 IDTechSDK.EMV\_Callback.lcd\_backlightTimeout

##### Backlight Timeout

If Normal Display or Menu Display, Total timeout for keypad entry, in second default is 30 seconds. 0x0000 = backlight off, 0xFFFF = backlight on

#### 15.1.2.4 EMV\_LCD\_DISPLAY\_MODE IDTechSDK.EMV\_Callback.lcd\_displayMode

##### Display Mode.

1- LCD\_DISPLAY\_MODE\_MENU: Menu selection, response required with selected menu index #, or 0 to cancel  
 2- LCD\_DISPLAY\_MODE\_PROMPT: Message Prompt, response required 'E' for Enter/Accept, or 'C' for cancel  
 3- LCD\_DISPLAY\_MODE\_MESSAGE: Display Message, no response required  
 8 – LCD\_DISPLAY\_MODE\_LANGUAGE\_SELECT: Language selection, response required with selected language index #  
 16 - LCD\_DISPLAY\_MODE\_CLEAR\_SCREEN: Request to clear LCD screen of information

#### 15.1.2.5 int IDTechSDK.EMV\_Callback.lcd\_entryTimeout

##### Keypad Entry Timeout

If Normal Display or Menu Display, Total timeout for keypad entry, in second default is 30 seconds.

#### 15.1.2.6 int IDTechSDK.EMV\_Callback.lcd\_entryTimeoutMinor

##### Keypad Entry Timeout Minor

If Normal Display or Menu Display, minor timeout during each keypad entry, in second, little endian, default is 10 seconds. Note: Minor timeout will erase all previous keypad entry.

#### 15.1.2.7 byte [] IDTechSDK.EMV\_Callback.lcd\_messages

##### Display Message

repeatable combination of [Line][Message][0x1C] [Line] - Display line number (1-First Line, n-nth Line), Maximum 16 lines. •The lower 7 bits is for line number. •The MSB is to indicate following message is a Message String or Message ID. •MSB – 0: Message String. (It is valid for “Menu Display” and “Language Menu Display”) •MSB – 1: Message ID. (It is only valid for “Menu Display”) [Message] - Message String or Message ID. Message String: •“Menu Display” : character in the range of 0x20 – 0x7f, Maximum 16 characters • “Language Menu Display” : 2 bytes Language ID EN - English (default) ES - Spanish ZH - Chinese FR – French

#### 15.1.2.8 bool IDTechSDK.EMV\_Callback.maskEntry

##### Mask Entry

If True, keypad entry should be masked with '\*'

**15.1.2.9 int IDTechSDK.EMV\_Callback.msr\_displayMessage**

MSR Message

Message to display during swipe request

**15.1.2.10 int IDTechSDK.EMV\_Callback.msr\_swipeTimeout**

Swipe Timeout

Timeout value waiting for MSR Swipe

**15.1.2.11 int IDTechSDK.EMV\_Callback.pin\_entryInterval**

PIN Entry Interval Timeout value of interval between input each PIN

**15.1.2.12 int IDTechSDK.EMV\_Callback.pin\_entryStartTimeout**

PIN Entry Start Timeout

Timeout value waiting for PIN entry to start

**15.1.2.13 byte [] IDTechSDK.EMV\_Callback.pin\_KSN**

PIN KSN

Pairing DUKPT KSN

**15.1.2.14 EMV\_PIN\_MODE IDTechSDK.EMV\_Callback.pin\_pinMode**

PIN Mode.

0- EMV\_PIN\_MODE\_CANCEL: Entry cancel through command. No response required 1- EMV\_PIN\_MODE\_ONLINE\_DUKPT: PIN\_DUKPT\_KEY required as response 2- EMV\_PIN\_MODE\_ONLINE\_MKSK: PIN\_SESSION\_KEY required as response 3 – EMV\_PIN\_MODE\_OFFLINE: PIN\_PAIRING\_DUKPT\_KEY required as response, unless devices does not implement pairing function, then plaintext PIN required as response

**15.1.2.15 byte [] IDTechSDK.EMV\_Callback.pin\_truncatedPAN**

Truncated PAN

Truncated PAN

The documentation for this class was generated from the following file:

- Source/IDT\_Transactions.cs

## 15.2 IDTechSDK.IDT\_SpectrumPro Class Reference

### Public Member Functions

- RETURN\_CODE [pin\\_getPIN](#) (int mode, int PANSource, string iccPAN, int startTimeout, int entryTimeout, string language)
- RETURN\_CODE [pin\\_cancelPINEntry](#) ()
- RETURN\_CODE [pin\\_passThroughMode](#) (bool enable)
- RETURN\_CODE [device\\_sendDataCommand](#) (string cmd, bool calcLRC, ref byte[] response)

- RETURN\_CODE [device\\_sendMacDataCommand](#) (byte taskID, byte[] functionID, byte[] data, bool macData, ref byte[] response)
- RETURN\_CODE [device\\_getFirmwareVersion](#) (ref string response)
- RETURN\_CODE [device\\_setUID](#) (string UID, int keyType, ref SpectrumInfo info)
- RETURN\_CODE [device\\_startRKI](#) ()
- RETURN\_CODE [device\\_updateFirmware](#) (byte[] firmwareData, string firmwareName, int encryptionType, byte[] keyBlob)
- RETURN\_CODE [device\\_setSpectrumProBDK](#) (string BDK)
- RETURN\_CODE [device\\_pollCardReader](#) (ref byte[] status)
- RETURN\_CODE [device\\_controlUserInterface](#) (byte[] values)
- RETURN\_CODE [msr\\_clearMSRData](#) ()
- RETURN\_CODE [msr\\_getMSRData](#) (ref IDTTransactionData card)
- RETURN\_CODE [config\\_getSerialNumber](#) (ref string response)
- RETURN\_CODE [config\\_getModelNumber](#) (ref string response)
- RETURN\_CODE [msr\\_startMSRSwipe](#) (int timeout)
- RETURN\_CODE [msr\\_cancelMSRSwipe](#) ()
- RETURN\_CODE [emv\\_getTerminalID](#) (ref string response)
- RETURN\_CODE [emv\\_setTerminalID](#) (string terminalID)
- RETURN\_CODE [emv\\_setTerminalMajorConfiguration](#) (int configuration)
- RETURN\_CODE [emv\\_getTerminalMajorConfiguration](#) (ref int configuration)
- RETURN\_CODE [emv\\_cancelTransaction](#) ()
- RETURN\_CODE [emv\\_clearTransactionLog](#) ()
- RETURN\_CODE [emv\\_startTransaction](#) (double amount, double amtOther, int exponent, int type, int timeout, byte[] tags, bool forceOnline)
- RETURN\_CODE [emv\\_authenticateTransaction](#) (byte[] updatedTLV)
- RETURN\_CODE [emv\\_completeTransaction](#) (bool commError, byte[] authCode, byte[] iad, byte[] tlvScripts, byte[] tlv)
- RETURN\_CODE [emv\\_callbackResponseLCD](#) (EMV\_LCD\_DISPLAY\_MODE type, byte selection)
- RETURN\_CODE [emv\\_callbackResponsePIN](#) (EMV\_PIN\_MODE type, byte[] KSN, byte[] PIN)
- RETURN\_CODE [emv\\_callbackResponseMSR](#) (byte[] MSR)
- RETURN\_CODE [emv\\_getEMVKernelVersion](#) (ref string response)
- RETURN\_CODE [emv\\_getEMVKernelCheckValue](#) (ref string response)
- RETURN\_CODE [emv\\_getEMVConfigurationCheckValue](#) (ref string response)
- RETURN\_CODE [emv\\_retrieveTransactionResult](#) (byte[] tags, ref IDTTransactionData tlv)
- RETURN\_CODE [device\\_rebootDevice](#) ()
- RETURN\_CODE [emv\\_removeApplicationData](#) (byte[] AID)
- RETURN\_CODE [emv\\_removeAllApplicationData](#) ()
- RETURN\_CODE [emv\\_removeCAPK](#) (byte[] capk)
- RETURN\_CODE [emv\\_removeAllCAPK](#) ()
- RETURN\_CODE [emv\\_removeCRL](#) (byte[] crlList)
- RETURN\_CODE [emv\\_removeAllCRL](#) ()
- RETURN\_CODE [emv\\_removeTerminalData](#) ()
- RETURN\_CODE [emv\\_retrieveAIDList](#) (ref byte[][] response)
- RETURN\_CODE [emv\\_retrieveApplicationData](#) (byte[] AID, ref byte[] tlv)
- RETURN\_CODE [emv\\_retrieveCAPK](#) (byte[] capk, ref byte[] key)
- RETURN\_CODE [emv\\_retrieveCAPKList](#) (ref byte[] keys)
- RETURN\_CODE [emv\\_retrieveCRLList](#) (ref byte[] list)
- RETURN\_CODE [emv\\_retrieveTerminalData](#) (ref byte[] tlv)
- RETURN\_CODE [emv\\_setApplicationData](#) (byte[] name, byte[] tlv)
- RETURN\_CODE [emv\\_setCAPK](#) (byte[] key)
- RETURN\_CODE [emv\\_setCRL](#) (byte[] list)
- RETURN\_CODE [emv\\_setTerminalData](#) (byte[] tlv)
- string [device\\_getResponseCodeString](#) (RETURN\_CODE eCode)
- RETURN\_CODE [device\\_getSpectrumProKSN](#) (int type, ref byte[] KSN)
- RETURN\_CODE [icc\\_powerOffICC](#) ()
- RETURN\_CODE [icc\\_powerOnICC](#) (ref byte[] ATR)
- RETURN\_CODE [icc\\_getICCReaderStatus](#) (ref byte status)
- RETURN\_CODE [icc\\_getICCStatus](#) (ref byte[] status, ref byte[] atr)

## Static Public Member Functions

- static bool [useSerialPort](#) (int port, bool isSRED)
- static bool [useSerialPort](#) (int port, int baud, bool isSRED)
- static bool [useUSB](#) ()
- static void [setCallback](#) (Callback my\_Callback)
- static void [setCallback](#) (IntPtr my\_Callback, SynchronizationContext context)
- static String [SDK\\_Version](#) ()
- static void [emv\\_autoAuthenticate](#) (bool authenticate)
- static void [emv\\_autoAuthenticate](#) (bool authenticate, byte[] tags)
- static void [emv\\_allowFallback](#) (bool allow)

## Properties

- static [IDT\\_SpectrumPro SharedController](#) [get]

### 15.2.1 Detailed Description

Class for [IDT\\_SpectrumPro](#) MSR/ICC reader

### 15.2.2 Member Function Documentation

#### 15.2.2.1 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.config\_getModelNumber ( ref string *response* )

Polls device for Model Number

##### Parameters

<i>response</i>	Returns Model Number
-----------------	----------------------

##### Returns

RETURN\_CODE: Values can be parsed with `device_getResponseCodeString`

#### 15.2.2.2 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.config\_getSerialNumber ( ref string *response* )

Polls device for Serial Number

##### Parameters

<i>response</i>	Returns Serial Number
-----------------	-----------------------

##### Returns

RETURN\_CODE: Values can be parsed with `device_getResponseCodeString`

#### 15.2.2.3 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_controlUserInterface ( byte[] *values* )

Control User Interface

Controls the User Interface: Display, Beep, LED

```
@param values Four bytes to control the user interface
Byte[0] = LCD Message
Messages 00-07 are normally controlled by the reader.
- 00h: Idle Message (Welcome)
- 01h: Present card (Please Present Card)
- 02h: Time Out or Transaction cancel (No Card)
- 03h: Transaction between reader and card is in the middle (Processing...)
- 04h: Transaction Pass (Thank You)
- 05h: Transaction Fail (Fail)
- 06h: Amount (Amount $ 0.00 Tap Card)
- 07h: Balance or Offline Available funds (Balance $ 0.00) Messages 08-0B are controlled by the terminal
- 08h: Insert or Swipe card (Use Chip & PIN)
- 09h: Try Again(Tap Again)
- 0Ah: Tells the customer to present only one card (Present 1 card only)
- 0Bh: Tells the customer to wait for authentication/authorization (Wait)
- FFh: indicates the command is setting the LED/Buzzer only.
Byte[1] = Beep Indicator
- 00h: No beep
- 01h: Single beep
- 02h: Double beep
- 03h: Three short beeps
- 04h: Four short beeps
- 05h: One long beep of 200 ms
- 06h: One long beep of 400 ms
- 07h: One long beep of 600 ms
- 08h: One long beep of 800 ms
Byte[2] = LED Number
- 00h: LED 0 (Power LED) 01h: LED 1
- 02h: LED 2
- 03h: LED 3
- FFh: All LEDs
Byte[3] = LED Status
- 00h: LED Off
- 01h: LED On
```

### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.4 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_getFirmwareVersion ( ref string *response* )

Polls device for Firmware Version

### Parameters

<i>response</i>	Response returned of Firmware Version
-----------------	---------------------------------------

### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.5 string IDTechSDK.IDT\_SpectrumPro.device\_getResponseCodeString ( RETURN\_CODE *eCode* )

Get the description of response result.

