Value through Innovation

IDTech Windows SDK Guide

for SmartPIN L100

#80141502-001

Rev. A

Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;L100;SDK;Win	3/12/2016
A	Updated UWP Implementation Information	9/21/2016

Contents

Chapter 1

IDTech Windows SDK Reference Guide for SmartPIN L100

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 SmartPIN L100 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.

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 MasterCards 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 merchants 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 [1]	Yes	Yes [1]	No
	Service Code [1]	Yes	Yes [1]	No
	Expiration Date [1]	Yes	Yes [1]	No
Sensitive Authentication Data [2]	Full Magnetic Stripe Data [3]	No	N/A	N/A
	CAV2/CID/CVC2/CVV2	No	N/A	N/A
	PIN/PIN Block	No	N/A	N/A

[1] 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.

[2] Do not store sensitive authentication data after authorization (even if encrypted).

[3] 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 P↩ A-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 datafor 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, IE↩ EE 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 S↩ NMP 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

Chapter 3

Sending Direct Commands

The main purpose of IDTech.framework for SmartPIN L100 is to expedite integration to the device by providing the connectivity and communication protocols. It also provides the main functions to display message, collect PIN entry and collect other pinpad entry.

The SmartPIN L100 may have other functionality not exposed by a method in this SDK. For those cases, there is a method to send any command. IDTech will provide the necessary command information to send using the following method:

IDTechSDK::IDT_L100::device_sendDataCommand()

Any function not supported by the SDK can be sent with the device_sendDataCommand.

Chapter 4

Connecting to SmartPIN L100

The SmartPIN L100 connects through either USB or Serial Interface (COM).

4.1 Connect with USB:

The SmartPIN L100 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 SmartPIN L200, you would execute the following command during program initialization to establish a connection:

```
IDT_L100.useUSB();
```

4.2 Connect with Serial Interface (COM)

The IDT_L100 can connect via Serial Interface. The default serial port settings are as follows:

- Speed: 38,400
- Bits: 8
- Stop Bit: 1
- Parity: None

To inform the SDK you are using the Serial Interface of the SmartPIN L100, 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_L100.useSerialPort(1, false);
```

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

```
IDT_L100.useSerialPort(1, 115200, false);
```

Chapter 5

Core Implementation: WinForms

IDTechSDK.dll includes class libraries to interface with the SmartPIN L100. 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 Kiosk III
- 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.

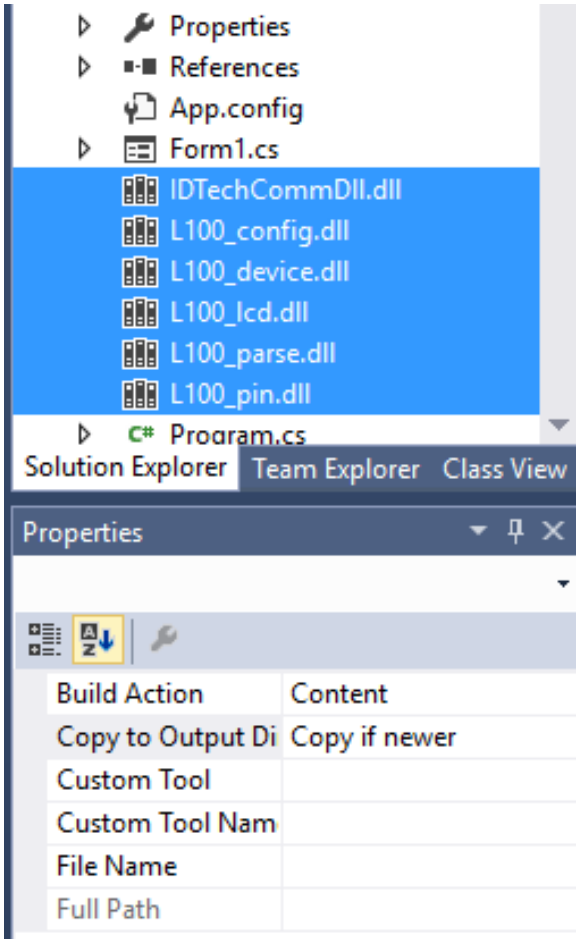
In addition, the following libraries need to be added to project folder and included in with the application distribution:

- IDTechCommDll.dll
- ssleay32.dll
- libeay32.dll
- L100_config.dll
- L100_device.dll
- L100_lcd.dll
- L100_parse.dll

- L100_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 occured 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 beign 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:
            //Device timeout
            break;
        case DeviceState.TransactionFailed:
            //Transaction failed to complete
            break;
        case DeviceState.EMVCallback:
            //Callback during EMV transaction retrieved from EMV_Callback emvCallback
             break;

       case DeviceState.PINpadKeypress:
            //Callback during PINPAD keypress
            break;
        default:
            break;

    }

}
```

5.5 Initialize SmartPIN L100:

A Singleton instance has been established in the IDT_L100 class. Establish the callback, and then set a connection to KioskIII through either USB or Serial.

```
public L100_Simple_Demo()
{
    InitializeComponent();
    IDT_L100.setCallback(MessageCallBack);
    cbBaud.SelectedIndex = 0;
}
```