### Parameters

<i>eCode</i>	the response result.
--------------	----------------------

## Return values

<i>the</i>	string for description of response result
------------	---

- case 0: "no error, beginning task";
- case 1: "no response from reader";
- case 2: "invalid response data";
- case 3: "time out for task or CMD";
- case 4: "wrong parameter";
- case 5: "SDK is doing MSR or ICC task";
- case 6: "SDK is doing PINPad task";
- case 7: "SDK is doing CTLS task";
- case 8: "SDK is doing EMV task";
- case 9: "SDK is doing Other task";
- case 10: "err response or data";
- case 11: "no reader attached";
- case 12: "mono audio is enabled";
- case 13: "did connection";
- case 14: "audio volume is too low";
- case 15: "task or CMD be canceled";
- case 16: "UF wrong string format";
- case 17: "UF file not found";
- case 18: "UF wrong file format";
- case 19: "Attempt to contact online host failed";
- case 20: "Attempt to perform RKI failed";
- case 0x300: "Key Type(TDES) of Session Key is not same as the related Master Key.";
- case 0x400: "Related Key was not loaded.";
- case 0x500: "Key Same.";
- case 0x501: "Key is all zero";
- case 0x502: "TR-31 format error";
- case 0x702: "PAN is Error Key.";
- case 0x705: "No Internal MSR PAN (or Internal MSR PAN is erased timeout)";
- case 0X0C01: "Incorrect Frame Tag";
- case 0X0C02: "Incorrect Frame Type";
- case 0X0C03: "Unknown Frame Type";
- case 0X0C04: "Unknown Command";



- case 0X0C05: "Unknown Sub-Command";
- case 0X0C06: "CRC Error";
- case 0X0C07: "Failed";
- case 0X0C08: "Timeout";
- case 0X0C0A: "Incorrect Parameter";
- case 0X0C0B: "Command Not Supported";
- case 0X0C0C: "Sub-Command Not Supported";
- case 0X0C0D: "Parameter Not Supported / Status Abort Command";
- case 0X0C0F: "Sub-Command Not Allowed";
- case 0X0D01: "Incorrect Header Tag";
- case 0X0D02: "Unknown Command";
- case 0X0D03: "Unknown Sub-Command";
- case 0X0D04: "CRC Error in Frame";
- case 0X0D05: "Incorrect Parameter";
- case 0X0D06: "Parameter Not Supported";
- case 0X0D07: "Mal-formatted Data";
- case 0X0D08: "Timeout";
- case 0X0D0A: "Failed / NACK";
- case 0X0D0B: "Command not Allowed";
- case 0X0D0C: "Sub-Command not Allowed";
- case 0X0D0D: "Buffer Overflow (Data Length too large for reader buffer)";
- case 0X0D0E: "User Interface Event";
- case 0X0D11: "Communication type not supported, VT-1, burst, etc.";
- case 0X0D12: "Secure interface is not functional or is in an intermediate state.";
- case 0X0D13: "Data field is not mod 8";
- case 0X0D14: "Pad 0x80 not found where expected";
- case 0X0D15: "Specified key type is invalid";
- case 0X0D1: "Could not retrieve key from the SAM(InitSecureComm)";
- case 0X0D17: "Hash code problem";
- case 0X0D18: "Could not store the key into the SAM(InstallKey)";
- case 0X0D19: "Frame is too large";
- case 0X0D1A: "Unit powered up in authentication state but POS must resend the InitSecureComm command";
- case 0X0D1B: "The EEPROM may not be initialized because SecCommInterface does not make sense";
- case 0X0D1C: "Problem encoding APDU";
- case 0X0D20: "Unsupported Index(ILM) SAM Transceiver error – problem communicating with the SAM(Key Mgr)";

- case 0X0D2: "Unexpected Sequence Counter in multiple frames for single bitmap(ILM) Length error in data returned from the SAM(Key Mgr)";
- case 0X0D22: "Improper bit map(ILM)";
- case 0X0D23: "Request Online Authorization";
- case 0X0D24: "ViVOCard3 raw data read successful";
- case 0X0D25: "Message index not available(ILM) ViVOcomm activate transaction card type(ViVOcomm)";
- case 0X0D26: "Version Information Mismatch(ILM)";
- case 0X0D27: "Not sending commands in correct index message index(ILM)";
- case 0X0D28: "Time out or next expected message not received(ILM)";
- case 0X0D29: "ILM languages not available for viewing(ILM)";
- case 0X0D2A: "Other language not supported(ILM)";
- case 0X0D41: "Unknown Error from SAM";
- case 0X0D42: "Invalid data detected by SAM";
- case 0X0D43: "Incomplete data detected by SAM";
- case 0X0D44: "Reserved";
- case 0X0D45: "Invalid key hash algorithm";
- case 0X0D46: "Invalid key encryption algorithm";
- case 0X0D47: "Invalid modulus length";
- case 0X0D48: "Invalid exponent";
- case 0X0D49: "Key already exists";
- case 0X0D4A: "No space for new RID";
- case 0X0D4B: "Key not found";
- case 0X0D4C: "Crypto not responding";
- case 0X0D4D: "Crypto communication error";
- case 0X0D4E: "Module-specific error for Key Manager";
- case 0X0D4F: "All key slots are full (maximum number of keys has been installed)";
- case 0X0D50: "Auto-Switch OK";
- case 0X0D51: "Auto-Switch failed";
- case 0X0D90: "Account DUKPT Key not exist";
- case 0X0D91: "Account DUKPT Key KSN exhausted";
- case 0x0D00: "This Key had been loaded.";
- case 0x0E00: "Base Time was loaded.";
- case 0x0F00: "Encryption Or Decryption Failed.";
- case 0x1000: "Battery Low Warning (It is High Priority Response while Battery is Low.)";
- case 0x1800: "Send "Cancel Command" after send "Get Encrypted PIN" & "Get Numeric" & "Get Amount";
- case 0x1900: "Press "Cancel" key after send "Get Encrypted PIN" & "Get Numeric" & "Get Amount";

- case 0x30FF: "Security Chip is not connect";
- case 0x3000: "Security Chip is deactivation & Device is In Removal Legally State.";
- case 0x3101: "Security Chip is activation & Device is In Removal Legally State.";
- case 0x5500: "No Admin DUKPT Key.";
- case 0x5501: "Admin DUKPT Key STOP.";
- case 0x5502: "Admin DUKPT Key KSN is Error.";
- case 0x5503: "Get Authentication Code1 Failed.";
- case 0x5504: "Validate Authentication Code Error.";
- case 0x5505: "Encrypt or Decrypt data failed.";
- case 0x5506: "Not Support the New Key Type.";
- case 0x5507: "New Key Index is Error.";
- case 0x5508: "Step Error.";
- case 0x5509: "KSN Error";
- case 0x550A: "MAC Error.";
- case 0x550B: "Key Usage Error.";
- case 0x550C: "Mode Of Use Error.";
- case 0x550F: "Other Error.";
- case 0x6000: "Save or Config Failed / Or Read Config Error.";
- case 0x6200: "No Serial Number.";
- case 0x6900: "Invalid Command - Protocol is right, but task ID is invalid.";
- case 0x6A01: "Unsupported Command – Protocol and task ID are right, but command is invalid – In this State";
- case 0x6A00: "Unsupported Command - Protocol and task ID are right, but command is invalid.";
- case 0x6B00: "Unknown parameter in command - Protocol task ID and command are right, but parameter is invalid.";
- case 0x6C00: "Unknown parameter in command – Protocol task ID and command are right, but length is out of the requirement.";
- case 0x7200: "Device is suspend (MKSK suspend or press password suspend).";
- case 0x7300: "PIN DUKPT is STOP (21 bit 1).";
- case 0x7400: "Device is Busy.";
- case 0xE100: "Can not enter sleep mode";
- case 0xE200: "File has existed";
- case 0xE300: "File has not existed";
- case 0xE313: "IO line low -- Card error after session start";
- case 0xE400: "Open File Error";
- case 0xE500: "SmartCard Error";
- case 0xE600: "Get MSR Card data is error";

- case 0xE700: "Command time out";
- case 0xE800: "File read or write is error";
- case 0xE900: "Active 1850 error!";
- case 0xEA00: "Load bootloader error";
- case 0xEF00: "Protocol Error- STX or ETX or check error.";
- case 0xEB00: "Picture is not exist";
- case 0x2C02: "No Microprocessor ICC seated";
- case 0x2C06: "no card seated to request ATR";
- case 0x2D01: "Card Not Supported,";
- case 0x2D03: "Card Not Supported, wants CRC";
- case 0x690D: "Command not supported on reader without ICC support";
- case 0x8100: "ICC error time out on power-up";
- case 0x8200: "invalid TS character received - Wrong operation step";
- case 0x8300: "Decode MSR Error";
- case 0x8400: "TriMagII no Response";
- case 0x8500: "No Swipe MSR Card";
- case 0x8510: "No Financial Card";
- case 0x8600: "Unsupported F, D, or combination of F and D";
- case 0x8700: "protocol not supported EMV TD1 out of range";
- case 0x8800: "power not at proper level";
- case 0x8900: "ATR length too long";
- case 0x8B01: "EMV invalid TA1 byte value";
- case 0x8B02: "EMV TB1 required";
- case 0x8B03: "EMV Unsupported TB1 only 00 allowed";
- case 0x8B04: "EMV Card Error, invalid BWI or CWI";
- case 0x8B06: "EMV TB2 not allowed in ATR";
- case 0x8B07: "EMV TC2 out of range";
- case 0x8B08: "EMV TC2 out of range";
- case 0x8B09: "per EMV96 TA3 must be > 0xF";
- case 0x8B10: "ICC error on power-up";
- case 0x8B11: "EMV T=1 then TB3 required";
- case 0x8B12: "Card Error, invalid BWI or CWI";
- case 0x8B13: "Card Error, invalid BWI or CWI";
- case 0x8B17: "EMV TC1/TB3 conflict-";
- case 0x8B20: "EMV TD2 out of range must be T=1";
- case 0x8C00: "TCK error";

- case 0xA304: "connector has no voltage setting";
- case 0xA305: "ICC error on power-up invalid (SBLK(IFSD) exchange";
- case 0xE301: "ICC error after session start";
- case 0xFF00: "Request to go online";
- case 0xFF01: "EMV: Accept the offline transaction";
- case 0xFF02: "EMV: Decline the offline transaction";
- case 0xFF03: "EMV: Accept the online transaction";
- case 0xFF04: "EMV: Decline the online transaction";
- case 0xFF05: "EMV: Application may fallback to magstripe technology";
- case 0xFF06: "EMV: ICC detected tah the conditions of use are not satisfied";
- case 0xFF07: "EMV: ICC didn't accept transaction";
- case 0xFF08: "EMV: Transaction was cancelled";
- case 0xFF09: "EMV: Application was not selected by kernel or ICC format error or ICC missing data error";
- case 0xFF0A: "EMV: Transaction is terminated";
- case 0xFF0B: "EMV: Other EMV Error";
- case 0xFFFF: "NO RESPONSE";
- case 0xF002: "ICC communication timeout";
- case 0xF003: "ICC communication Error";
- case 0xF00F: "ICC Card Seated and Highest Priority, disable MSR work request";
- case 0xF200: "AID List / Application Data is not exist";
- case 0xF201: "Terminal Data is not exist";
- case 0xF202: "TLV format is error";
- case 0xF203: "AID List is full";
- case 0xF204: "Any CA Key is not exist";
- case 0xF205: "CA Key RID is not exist";
- case 0xF206: "CA Key Index it not exist";
- case 0xF207: "CA Key is full";
- case 0xF208: "CA Key Hash Value is Error";
- case 0xF209: "Transaction format error";
- case 0xF20A: "The command will not be processing";
- case 0xF20B: "CRL is not exist";
- case 0xF20C: "CRL number exceed max number";
- case 0xF20D: "Amount,Other Amount,Trasaction Type are missing";
- case 0xF20E: "The Identification of algorithm is mistake";
- case 0xF20F: "No Financial Card";
- case 0xF210: "In Encrypt Result state, TLV total Length is greater than Max Length";

- case 0x1001: "INVALID ARG";
- case 0x1002: "FILE\_OPEN\_FAILED";
- case 0x1003: "FILE OPERATION\_FAILED";
- case 0x2001: "MEMORY\_NOT\_ENOUGH";
- case 0x3002: "SMARTCARD\_FAIL";
- case 0x3003: "SMARTCARD\_INIT\_FAILED";
- case 0x3004: "FALLBACK\_SITUATION";
- case 0x3005: "SMARTCARD\_ABSENT";
- case 0x3006: "SMARTCARD\_TIMEOUT";
- case 0x5001: "EMV\_PARSING\_TAGS\_FAILED";
- case 0x5002: "EMV\_DUPLICATE\_CARD\_DATA\_ELEMENT";
- case 0x5003: "EMV\_DATA\_FORMAT\_INCORRECT";
- case 0x5004: "EMV\_NO\_TERM\_APP";
- case 0x5005: "EMV\_NO\_MATCHING\_APP";
- case 0x5006: "EMV\_MISSING\_MANDATORY\_OBJECT";
- case 0x5007: "EMV\_APP\_SELECTION\_RETRY";
- case 0x5008: "EMV\_GET\_AMOUNT\_ERROR";
- case 0x5009: "EMV\_CARD\_REJECTED";
- case 0x5010: "EMV\_AIP\_NOT\_RECEIVED";
- case 0x5011: "EMV\_AFL\_NOT\_RECEIVED";
- case 0x5012: "EMV\_AFL\_LEN\_OUT\_OF\_RANGE";
- case 0x5013: "EMV\_SFI\_OUT\_OF\_RANGE";
- case 0x5014: "EMV\_AFL\_INCORRECT";
- case 0x5015: "EMV\_EXP\_DATE\_INCORRECT";
- case 0x5016: "EMV\_EFF\_DATE\_INCORRECT";
- case 0x5017: "EMV\_ISS\_COD\_TBL\_OUT\_OF\_RANGE";
- case 0x5018: "EMV\_CRYPTOGAM\_TYPE\_INCORRECT";
- case 0x5019: "EMV\_PSE\_NOT\_SUPPORTED\_BY\_CARD";
- case 0x5020: "EMV\_USER\_SELECTED\_LANGUAGE";
- case 0x5021: "EMV\_SERVICE\_NOT\_ALLOWED";
- case 0x5022: "EMV\_NO\_TAG\_FOUND";
- case 0x5023: "EMV\_CARD\_BLOCKED";
- case 0x5024: "EMV\_LEN\_INCORRECT";
- case 0x5025: "CARD\_COM\_ERROR";
- case 0x5026: "EMV\_TSC\_NOT\_INCREASED";
- case 0x5027: "EMV\_HASH\_INCORRECT";

- case 0x5028: "EMV\_NO\_ARC";
- case 0x5029: "EMV\_INVALID\_ARC";
- case 0x5030: "EMV\_NO\_ONLINE\_COMM";
- case 0x5031: "TRAN\_TYPE\_INCORRECT";
- case 0x5032: "EMV\_APP\_NO\_SUPPORT";
- case 0x5033: "EMV\_APP\_NOT\_SELECT";
- case 0x5034: "EMV\_LANG\_NOT\_SELECT";
- case 0x5035: "EMV\_NO\_TERM\_DATA";
- case 0x6001: "CVM\_TYPE\_UNKNOWN";
- case 0x6002: "CVM\_AIP\_NOT\_SUPPORTED";
- case 0x6003: "CVM\_TAG\_8E\_MISSING";
- case 0x6004: "CVM\_TAG\_8E\_FORMAT\_ERROR";
- case 0x6005: "CVM\_CODE\_IS\_NOT\_SUPPORTED";
- case 0x6006: "CVM\_COND\_CODE\_IS\_NOT\_SUPPORTED";
- case 0x6007: "NO\_MORE\_CVM";
- case 0x6008: "PIN\_BYPASSED\_BEFORE";
- case 0x7001: "PK\_BUFFER\_SIZE\_TOO\_BIG";
- case 0x7002: "PK\_FILE\_WRITE\_ERROR";
- case 0x7003: "PK\_HASH\_ERROR";
- case 0x8001: "NO\_CARD\_HOLDER\_CONFIRMATION";
- case 0x8002: "GET\_ONLINE\_PIN";
- case 0xD000: "Data not exist";
- case 0xD001: "Data access error";
- case 0xD100: "RID not exist";
- case 0xD101: "RID existed";
- case 0xD102: "Index not exist";
- case 0xD200: "Maximum exceeded";
- case 0xD201: "Hash error";
- case 0xD205: "System Busy";
- case 0x0E01: "Unable to go online";
- case 0x0E02: "Technical Issue";
- case 0x0E03: "Declined";
- case 0x0E04: "Issuer Referral transaction";
- case 0x0F01: "Decline the online transaction";
- case 0x0F02: "Request to go online";
- case 0x0F03: "Transaction is terminated";

- case 0x0F05: "Application was not selected by kernel or ICC format error or ICC missing data error";
- case 0x0F07: "ICC didn't accept transaction";
- case 0x0F0A: "Application may fallback to magstripe technology";
- case 0x0F0C: "Transaction was cancelled";
- case 0x0F0D: "Timeout";
- case 0x0F0F: "Other EMV Error";
- case 0x0F10: "Accept the offline transaction";
- case 0x0F11: "Decline the offline transaction";
- case 0x0F21: "ICC detected tah the conditions of use are not satisfied";
- case 0x0F22: "No app were found on card matching terminal configuration";
- case 0x0F23: "Terminal file does not exist";
- case 0x0F24: "CAPK file does not exist";
- case 0x0F25: "CRL Entry does not exist";
- case 0x0FFE: "code when blocking is disabled";
- case 0x0FFF: "code when command is not applicable on the selected device";
- case 0xF005: "ICC Encrypted C-APDU Data Structure Length Error Or Format Error.";
- case 0xBBE0: "CM100 Success";
- case 0xBBE1: "CM100 Parameter Error";
- case 0xBBE2: "CM100 Low Output Buffer";
- case 0xBBE3: "CM100 Card Not Found";
- case 0xBBE4: "CM100 Collision Card Exists";
- case 0xBBE5: "CM100 Too Many Cards Exist";
- case 0xBBE6: "CM100 Saved Data Does Not Exist";
- case 0xBBE8: "CM100 No Data Available";
- case 0xBBE9: "CM100 Invalid CID Returned";
- case 0xBBEA: "CM100 Invalid Card Exists";
- case 0xBBEC: "CM100 Command Unsupported";
- case 0xBBED: "CM100 Error In Command Process";
- case 0xBBEE: "CM100 Invalid Command";
- case 0X9031: "Unknown command";
- case 0X9032: "Wrong parameter (such as the length of the command is incorrect)";
- case 0X9038: "Wait (the command couldn't be finished in BWT)";
- case 0X9039: "Busy (a previously command has not been finished)";
- case 0X903A: "Number of retries over limit";
- case 0X9040: "Invalid Manufacturing system data";
- case 0X9041: "Not authenticated";



- case 0X9042: "Invalid Master DUKPT Key";
- case 0X9043: "Invalid MAC Key";
- case 0X9044: "Reserved for future use";
- case 0X9045: "Reserved for future use";
- case 0X9046: "Invalid DATA DUKPT Key";
- case 0X9047: "Invalid PIN Pairing DUKPT Key";
- case 0X9048: "Invalid DATA Pairing DUKPT Key";
- case 0X9049: "No nonce generated";
- case 0X9949: "No GUID available. Perform getVersion first.";
- case 0X9950: "MAC Calculation unsuccessful. Check BDK value.";
- case 0X904A: "Not ready";
- case 0X904B: "Not MAC data";
- case 0X9050: "Invalid Certificate";
- case 0X9051: "Duplicate key detected";
- case 0X9052: "AT checks failed";
- case 0X9053: "TR34 checks failed";
- case 0X9054: "TR31 checks failed";
- case 0X9055: "MAC checks failed";
- case 0X9056: "Firmware download failed";
- case 0X9060: "Log is full";
- case 0X9061: "Removal sensor unengaged";
- case 0X9062: "Any hardware problems";
- case 0X9070: "ICC communication timeout";
- case 0X9071: "ICC data error (such check sum error)";
- case 0X9072: "Smart Card not powered up";

#### 15.2.2.6 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_getSpectrumProKSN ( int type, ref byte[] KSN )

Get DUKPT KSN

Returns the KSN for the provided key index

**Parameters**

<i>type</i>	Key type: <ul style="list-style-type: none"> <li>• 0: Key Encryption Key (Master Key or KEK)</li> <li>• 2: Data Encryption Key (DEK)</li> <li>• 5: MAC Key (MAK)</li> <li>• 10: RKL Key Encryption Key (REK)</li> <li>• 20: HSM DUKPT Key</li> </ul>
<i>KSN</i>	Key Serial Number

**Returns**

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

**15.2.2.7 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_pollCardReader ( ref byte[] status )****Poll Card Reader**

Provides information about the state of the Card Reader

@param status Six bytes indicating card reader information

Byte 0:

- Bit 0: Device Manufacturing CA data valid
- Bit 1: Device Manufacturing Secure data valid
- Bit 2: HOST\_CR\_MASTER\_DUKPT Key valid
- Bit 3: HOST\_CR\_MAC Keys valid (Authenticated)
- Bit 4: RFU
- Bit 5: RFU
- Bit 6: DATA\_DUKPT Key Valid
- Bit 7: Key is initialized (MFK and RSA Key pairs)

Byte 1:

- Bit 0: Firmware Key Valid
- Bit 1: RFU
- Bit 2: CR\_PINPAD\_MASTER\_DUKPT Key valid
- Bit 3: CR\_PINPAD\_MAC Keys valid (Authenticated)
- Bit 4: DATA Pairing DUKPT Key valid
- Bit 5: PIN Pairing DUKPT Key Valid
- Bit 6: RFU
- Bit 7: RFU

Byte 2:

- Bit 0: RFU
- Bit 1: Tamper Switch #1 Error
- Bit 2: Battery Backup Error
- Bit 3: Temperature Error
- Bit 4: Voltage Sensor Error
- Bit 5: Firmware Authentication Error
- Bit 6: Tamper Switch #2 Error
- Bit 7: Removal Tamper Error

Byte 3:

- Battery Voltage (example 0x32 = 3.2V, 0x24 = 2.4V)

Byte 4:

- Bit 0: Log is Full
- Bit 1: Mag Data Present
- Bit 2: Card Insert
- Bit 3: Removal Sensor connected
- Bit 4: Card Seated
- Bit 5: Latch Mechanism Active

- Bit 6: Removal Sensor Active
- Bit 7: Tamper Detector Active

Byte 5:

- Bit 0: SAM Available
- Bit 1: Chip Card Reader Available
- Bit 2: Host Connected
- Bit 3: Contactless Available
- Bit 4: PINPAD connected
- Bit 5: MSR Header connected
- Bit 6: RFU
- Bit 7: Production Unit

#### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.8 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_rebootDevice ( )

##### Reboot Device

Executes a command to restart the device.

- Card data is cleared, resetting card status bits.
- Response data of the previous command is cleared.
- Resetting firmware.

#### Returns

RETURN\_CODE: Values can be parsed with `device_getResponseCodeString`

#### 15.2.2.9 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_sendDataCommand ( string cmd, bool calcLRC, ref byte[] response )

##### Send a Command to device

Sends a command to the device .

#### Parameters

<i>cmd</i>	Hex string representation of command to execute
<i>calcLRC</i>	If TRUE, this will wrap command with start/length/lrc/sum/end: '{STX}{Len_Low}{Len_High} data {CheckLRC} {CheckSUM} {ETX}'
<i>response</i>	Response data

#### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.10 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_sendMacDataCommand ( byte taskId, byte[] functionID, byte[] data, bool macData, ref byte[] response )

##### Send a Command to device with MAC

Sends a command to the device with Mac calculation (if specified).

## Parameters

<i>taskID</i>	Task ID to execute
<i>functionID</i>	Function ID to execute (one or more bytes)
<i>data</i>	Data to send (if any)
<i>macData</i>	If <code>TRUE</code> , this will include the MAC with the command
<i>response</i>	Response data

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.11 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_setSpectrumProBDK ( string *BDK* )

## Set Spectrum Pro BDK

Tells the SDK which BDK to use when generating MAC values for Spectrum Pro commands. SDK default value is "0123456789ABCDEFDCBA9876543210"

## Parameters

<i>BDK</i>	Value of BDK, 32 ASCII characters representing 16 HEX values
------------	--

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.12 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_setUID ( string *UID*, int *keyType*, ref SpectrumInfo *info* )

## Set Spectrum Pro UID

Sets the UID for the Spectrum Pro. SDK default value is "0001020304050607"

## Parameters

<i>UID</i>	Value of UID, 16 ASCII characters representing 8 HEX values
<i>keyType</i>	Key type <ul style="list-style-type: none"> <li>• 0 : Symmetric only</li> <li>• 1 : Asymmetric</li> <li>• 2 : Both Symmetric and Asymmetric</li> </ul>
<i>info</i>	Spectrum Pro return info

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

## 15.2.2.13 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_startRKI ( )

## Start Remote Key Injection

Starts a remote key injection request with IDTech RKI servers.

#### Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

**15.2.2.14** RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.device\_updateFirmware ( byte[] *firmwareData*, string *firmwareName*, int *encryptionType*, byte[] *keyBlob* )

#### Update Firmware

Updates the firmware of the Spectrum Pro K21 HUB or Maxq1050.

#### Parameters

<i>firmwareData</i>	Signed binary data of a firmware file provided by IDTech
<i>firmwareName</i>	Firmware name. Must be one of the following two strings (with appropriate version information) <ul style="list-style-type: none"> <li>• "SP K21 APP Vx.xx.xxx"</li> <li>• "SP MAX APP Vx.xx.xxx"</li> </ul>
<i>encryptionType</i>	Encryption type <ul style="list-style-type: none"> <li>• 0 : Plaintext</li> <li>• 1 : TDES ECB, PKCS#5 padding</li> <li>• 2 : TDES CBC, PKCS#5, IV is all 0</li> </ul>
<i>keyBlob</i>	Encrypted firmware session key blob, TR-31 Rev B, wrapped by FW Key (Optional, none if firmware is plaintext)

#### Returns

RETURN\_CODE: Values can be parsed with errorCode.getErrorString()

Firmware update status is returned in the callback with the following values: sender = SPECTRUM\_PRO state = DeviceState.FirmwareUpdate data = File Progress. Two bytes, with byte[0] = current block, and byte[1] = total blocks. 0x0310 = block 3 of 16 transactionResultCode:

- RETURN\_CODE\_DO\_SUCCESS = Firmware Update Completed Successfully
- RETURN\_CODE\_BLOCK\_TRANSFER\_SUCCESS = Current block transferred successfully
- Any other return code represents an error condition

**15.2.2.15** static void IDTechSDK.IDT\_SpectrumPro.emv\_allowFallback ( bool *allow* ) [static]

Allow fallback for EMV transactions. Default is TRUE

#### Parameters

<i>allow</i>	TRUE = allow fallback, FALSE = don't allow fallback
--------------	---

#### 15.2.2.16 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_authenticateTransaction ( byte[] updatedTLV )

##### Authenticate EMV Transaction Request

Authenticates the EMV transaction for an ICC card. Execute this after receiving response with result code 0x10 to emv\_startTransaction

The tags will be returned in the callback routine.

##### Parameters

<i>updatedTLV</i>	TLV stream that can be used to update the following values: <ul style="list-style-type: none"> <li>• 9F02: Amount</li> <li>• 9F03: Other amount</li> <li>• 9C: Transaction type</li> <li>• 5F57: Account type In addition tag DFEE1A can be sent to specify tag list to include in results. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37</li> </ul>
-------------------	---

##### Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

#### 15.2.2.17 static void IDTechSDK.IDT\_SpectrumPro.emv\_autoAuthenticate ( bool authenticate ) [static]

Enables authenticate for EMV transactions. If a startEMVTransaction results in code 0x0010 (start transaction success), then emv\_authenticateEMVTransaction can automatically execute if parameter is set to TRUE

##### Parameters

<i>authenticate</i>	TRUE = auto authenticate, FALSE = manually authenticate
---------------------	---

#### 15.2.2.18 static void IDTechSDK.IDT\_SpectrumPro.emv\_autoAuthenticate ( bool authenticate, byte[] tags ) [static]

Enables authenticate for EMV transactions. If a startEMVTransaction results in code 0x0010 (start transaction success), then emv\_authenticateEMVTransaction can automatically execute if parameter is set to TRUE

##### Parameters

<i>authenticate</i>	TRUE = auto authenticate, FALSE = manually authenticate
<i>tags</i>	Tags to pass during authentication stage;

#### 15.2.2.19 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_callbackResponseLCD ( EMV\_LCD\_DISPLAY\_MODE type, byte selection )

##### Callback Response LCD Display

Provides menu selection responses to the kernel after a callback was received with DeviceState.EMVCallback, and callbackType = EMV\_CALLBACK\_TYPE.EMV\_CALLBACK\_TYPE\_LCD, and lcd\_displayMode = EMV\_LCD\_DISPLAY\_MODE\_MENU, EMV\_LCD\_DISPLAY\_MODE\_PROMPT, or EMV\_LCD\_DISPLAY\_MODE\_LANGUAGE

**\_SELECT****Parameters**

<i>type</i>	If Cancel key pressed during menu selection, then value is EMV_LCD_DISPLAY_MODE_CANCEL. Otherwise, value can be EMV_LCD_DISPLAY_MODE_MENU, EMV_LCD_DISPLAY_MODE_PROMPT, or EMV_LCD_DISPLAY_MODE_LANGUAGE_SELECT
<i>selection</i>	If type = EMV_LCD_DISPLAY_MODE_MENU or EMV_LCD_DISPLAY_MODE_LANGUAGE_SELECT, provide the selection ID line number. Otherwise, if type = EMV_LCD_DISPLAY_MODE_PROMPT supply either 0x43 ('C') for Cancel, or 0x45 ('E') for Enter/accept

**Returns**

RETURN\_CODE: Values can be parsed with errorCode.getErrorString()

**15.2.2.20 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_callbackResponseMSR ( byte[] MSR )****Callback Response MSR Entry**

Provides MSR information to kernel after a callback was received with DeviceState.EMVCallback, and callbackType = EMV\_CALLBACK\_MSR

**Parameters**

<i>MSR</i>	Swiped track data
------------	-------------------

**Returns**

RETURN\_CODE: Values can be parsed with errorCode.getErrorString()

**15.2.2.21 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_callbackResponsePIN ( EMV\_PIN\_MODE type, byte[] KSN, byte[] PIN )****Callback Response PIN Entry**

Provides (or cancels) PIN entry information to kernel after a callback was received with DeviceState.EMVCallback, and callbackType = EMV\_CALLBACK\_TYPE\_PINPAD

**Parameters**

<i>type</i>	If Cancel key pressed during PIN entry, then value is EMV_PIN_MODE_CANCEL. Otherwise, value can be EMV_PIN_MODE_ONLINE_DUKPT, EMV_PIN_MODE_ONLINE_MKSK, or EMV_PIN_MODE_OFFLINE
<i>KSN</i>	If enciphered PIN, this is either the PIN DUKPT Key (EMV_PIN_MODE_ONLINE_DUKPT) or PIN Session Key (EMV_PIN_MODE_ONLINE_MKSK), or PIN Pairing DUKPT key (EMV_PIN_MODE_OFFLINE)
<i>PIN</i>	If enciphered PIN, this is encrypted PIN block. If device does not implement pairing function, this is plaintext PIN

**Returns**

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

#### 15.2.2.22 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_cancelTransaction ( )

Cancel EMV Transaction

Cancels the currently executing EMV transaction.

Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.23 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_clearTransactionLog ( )

Clear Transaction Log

Clears the transaction log.

Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.24 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_completeTransaction ( bool *commError*, byte[] *authCode*, byte[] *iad*, byte[] *tlvScripts*, byte[] *tlv* )

Complete EMV Transaction Request

Completes the EMV transaction for an ICC card when online authorization request is received from `emv_authenticateTransaction`

The tags will be returned in the callback routine.

Parameters

<i>commError</i>	Communication error with host. Set to TRUE if host was unreachable, or FALSE if host response received. If Communication error, <i>authCode</i> , <i>iad</i> , <i>tlvScripts</i> can be null.
<i>authCode</i>	Authorization code from host. Two bytes. Example 0x3030. (Tag value 8A). Required
<i>iad</i>	Issuer Authentication Data, if any. Example 0x11223344556677883030 (tag value 91).
<i>tlvScripts</i>	71/72 scripts, if any
<i>tlv</i>	Additional TVL data to return with transaction results (if any)

Returns

RETURN\_CODE: Values can be parsed with `device_getResponseCodeString`

#### 15.2.2.25 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_getEMVConfigurationCheckValue ( ref string *response* )

Get EMV Kernel configuration check value info

Parameters

<i>response</i>	Response returned of Kernel configuration check value info
-----------------	--



**Returns**

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

**15.2.2.26 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_getEMVKernelCheckValue ( ref string *response* )**

Get EMV Kernel check value info

**Parameters**

<i>response</i>	Response returned of Kernel check value info
-----------------	--

**Returns**

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

**15.2.2.27 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_getEMVKernelVersion ( ref string *response* )**

Polls device for EMV Kernel Version

**Parameters**

<i>response</i>	Response returned of Kernel Version
-----------------	-------------------------------------

**Returns**

RETURN\_CODE: Values can be parsed with errorCode.getErrorString()

**15.2.2.28 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_getTerminalID ( ref string *response* )**

Gets the terminal ID as printable characters .

**Parameters**

<i>response</i>	Terminal ID
-----------------	-------------

**Returns**

RETURN\_CODE: Values can be parsed with errorCode.getErrorString()

**15.2.2.29 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_getTerminalMajorConfiguration ( ref int *configuration* )**

Gets the terminal major configuration in ICS .

**Parameters**

<i>configuration</i>	A configuration value, range 1-4 <ul style="list-style-type: none"> <li>• 1 = 1C</li> <li>• 2 = 2C</li> <li>• 3 = 3C</li> <li>• 4 = 4C</li> </ul>
----------------------	---

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.30 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeAllApplicationData ( )

Remove All Application Data

Removes all the Application Data

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.31 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeAllCAPK ( )

Remove All Certificate Authority Public Key

Removes all the CAPK

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.32 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeAllCRL ( )

Remove All Certificate Revocation List Entries

Removes all CRLEntry entries

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.33 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeApplicationData ( byte[] AID )

Remove Application Data by AID

Removes the Application Data as specified by the AID name passed as a parameter

**Parameters**

<i>AID</i>	Name of ApplicationID Must be between 5 and 16 bytes
------------	--

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

**15.2.2.34 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeCAPK ( byte[] *capk* )**

Remove Certificate Authority Public Key

Removes the CAPK as specified by the RID/Index

**Parameters**

<i>capk</i>	6 byte CAPK = 5 bytes RID + 1 byte INDEX
-------------	--

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

**15.2.2.35 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeCRL ( byte[] *crlList* )**

Remove Certificate Revocation List Entries

Removes CRL entries as specified by the RID and Index and serial number passed as 9 bytes

**Parameters**

<i>crlList</i>	containing the list of CRL to remove: [CRL1][CRL2]...[CRLn] where each [CRL] is 9 bytes: [5 bytes RID][1 byte CAPK Index][3 bytes serial number]
----------------	--

**Returns**

RETURN\_CODE: Values can be parsed with `device.getResponseCodeString`

**15.2.2.36 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_removeTerminalData ( )**

Remove Terminal Data

Removes the Terminal Data

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

**15.2.2.37 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveAIDList ( ref byte *response*[[ ] ] )**

Retrieve AID list

Returns all the AID names installed on the terminal.

**Parameters**

<i>response</i>	array of AID name byte arrays
-----------------	-------------------------------

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.38 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveApplicationData ( byte[] AID, ref byte[] tlv )

Retrieve Application Data by AID

Retrieves the Application Data as specified by the AID name passed as a parameter.

**Parameters**

<i>AID</i>	Name of ApplicationID. Must be between 5 and 16 bytes
<i>tlv</i>	The TLV elements of the requested AID

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.39 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveCAPK ( byte[] capk, ref byte[] key )

Retrieve Certificate Authority Public Key

Retrieves the CAPK as specified by the RID/Index passed as a parameter.

**Parameters**

<i>capk</i>	6 bytes CAPK = 5 bytes RID + 1 byte Index
<i>key</i>	<p>Response returned as a CAKey format: [5 bytes RID][1 byte Index][1 byte Hash Algorithm][1 byte Encryption Algorithm] [20 bytes HashValue][4 bytes Public Key Exponent][2 bytes Modulus Length][Variable bytes Modulus] Where:</p> <ul style="list-style-type: none"> <li>• Hash Algorithm: The only algorithm supported is SHA-1. The value is set to 0x01</li> <li>• Encryption Algorithm: The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to 0x01.</li> <li>• HashValue: Which is calculated using SHA-1 over the following fields: RID &amp; Index &amp; Modulus &amp; Exponent</li> <li>• Public Key Exponent: Actually, the real length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03), or 65537 (Format is 0x00 01 00 01)</li> <li>• Modulus Length: LenL LenH Indicated the length of the next field.</li> <li>• Modulus: This is the modulus field of the public key. Its length is specified in the field above.</li> </ul>

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.40 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveCAPKList ( ref byte[] keys )

Retrieve the Certificate Authority Public Key list

Returns all the CAPK RID and Index installed on the terminal.

## Parameters

<i>keys</i>	[key1][key2]...[keyn], each key 6 bytes where key = 5 bytes RID + 1 byte index
-------------	--

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.41 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveCRLList ( ref byte[] *list* )

Retrieve the Certificate Revocation List

Returns the CRL entries on the terminal.

## Parameters

<i>list</i>	[CRL1][CRL2]...[CRLn], each CRL 9 bytes where CRL = 5 bytes RID + 1 byte index + 3 bytes serial number
-------------	--

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.42 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveTerminalData ( ref byte[] *tlv* )

Retrieve Terminal Data

Retrieves the Terminal Data.

## Parameters

<i>tlv</i>	Response returned as a TLV
------------	----------------------------

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.43 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_retrieveTransactionResult ( byte[] *tags*, ref IDTTransactionData *tlv* )

Retrieve Transaction Results

Retrieves specified EMV tags from the currently executing transaction.

## Parameters

<i>tags</i>	Tags to be retrieved. Example 0x9F028A will retrieve tags 9F02 and 8A
<i>tlv</i>	All requested tags returned as unencrypted, encrypted and masked TLV data in IDTTransactionData object

## Returns

RETURN\_CODE: Values can be parsed with `device.getResponseCodeString`

#### 15.2.2.44 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_setApplicationData ( byte[] name, byte[] tlv )

Set Application Data by AID

Sets the Application Data as specified by the application name and TLV data

##### Parameters

<i>name</i>	Application name, 10-32 ASCII hex characters representing 5-16 bytes Example "a0000000031010"
<i>tlv</i>	Application data in TLV format

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.45 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_setCAPK ( byte[] key )

Set Certificate Authority Public Key

Sets the CAPK as specified by the CAKey structure

##### Parameters

<i>key</i>	<p>CAKey format: [5 bytes RID][1 byte Index][1 byte Hash Algorithm][1 byte Encryption Algorithm][20 bytes HashValue][4 bytes Public Key Exponent][2 bytes Modulus Length][Variable bytes Modulus] Where:</p> <ul style="list-style-type: none"> <li>• Hash Algorithm: The only algorithm supported is SHA-1. The value is set to 0x01</li> <li>• Encryption Algorithm: The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to 0x01.</li> <li>• HashValue: Which is calculated using SHA-1 over the following fields: RID &amp; Index &amp; Modulus &amp; Exponent</li> <li>• Public Key Exponent: Actually, the real length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03), or 65537 (Format is 0x00 01 00 01)</li> <li>• Modulus Length: LenL LenH Indicated the length of the next field.</li> <li>• Modulus: This is the modulus field of the public key. Its length is specified in the field above.</li> </ul>
------------	--

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.46 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_setCRL ( byte[] list )

Set Certificate Revocation List

Sets the CRL

##### Parameters

<i>list</i>	CRL Entries containing the RID, Index, and serial numbers to set [CRL1][CRL2]...[CRLn] where each [CRL] is 9 bytes: [5 bytes RID][1 byte CAPK Index][3 bytes serial number]
-------------	---

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.47 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.env\_setTerminalData ( byte[] *t/v* )

## Set Terminal Data

Sets the Terminal Data as specified by the TerminalData structure passed as a parameter

## Parameters

<i>data</i>	TerminalData configuration file
-------------	---------------------------------

## Return values

<i>RETURN_CODE</i>	Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with <code>IDT_BTPay::device_getResponseCodeString:()</code>
--------------------	---

15.2.2.48 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.env\_setTerminalID ( string *terminalID* )

Sets the terminal ID as printable characters .

## Parameters

<i>terminalID</i>	Terminal ID to set
-------------------	--------------------

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.49 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.env\_setTerminalMajorConfiguration ( int *configuration* )

Sets the terminal major configuration in ICS .

## Parameters

<i>configuration</i>	A configuration value, range 1-4 <ul style="list-style-type: none"> <li>• 1 = 1C</li> <li>• 2 = 2C</li> <li>• 3 = 3C</li> <li>• 4 = 4C</li> </ul>
----------------------	---

## Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.50 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.emv\_startTransaction ( double *amount*, double *amtOther*, int *exponent*, int *type*, int *timeout*, byte[] *tags*, bool *forceOnline* )

Start EMV Transaction Request

Authorizes the EMV transaction for an ICC card

The tags will be returned in the callback routine.

##### Parameters

<i>amount</i>	Transaction amount value (tag value 9F02)
<i>amtOther</i>	Other amount value, if any (tag value 9F03)
<i>exponent</i>	Number of characters after decimile point
<i>type</i>	Transaction type (tag value 9C).
<i>timeout</i>	Timeout value in seconds.
<i>tags</i>	Any other tags to be included in the request. Passed as a string. Example, tag 9F0C with amount 0x000000000100 would be "9F0C06000000000100" If tags 9F02 (amount), 9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method.
<i>forceOnline</i>	TRUE = do not allow offline approval, FALSE = allow ICC to approve offline if terminal capable Note: To request tags to be included in default response, use tag DFEE1A, and specify tag list. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369F37

##### Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

#### 15.2.2.51 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.icc\_getICCReaderStatus ( ref byte *status* )

Get Reader Status

Returns the reader status

##### Parameters

<i>status</i>	Pointer that will return with the ICCReaderStatus results. bit 0: 0 = ICC Power Not Ready, 1 = ICC Powered bit 1: 0 = Card not seated, 1 = card seated
---------------	--

##### Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

#### 15.2.2.52 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.icc\_getICCStatus ( ref byte[] *status*, ref byte[] *atr* )

Get Reader Status

Returns the ICC ATR Data and the status of the Smart Card interface: Smart Card voltage, protocols and parameters



## Parameters

<i>status</i>	<p>Pointer that will return with the ICC info. byte 0: Card Presence</p> <ul style="list-style-type: none"> <li>• Bit 0: Card Inserted</li> <li>• Bit 1: Error communicating with ICC</li> <li>• Bit 2: Error activating/resetting ICC</li> <li>• Bit 5: Latch active</li> <li>• Bit 3,4,6,7: RFU 0 byte 1: Current setting and voltage of ICC <ul style="list-style-type: none"> <li>– 0 = Card is not powered up</li> <li>– 1 = Card is powered with 1.8 V</li> <li>– 2 = Card is powered with 3 V</li> <li>– 3 = Card is powered with 5 V byte 2: Protocol</li> <li>– 0 = (T = 0)</li> <li>– 1 = (T = 1) bytes 3-5: T=0 Protocol parameters (if T=1, bytes 3-5 all 00)</li> <li>– Byte 3 = TA1 (FI/DI)</li> <li>– Byte 4 = TC1 (Guard Time)</li> <li>– Byte 5 = WI bytes 6-9: T=1 Protocol parameters (if T=0, bytes 6-9 all 00)</li> <li>– Byte 6 = TA1 (FI/DI)</li> <li>– Byte 7 = TC1 (EGT)</li> <li>– Byte 8 = TB3 (BWI/CWI)</li> <li>– Byte 9 = IFSC</li> </ul> </li> </ul>
<i>atr</i>	ATR Data is included when card is inserted and no other errors returned.

## Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

### 15.2.2.53 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.icc\_powerOffICC ( )

Power Off ICC

Powers down the ICC

## Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

If Success, empty If Failure, ASCII encoded data of error string

### 15.2.2.54 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.icc\_powerOnICC ( ref byte[] ATR )

Power On ICC

Power up the currently selected microprocessor card in the ICC reader

## Returns

RETURN\_CODE: Values can be parsed with device\_getResponseCodeString

#### 15.2.2.55 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.msr\_cancelMSRSwipe ( )

Disable MSR Swipe Cancels MSR swipe request.

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.56 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.msr\_clearMSRData ( )

Clear MSR Data

Clears the MSR Data buffer

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.57 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.msr\_getMSRData ( ref IDTTransactionData card )

Get MSR Data

Reads the MSR Data buffer

@param card Card data

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.58 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.msr\_startMSRSwipe ( int timeout )

Enable MSR Swipe

Enables MSR, waiting for swipe to occur. Allows track selection. Returns IDTMSRData instance to `deviceDelegate::swipeMSRData:()`

##### Parameters

<i>timeout</i>	Swipe Timeout Value
----------------	---------------------

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

#### 15.2.2.59 RETURN\_CODE IDTechSDK.IDT\_SpectrumPro.pin\_cancelPINEntry ( )

Cancel PIN Entry

Cancels PIN entry request

##### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.60 **RETURN\_CODE** IDTechSDK.IDT\_SpectrumPro.pin\_getPIN ( int *mode*, int *PANSource*, string *iccPAN*, int *startTimeout*, int *entryTimeout*, string *language* )

Get Encrypted PIN

Requests PIN Entry

#### Parameters

<i>mode</i>	<ul style="list-style-type: none"> <li>• 0x00- Cancel: Cancels PIN entry = also can execute <a href="#">pin_cancelPINEntry()</a>. All other parameters for this method will be ignored</li> <li>• 0x01- Online PIN DUKPT</li> <li>• 0x02- Online PIN MKSK</li> <li>• 0x03- Offline PIN (No need to define PAN Source or ICC PAN)</li> </ul>
<i>PANSource</i>	<ul style="list-style-type: none"> <li>• 0x00- ICC: PAN Captured from ICC and must be provided in iccPAN parameter</li> <li>• 0x01- MSR: PAN Captured from MSR swipe and will be inserted by Spectrum Pro. No need to provide iccPAN parameter.</li> </ul>
<i>iccPAN</i>	PAN captured from ICC. When PAN captured from MSR, this parameter will be ignored
<i>startTimeout</i>	The amount of time allowed to start PIN entry before timeout
<i>entryTimeout</i>	The amount of time to enter the PIN after first digit selected before timeout
<i>language</i>	Valid values "EN" for English, "ES" for Spanish, "ZH" for Chinese, "FR" for French

#### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.61 **RETURN\_CODE** IDTechSDK.IDT\_SpectrumPro.pin\_passThroughMode ( bool *enable* )

Pin Pass Through

Enables all commands to be passed through the PINpad

#### Parameters

<i>enable</i>	TRUE = all commands passed through, FALSE = commands not passed through
---------------	---

#### Returns

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

15.2.2.62 **static String** IDTechSDK.IDT\_SpectrumPro.SDK\_Version ( ) [static]

SDK Version

- All Devices

Returns the current version of SDK

**Returns**

Framework version

**15.2.2.63** `static void IDTechSDK.IDT_SpectrumPro.setCallback ( CallBack my_Callback )` `[static]`

**Set Callback**

Sets the class callback

**15.2.2.64** `static void IDTechSDK.IDT_SpectrumPro.setCallback ( IntPtr my_Callback, SynchronizationContext context )`  
`[static]`

**Set Callback**

Sets the class callback

**Parameters**

<i>my_Callback</i>	The callback function to receive the response message from device. defined as follows. public unsafe delegate void MFCCallBack(Parameters parameters);
<i>context</i>	The context of the UI thread

**15.2.2.65** `static bool IDTechSDK.IDT_SpectrumPro.useSerialPort ( int port, bool isSRED )` `[static]`

**Use Serial Port Interface**

Instructs SDK to attempt to use the Serial Port for communication with Spectrum Pro using default baud rate

**Parameters**

<i>port</i>	Serial Port to use. Example COM1 = 1.
<i>isSRED</i>	IF Secure/SRED, pass TRUE. Otherwise, pass FALSE

**Returns**

bool TRUE=successful, FALSE=failure

**15.2.2.66** `static bool IDTechSDK.IDT_SpectrumPro.useSerialPort ( int port, int baud, bool isSRED )` `[static]`

**Use Serial Port Interface with baud rate**

Instructs SDK to attempt to use the Serial Port for communication with Spectrum Pro

**Parameters**

<i>port</i>	Serial Port to use. Example COM1 = 1.
<i>baud</i>	Baud rate to override default. Example 115200;
<i>isSRED</i>	IF Secure/SRED, pass TRUE. Otherwise, pass FALSE

**Returns**

bool TRUE=successful, FALSE=failure

15.2.2.67 static bool IDTechSDK.IDT\_SpectrumPro.useUSB ( ) [static]

Use USB Interface

Instructs SDK to attempt to use USB for communication with Spectrum Pro

**15.2.3 Property Documentation**

15.2.3.1 IDT\_SpectrumPro IDTechSDK.IDT\_SpectrumPro.SharedController [static],[get]

Singleton Instance

Establishes an singleton instance of Spectrum Pro class.

**Returns**

Instance of Spectrum Pro

The documentation for this class was generated from the following file:

- Source/IDT\_SpectrumPro.cs

**15.3 IDTechSDK.IDTTransactionData Class Reference****Static Public Member Functions**

- static byte[] [decryptData](#) (bool isAES, byte[] ksn, string BDK, byte[] encodedData)
- static byte[] [decryptData](#) (bool isAES, byte[] ksn, string BDK1, string BDK2, byte[] encodedData)
- static void **setTransactionAttributes** (byte attribute, ref [IDTTransactionData](#) data)

**Public Attributes**

- EVENT\_TRANSACTION\_DATA\_Types [Event](#)
- EVENT\_NOTIFICATION\_Types [Notification](#)
- byte[] [msr\\_rawData](#)
- byte[] [msr\\_encTrack1](#)
- byte[] [msr\\_encTrack2](#)
- byte[] [msr\\_encTrack3](#)
- String [msr\\_track1](#)
- String [msr\\_track2](#)
- String [msr\\_track3](#)
- String [device\\_RSN](#)
- byte[] [msr\\_KSN](#)
- int [msr\\_track1Length](#)
- int [msr\\_track2Length](#)
- int [msr\\_track3Length](#)
- CAPTURE\_ENCODE\_TYPE [msr\\_cardType](#)
- byte [msr\\_captureEncodeStatus](#)
- CAPTURE\_ENCRYPT\_TYPE [captureEncryptType](#)
- CAPTURE\_ENCRYPT\_TYPE [captureEncryptTypeEMV](#)

- CAPTURE\_CARD\_TYPE [captureCardType](#)
- int [msr\\_errorCode](#)
- int [emv\\_rfStateCode](#)
- int [iccPresent](#)
- byte[] [msr\\_sessionID](#)
- byte[] [msr\\_hashTrack1](#)
- byte[] [msr\\_hashTrack2](#)
- byte[] [msr\\_hashTrack3](#)
- KEY\_VARIANT\_TYPE [msr\\_keyVariantType](#)
- byte[] [msr\\_extendedField](#)
- int [isCTLS](#)
- CTLS\_APPLICATION [ctlsApplication](#)
- byte[] [emv\\_clearingRecord](#)
- byte[] [emv\\_encryptedTags](#)
- byte[] [emv\\_unencryptedTags](#)
- EMV\_RESULT\_CODE [emv\\_resultCode](#)
- byte[] [emv\\_maskedTags](#)
- bool [emv\\_hasAdvise](#)
- bool [emv\\_hasReversal](#)
- string [pin\\_pinblock](#)
- string [pin\\_KSN](#)
- string [pin\\_KeyEntry](#)
- byte [SW1](#)
- byte [SW2](#)
- byte[] [mac](#)
- byte[] [macKSN](#)
- TRANS\_ERROR\_CODE [emv\\_transaction\\_Error\\_Code](#)
- RF\_STATE [emv\\_RF\\_State](#)
- EXTENDED\_STATUS\_CODES [emv\\_ESC](#)
- CEMV\_APP\_ERROR\_FN [emv\\_appErrorFn](#)
- CEMV\_APP\_ERROR\_STATE [emv\\_appErrorState](#)

### 15.3.1 Detailed Description

Class for swipe data

### 15.3.2 Member Function Documentation

15.3.2.1 `static byte [] IDTechSDK.IDTTransactionData.decryptData ( bool isAES, byte [] ksn, string BDK, byte [] encodedData ) [static]`

Decrypt Data

Decrypted TDES or AES encrypted data if the BDK and KSN are known. Requires DecryptDLL.dll

Parameters

<i>isAES</i>	TRUE = AES, FALSE = TDES
<i>ksn</i>	Key Serial Number
<i>BDK</i>	Base Derivative Key

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

**15.3.2.2** `static byte [] IDTechSDK.IDTTransactionData.decryptData ( bool isAES, byte [] ksn, string BDK1, string BDK2, byte [] encodedData ) [static]`

**Decrypt Data**

Decrypted TDES or AES encrypted data if two initial BDK's and KSN are known. Requires DecryptDLL.dll

**Parameters**

<i>isAES</i>	TRUE = AES, FALSE = TDES
<i>ksn</i>	Key Serial Number
<i>BDK1</i>	First Base Derivative Key
<i>BDK2</i>	Second Base Derivative Key

**Returns**

RETURN\_CODE: Values can be parsed with `errorCode.getErrorString()`

**15.3.3 Member Data Documentation****15.3.3.1 CAPTURE\_CARD\_TYPE IDTechSDK.IDTTransactionData.captureCardType**

Get the captured card type, please see CAPTURE\_CARD\_TYPE for more information.

CAPTURE\_CARD\_TYPE\_UNKNOWN;  
CAPTURE\_CARD\_TYPE\_CONTACT;  
CAPTURE\_CARD\_TYPE\_CTLS\_EMV;  
CAPTURE\_CARD\_TYPE\_CTLS\_MSD;  
CAPTURE\_CARD\_TYPE\_MSR;

**15.3.3.2 CAPTURE\_ENCRYPT\_TYPE IDTechSDK.IDTTransactionData.captureEncryptType**

Get the encrypted type, please see CAPTURE\_ENCRYPT\_TYPE for more information.

CAPTURE\_ENCRYPT\_TYPE\_TDES:TDES;  
CAPTURE\_ENCRYPT\_TYPE\_AES:AES;

**15.3.3.3 CAPTURE\_ENCRYPT\_TYPE IDTechSDK.IDTTransactionData.captureEncryptTypeEMV**

Get the encrypted type for EMV, please see CAPTURE\_ENCRYPT\_TYPE for more information.

CAPTURE\_ENCRYPT\_TYPE\_NONE:NONE;  
CAPTURE\_ENCRYPT\_TYPE\_TDES:TDES;  
CAPTURE\_ENCRYPT\_TYPE\_AES:AES;

**15.3.3.4 CTLS\_APPLICATION IDTechSDK.IDTTransactionData.ctlsApplication****CTLS Application**

#### 15.3.3.5 String IDTechSDK.IDTTransactionData.device\_RSN

Get the Reader Serial Number.

#### 15.3.3.6 CEMV\_APP\_ERROR\_FN IDTechSDK.IDTTransactionData.emv\_appErrorFn

EMV App Error Function (select AR products)

#### 15.3.3.7 CEMV\_APP\_ERROR\_STATE IDTechSDK.IDTTransactionData.emv\_appErrorState

EMV App Error State (select AR products)

#### 15.3.3.8 byte [] IDTechSDK.IDTTransactionData.emv\_clearingRecord

clearing record TLV

#### 15.3.3.9 byte [] IDTechSDK.IDTTransactionData.emv\_encryptedTags

Encrypted Tags TLV

#### 15.3.3.10 EXTENDED\_STATUS\_CODES IDTechSDK.IDTTransactionData.emv\_ESC

Extended Status Code (select AR products)

#### 15.3.3.11 bool IDTechSDK.IDTTransactionData.emv\_hasAdvise

Advise

#### 15.3.3.12 bool IDTechSDK.IDTTransactionData.emv\_hasReversal

Reversal

#### 15.3.3.13 byte [] IDTechSDK.IDTTransactionData.emv\_maskedTags

Masked Tags TLV

#### 15.3.3.14 EMV\_RESULT\_CODE IDTechSDK.IDTTransactionData.emv\_resultCode

EMV Result Code

#### 15.3.3.15 RF\_STATE IDTechSDK.IDTTransactionData.emv\_RF\_State

RF\_State (select AR products)

#### 15.3.3.16 int IDTechSDK.IDTTransactionData.emv\_rfStateCode

For some Error Codes, the RF State Code indicates the exact Reader-Card command that failed. This helps determine the exact place where the failure occurred.



**15.3.3.17 TRANS\_ERROR\_CODE IDTechSDK.IDTTransactionData.emv\_transaction\_Error\_Code**

Transaction Error Code (select AR products)

**15.3.3.18 byte [] IDTechSDK.IDTTransactionData.emv\_unencryptedTags**

Unencrypted Tags TLV

**15.3.3.19 EVENT\_TRANSACTION\_DATA\_Types IDTechSDK.IDTTransactionData.Event**

Transaction Data type, please see EVENT\_TRANSACTION\_DATA\_Types for more information.

**15.3.3.20 int IDTechSDK.IDTTransactionData.iccPresent**

Get the swiped card ICC Status.

0 = Unknown 1 = True 2 = False

**15.3.3.21 int IDTechSDK.IDTTransactionData.isCTLS**

Track data was captured via CTLS interface 0 = Unknown 1 = True 2 = False

**15.3.3.22 byte [] IDTechSDK.IDTTransactionData.mac**

Message Authentication Code

**15.3.3.23 byte [] IDTechSDK.IDTTransactionData.macKSN**

Message Authentication Code Key Serial Number

**15.3.3.24 byte IDTechSDK.IDTTransactionData.msr\_captureEncodeStatus**

Get the swiped card decoded status.

0x00:decoded data success;

Bit0:1-track1 data error;

Bit1:1-track2 data error;

Bit2:1-track3 data error;

Bit3:1-track1 encrypted data error;

Bit4:1-track2 encrypted data error;

Bit5:1-track3 encrypted data error;

Bit6:1-KSN error;

**15.3.3.25 CAPTURE\_ENCODE\_TYPE IDTechSDK.IDTTransactionData.msr\_cardType**

Get the swiped card type, please see CAPTURE\_ENCODE\_TYPE for more information.

MSR card type:

CAPTURE\_ENCODE\_TYPE\_ISOABA:ISO/ABA format

CAPTURE\_ENCODE\_TYPE\_AAMVA:AAMVA format

CAPTURE\_ENCODE\_TYPE\_Other:Other

CAPTURE\_ENCODE\_TYPE\_Raw:Raw; undecoded format

CAPTURE\_ENCODE\_TYPE\_JisI\_II:JIS I or JIS II

**15.3.3.26    byte [] IDTechSDK.IDTTransactionData.msr\_encTrack1**

Get the swiped card Track1 encrypted data.  
A byte array containing Track1 encrypted data.

**15.3.3.27    byte [] IDTechSDK.IDTTransactionData.msr\_encTrack2**

Get the swiped card Track2 encrypted data.  
A byte array containing Track2 encrypted data.

**15.3.3.28    byte [] IDTechSDK.IDTTransactionData.msr\_encTrack3**

Get the swiped card Track3 encrypted data.  
A byte array containing Track3 encrypted data.

**15.3.3.29    int IDTechSDK.IDTTransactionData.msr\_errorCode**

Contains error code when data is not returned

**15.3.3.30    byte [] IDTechSDK.IDTTransactionData.msr\_extendedField**

Extended Field Data. Byte 0: 1 = Hash-SHA256

**15.3.3.31    byte [] IDTechSDK.IDTTransactionData.msr\_hashTrack1**

Get the swiped card Track1 hash data.  
A byte array containing Track1 hash data.

**15.3.3.32    byte [] IDTechSDK.IDTTransactionData.msr\_hashTrack2**

Get the swiped card Track2 hash data.  
A byte array containing Track2 hash data.

**15.3.3.33    byte [] IDTechSDK.IDTTransactionData.msr\_hashTrack3**

Get the swiped card Track3 hash data.  
A byte array containing Track3 hash data.

**15.3.3.34    KEY\_VARIANT\_TYPE IDTechSDK.IDTTransactionData.msr\_keyVariantType**

KEY\_VARIANT\_TYPE\_DATA = Data Variant key used  
KEY\_VARIANT\_TYPE\_PIN = PIN Variant key used

**15.3.3.35    byte [] IDTechSDK.IDTTransactionData.msr\_KSN**

Get the swiped card KSN (Key Serial Number).  
A byte array containing 10 bytes.

**15.3.3.36    byte [] IDTechSDK.IDTTransactionData.msr\_rawData**

Get the card data raw data.  
Containing complete unparsed transaction data as received from device.

**15.3.3.37    byte [] IDTechSDK.IDTTransactionData.msr\_sessionID**

Get the swiped card Session ID.  
A byte array to get session ID, if exists.

**15.3.3.38    String IDTechSDK.IDTTransactionData.msr\_track1**

Get the swiped card Track1 data.  
A string containing Track1 masked data expressed as hex characters.

**15.3.3.39    int IDTechSDK.IDTTransactionData.msr\_track1Length**

Get the swiped card length of Track1 data.

**15.3.3.40    String IDTechSDK.IDTTransactionData.msr\_track2**

Get the swiped card Track2 data.  
A string containing Track2 masked data expressed as hex characters.

**15.3.3.41    int IDTechSDK.IDTTransactionData.msr\_track2Length**

Get the swiped card length of Track2 data.

**15.3.3.42    String IDTechSDK.IDTTransactionData.msr\_track3**

Get the swiped card Track3 data.  
A string containing Track3 masked data expressed as hex characters.

**15.3.3.43    int IDTechSDK.IDTTransactionData.msr\_track3Length**

Get the swiped card length of Track3 data.

**15.3.3.44    EVENT\_NOTIFICATION\_Types IDTechSDK.IDTTransactionData.Notification**

Event Notification type, please see EVENT\_NOTIFICATION\_Types for more information.

**15.3.3.45    string IDTechSDK.IDTTransactionData.pin\_KeyEntry**

KSN for Pinblock

**15.3.3.46    string IDTechSDK.IDTTransactionData.pin\_KSN**

KSN for Pinblock

15.3.3.47 `string IDTechSDK.IDTTransactionData.pin_pinblock`

PIN block from PINPAD

15.3.3.48 `byte IDTechSDK.IDTTransactionData.SW1`

SW1

15.3.3.49 `byte IDTechSDK.IDTTransactionData.SW2`

SW2

The documentation for this class was generated from the following file:

- `Source/IDT_Transactions.cs`

# Index

CTLS\_APPLICATION  
    IDTechSDK, [64](#)  
callbackType  
    IDTechSDK::EMV\_Callback, [66](#)  
captureCardType  
    IDTechSDK::IDTTransactionData, [103](#)  
captureEncryptType  
    IDTechSDK::IDTTransactionData, [103](#)  
captureEncryptTypeEMV  
    IDTechSDK::IDTTransactionData, [103](#)  
config\_getModelNumber  
    IDTechSDK::IDT\_SpectrumPro, [70](#)  
config\_getSerialNumber  
    IDTechSDK::IDT\_SpectrumPro, [70](#)  
ctlsApplication  
    IDTechSDK::IDTTransactionData, [103](#)  
  
decryptData  
    IDTechSDK::IDTTransactionData, [102](#), [103](#)  
device\_RSN  
    IDTechSDK::IDTTransactionData, [103](#)  
device\_controlUserInterface  
    IDTechSDK::IDT\_SpectrumPro, [70](#)  
device\_getFirmwareVersion  
    IDTechSDK::IDT\_SpectrumPro, [71](#)  
device\_getResponseCodeString  
    IDTechSDK::IDT\_SpectrumPro, [71](#)  
device\_getSpectrumProKSN  
    IDTechSDK::IDT\_SpectrumPro, [81](#)  
device\_pollCardReader  
    IDTechSDK::IDT\_SpectrumPro, [82](#)  
device\_rebootDevice  
    IDTechSDK::IDT\_SpectrumPro, [83](#)  
device\_sendDataCommand  
    IDTechSDK::IDT\_SpectrumPro, [83](#)  
device\_sendMacDataCommand  
    IDTechSDK::IDT\_SpectrumPro, [83](#)  
device\_setSpectrumProBDK  
    IDTechSDK::IDT\_SpectrumPro, [84](#)  
device\_setUID  
    IDTechSDK::IDT\_SpectrumPro, [84](#)  
device\_startRKI  
    IDTechSDK::IDT\_SpectrumPro, [84](#)  
device\_updateFirmware  
    IDTechSDK::IDT\_SpectrumPro, [85](#)  
  
EMV\_CALLBACK\_TYPE  
    IDTechSDK, [64](#)  
EMV\_LCD\_DISPLAY\_MODE  
    IDTechSDK, [64](#)  
  
EMV\_PIN\_MODE  
    IDTechSDK, [65](#)  
EMV\_RESULT\_CODE  
    IDTechSDK, [65](#)  
emv\_ESC  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_RF\_State  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_allowFallback  
    IDTechSDK::IDT\_SpectrumPro, [85](#)  
emv\_appErrorFn  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_appErrorState  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_authenticateTransaction  
    IDTechSDK::IDT\_SpectrumPro, [85](#)  
emv\_autoAuthenticate  
    IDTechSDK::IDT\_SpectrumPro, [86](#)  
emv\_callbackResponseLCD  
    IDTechSDK::IDT\_SpectrumPro, [86](#)  
emv\_callbackResponseMSR  
    IDTechSDK::IDT\_SpectrumPro, [87](#)  
emv\_callbackResponsePIN  
    IDTechSDK::IDT\_SpectrumPro, [87](#)  
emv\_cancelTransaction  
    IDTechSDK::IDT\_SpectrumPro, [87](#)  
emv\_clearTransactionLog  
    IDTechSDK::IDT\_SpectrumPro, [88](#)  
emv\_clearingRecord  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_completeTransaction  
    IDTechSDK::IDT\_SpectrumPro, [88](#)  
emv\_encryptedTags  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_getEMVConfigurationCheckValue  
    IDTechSDK::IDT\_SpectrumPro, [88](#)  
emv\_getEMVKernelCheckValue  
    IDTechSDK::IDT\_SpectrumPro, [89](#)  
emv\_getEMVKernelVersion  
    IDTechSDK::IDT\_SpectrumPro, [89](#)  
emv\_getTerminalID  
    IDTechSDK::IDT\_SpectrumPro, [89](#)  
emv\_getTerminalMajorConfiguration  
    IDTechSDK::IDT\_SpectrumPro, [89](#)  
emv\_hasAdvise  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_hasReversal  
    IDTechSDK::IDTTransactionData, [104](#)  
emv\_maskedTags

- IDTechSDK::IDTTransactionData, 104
- emv\_removeAllApplicationData
  - IDTechSDK::IDT\_SpectrumPro, 90
- emv\_removeAllCAPK
  - IDTechSDK::IDT\_SpectrumPro, 90
- emv\_removeAllCRL
  - IDTechSDK::IDT\_SpectrumPro, 90
- emv\_removeApplicationData
  - IDTechSDK::IDT\_SpectrumPro, 90
- emv\_removeCAPK
  - IDTechSDK::IDT\_SpectrumPro, 91
- emv\_removeCRL
  - IDTechSDK::IDT\_SpectrumPro, 91
- emv\_removeTerminalData
  - IDTechSDK::IDT\_SpectrumPro, 91
- emv\_resultCode
  - IDTechSDK::IDTTransactionData, 104
- emv\_retrieveAIDList
  - IDTechSDK::IDT\_SpectrumPro, 91
- emv\_retrieveApplicationData
  - IDTechSDK::IDT\_SpectrumPro, 92
- emv\_retrieveCAPKList
  - IDTechSDK::IDT\_SpectrumPro, 92
- emv\_retrieveCAPK
  - IDTechSDK::IDT\_SpectrumPro, 92
- emv\_retrieveCRLList
  - IDTechSDK::IDT\_SpectrumPro, 93
- emv\_retrieveTerminalData
  - IDTechSDK::IDT\_SpectrumPro, 93
- emv\_retrieveTransactionResult
  - IDTechSDK::IDT\_SpectrumPro, 93
- emv\_rfStateCode
  - IDTechSDK::IDTTransactionData, 104
- emv\_setApplicationData
  - IDTechSDK::IDT\_SpectrumPro, 93
- emv\_setCAPK
  - IDTechSDK::IDT\_SpectrumPro, 94
- emv\_setCRL
  - IDTechSDK::IDT\_SpectrumPro, 94
- emv\_setTerminalData
  - IDTechSDK::IDT\_SpectrumPro, 95
- emv\_setTerminalID
  - IDTechSDK::IDT\_SpectrumPro, 95
- emv\_setTerminalMajorConfiguration
  - IDTechSDK::IDT\_SpectrumPro, 95
- emv\_startTransaction
  - IDTechSDK::IDT\_SpectrumPro, 95
- emv\_transaction\_Error\_Code
  - IDTechSDK::IDTTransactionData, 104
- emv\_unencryptedTags
  - IDTechSDK::IDTTransactionData, 105
- Event
  - IDTechSDK::IDTTransactionData, 105
- IDTechSDK.EMV\_Callback, 66
- IDTechSDK.IDT\_SpectrumPro, 68
- IDTechSDK.IDTTransactionData, 101
- IDTechSDK::EMV\_Callback
  - callbackType, 66
  - language, 66
  - lcd\_backlightTimeout, 67
  - lcd\_displayMode, 67
  - lcd\_entryTimeout, 67
  - lcd\_entryTimeoutMinor, 67
  - lcd\_messages, 67
  - maskEntry, 67
  - msr\_displayMessage, 67
  - msr\_swipeTimeout, 68
  - pin\_KSN, 68
  - pin\_entryInterval, 68
  - pin\_entryStartTimeout, 68
  - pin\_pinMode, 68
  - pin\_truncatedPAN, 68
- IDTechSDK::IDT\_SpectrumPro
  - config\_getModelNumber, 70
  - config\_getSerialNumber, 70
  - device\_controlUserInterface, 70
  - device\_getFirmwareVersion, 71
  - device\_getResponseCodeString, 71
  - device\_getSpectrumProKSN, 81
  - device\_pollCardReader, 82
  - device\_rebootDevice, 83
  - device\_sendDataCommand, 83
  - device\_sendMacDataCommand, 83
  - device\_setSpectrumProBDK, 84
  - device\_setUID, 84
  - device\_startRKI, 84
  - device\_updateFirmware, 85
  - emv\_allowFallback, 85
  - emv\_authenticateTransaction, 85
  - emv\_autoAuthenticate, 86
  - emv\_callbackResponseLCD, 86
  - emv\_callbackResponseMSR, 87
  - emv\_callbackResponsePIN, 87
  - emv\_cancelTransaction, 87
  - emv\_clearTransactionLog, 88
  - emv\_completeTransaction, 88
  - emv\_getEMVConfigurationCheckValue, 88
  - emv\_getEMVKernelCheckValue, 89
  - emv\_getEMVKernelVersion, 89
  - emv\_getTerminalID, 89
  - emv\_getTerminalMajorConfiguration, 89
  - emv\_removeAllApplicationData, 90
  - emv\_removeAllCAPK, 90
  - emv\_removeAllCRL, 90
  - emv\_removeApplicationData, 90
  - emv\_removeCAPK, 91
  - emv\_removeCRL, 91
  - emv\_removeTerminalData, 91
  - emv\_retrieveAIDList, 91
  - emv\_retrieveApplicationData, 92
  - emv\_retrieveCAPKList, 92
  - emv\_retrieveCAPK, 92
  - emv\_retrieveCRLList, 93
  - emv\_retrieveTerminalData, 93
  - emv\_retrieveTransactionResult, 93
  - emv\_setApplicationData, 93

- emv\_setCAPK, 94
- emv\_setCRL, 94
- emv\_setTerminalData, 95
- emv\_setTerminalID, 95
- emv\_setTerminalMajorConfiguration, 95
- emv\_startTransaction, 95
- icc\_getICCRReaderStatus, 96
- icc\_getICCStatus, 96
- icc\_powerOffICC, 97
- icc\_powerOnICC, 97
- msr\_cancelMSRSwipe, 97
- msr\_clearMSRData, 98
- msr\_getMSRData, 98
- msr\_startMSRSwipe, 98
- pin\_cancelPINEntry, 98
- pin\_getPIN, 98
- pin\_passThroughMode, 99
- SDK\_Version, 99
- setCallback, 100
- SharedController, 101
- useSerialPort, 100
- useUSB, 101
- IDTechSDK::IDTTTransactionData
  - captureCardType, 103
  - captureEncryptType, 103
  - captureEncryptTypeEMV, 103
  - ctlsApplication, 103
  - decryptData, 102, 103
  - device\_RSN, 103
  - emv\_ESC, 104
  - emv\_RF\_State, 104
  - emv\_appErrorFn, 104
  - emv\_appErrorState, 104
  - emv\_clearingRecord, 104
  - emv\_encryptedTags, 104
  - emv\_hasAdvise, 104
  - emv\_hasReversal, 104
  - emv\_maskedTags, 104
  - emv\_resultCode, 104
  - emv\_rfStateCode, 104
  - emv\_transaction\_Error\_Code, 104
  - emv\_unencryptedTags, 105
  - Event, 105
  - iccPresent, 105
  - isCTLS, 105
  - mac, 105
  - mackSN, 105
  - msr\_KSN, 106
  - msr\_captureEncodeStatus, 105
  - msr\_cardType, 105
  - msr\_encTrack1, 105
  - msr\_encTrack2, 106
  - msr\_encTrack3, 106
  - msr\_errorCode, 106
  - msr\_extendedField, 106
  - msr\_hashTrack1, 106
  - msr\_hashTrack2, 106
  - msr\_hashTrack3, 106
  - msr\_keyVariantType, 106
  - msr\_rawData, 106
  - msr\_sessionID, 107
  - msr\_track1, 107
  - msr\_track1Length, 107
  - msr\_track2, 107
  - msr\_track2Length, 107
  - msr\_track3, 107
  - msr\_track3Length, 107
  - Notification, 107
  - pin\_KSN, 107
  - pin\_KeyEntry, 107
  - pin\_pinblock, 107
  - SW1, 108
  - SW2, 108
- IDTechSDK, 63
  - CTLS\_APPLICATION, 64
  - EMV\_CALLBACK\_TYPE, 64
  - EMV\_LCD\_DISPLAY\_MODE, 64
  - EMV\_PIN\_MODE, 65
  - EMV\_RESULT\_CODE, 65
- icc\_getICCRReaderStatus
  - IDTechSDK::IDT\_SpectrumPro, 96
- icc\_getICCStatus
  - IDTechSDK::IDT\_SpectrumPro, 96
- icc\_powerOffICC
  - IDTechSDK::IDT\_SpectrumPro, 97
- icc\_powerOnICC
  - IDTechSDK::IDT\_SpectrumPro, 97
- iccPresent
  - IDTechSDK::IDTTTransactionData, 105
- isCTLS
  - IDTechSDK::IDTTTransactionData, 105
- language
  - IDTechSDK::EMV\_Callback, 66
- lcd\_backlightTimeout
  - IDTechSDK::EMV\_Callback, 67
- lcd\_displayMode
  - IDTechSDK::EMV\_Callback, 67
- lcd\_entryTimeout
  - IDTechSDK::EMV\_Callback, 67
- lcd\_entryTimeoutMinor
  - IDTechSDK::EMV\_Callback, 67
- lcd\_messages
  - IDTechSDK::EMV\_Callback, 67
- mac
  - IDTechSDK::IDTTTransactionData, 105
- mackSN
  - IDTechSDK::IDTTTransactionData, 105
- maskEntry
  - IDTechSDK::EMV\_Callback, 67
- msr\_KSN
  - IDTechSDK::IDTTTransactionData, 106
- msr\_cancelMSRSwipe
  - IDTechSDK::IDT\_SpectrumPro, 97
- msr\_captureEncodeStatus
  - IDTechSDK::IDTTTransactionData, 105

- msr\_cardType
  - IDTechSDK::IDTTransactionData, 105
- msr\_clearMSRData
  - IDTechSDK::IDT\_SpectrumPro, 98
- msr\_displayMessage
  - IDTechSDK::EMV\_Callback, 67
- msr\_encTrack1
  - IDTechSDK::IDTTransactionData, 105
- msr\_encTrack2
  - IDTechSDK::IDTTransactionData, 106
- msr\_encTrack3
  - IDTechSDK::IDTTransactionData, 106
- msr\_errorCode
  - IDTechSDK::IDTTransactionData, 106
- msr\_extendedField
  - IDTechSDK::IDTTransactionData, 106
- msr\_getMSRData
  - IDTechSDK::IDT\_SpectrumPro, 98
- msr\_hashTrack1
  - IDTechSDK::IDTTransactionData, 106
- msr\_hashTrack2
  - IDTechSDK::IDTTransactionData, 106
- msr\_hashTrack3
  - IDTechSDK::IDTTransactionData, 106
- msr\_keyVariantType
  - IDTechSDK::IDTTransactionData, 106
- msr\_rawData
  - IDTechSDK::IDTTransactionData, 106
- msr\_sessionID
  - IDTechSDK::IDTTransactionData, 107
- msr\_startMSRSwipe
  - IDTechSDK::IDT\_SpectrumPro, 98
- msr\_swipeTimeout
  - IDTechSDK::EMV\_Callback, 68
- msr\_track1
  - IDTechSDK::IDTTransactionData, 107
- msr\_track1Length
  - IDTechSDK::IDTTransactionData, 107
- msr\_track2
  - IDTechSDK::IDTTransactionData, 107
- msr\_track2Length
  - IDTechSDK::IDTTransactionData, 107
- msr\_track3
  - IDTechSDK::IDTTransactionData, 107
- msr\_track3Length
  - IDTechSDK::IDTTransactionData, 107
- Notification
  - IDTechSDK::IDTTransactionData, 107
- pin\_KSN
  - IDTechSDK::EMV\_Callback, 68
  - IDTechSDK::IDTTransactionData, 107
- pin\_KeyEntry
  - IDTechSDK::IDTTransactionData, 107
- pin\_cancelPINEntry
  - IDTechSDK::IDT\_SpectrumPro, 98
- pin\_entryInterval
  - IDTechSDK::EMV\_Callback, 68
- pin\_entryStartTimeout
  - IDTechSDK::EMV\_Callback, 68
- pin\_getPIN
  - IDTechSDK::IDT\_SpectrumPro, 98
- pin\_passThroughMode
  - IDTechSDK::IDT\_SpectrumPro, 99
- pin\_pinMode
  - IDTechSDK::EMV\_Callback, 68
- pin\_pinblock
  - IDTechSDK::IDTTransactionData, 107
- pin\_truncatedPAN
  - IDTechSDK::EMV\_Callback, 68
- SDK\_Version
  - IDTechSDK::IDT\_SpectrumPro, 99
- SW1
  - IDTechSDK::IDTTransactionData, 108
- SW2
  - IDTechSDK::IDTTransactionData, 108
- setCallback
  - IDTechSDK::IDT\_SpectrumPro, 100
- SharedController
  - IDTechSDK::IDT\_SpectrumPro, 101
- useSerialPort
  - IDTechSDK::IDT\_SpectrumPro, 100
- useUSB
  - IDTechSDK::IDT\_SpectrumPro, 101