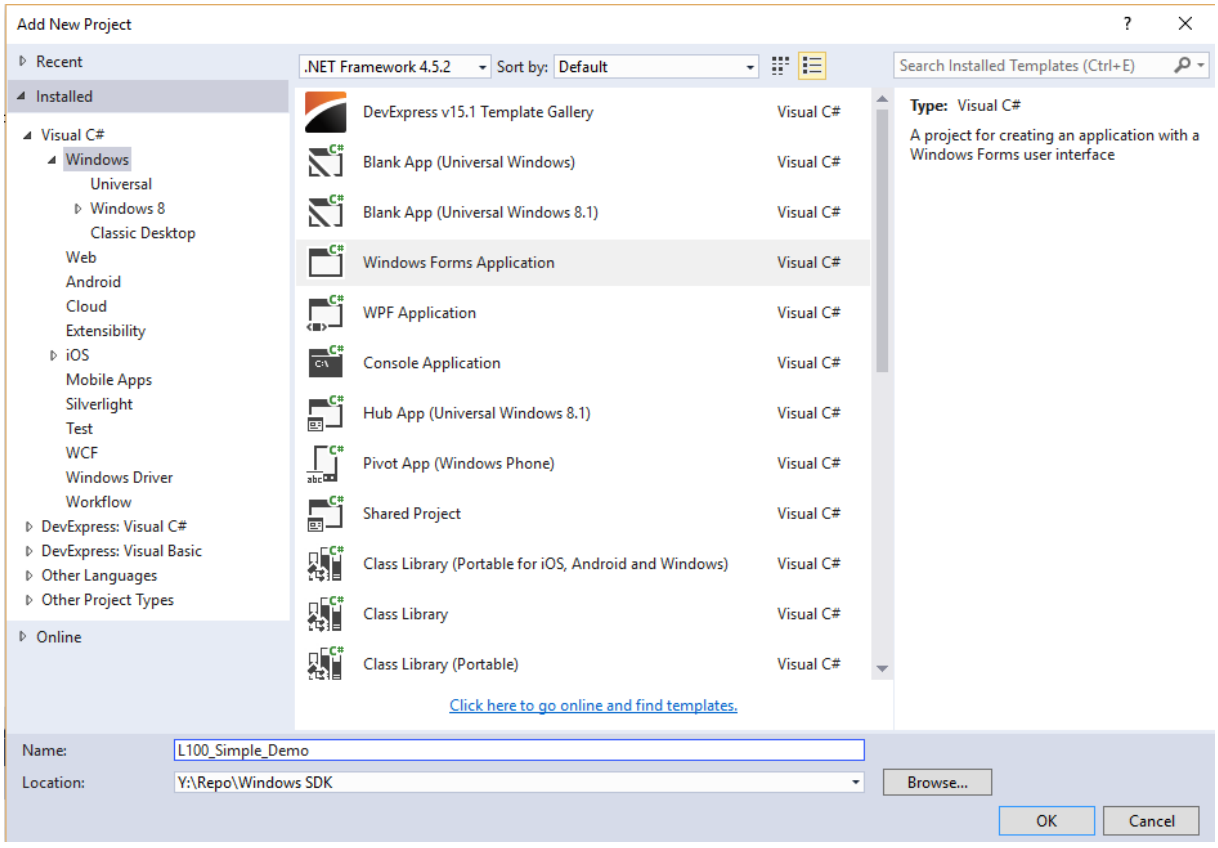
5.6 Sample Project Tutorial

Using Visual Studio 2015, we will create a C# sample project that will interface with the SmartPin L100 and will perform the following activities:

- Get firmware version
- Prompt for PIN Entry and return data
- Display a secure message and prompt for entry
- Display a non-secure message
- Connect to either USB or Serial Versions of L100.
- Show log of all data going to/from L100

5.6.1 Step 1: Create New Project

Create a new Windows Form Application in Visual Studio. In our example, we use project name L100_Simple_Demo



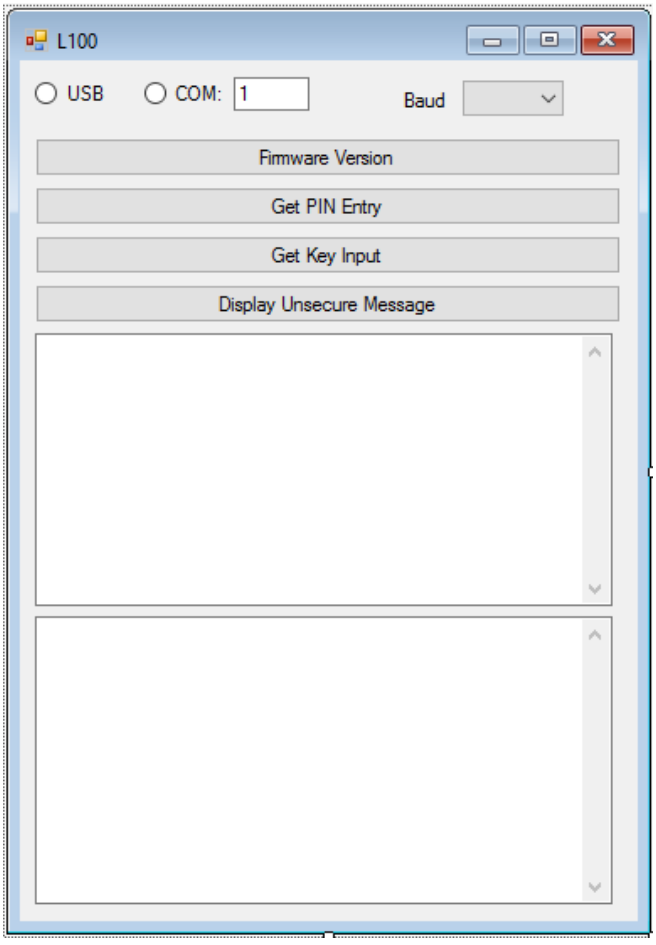
5.6.2 Step 2: Import Libraries

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 a drop-down for baud rate
- Add buttons to execute the following functions:
  - Get Firmware
  - Get PIN Entry
  - Secure Message Prompt
  - Unsecure Message Prompt
- Add a text box to communicate data from the L100.
- Add a text box for the log of L100.




```
namespace L100_Simple_Demo
{
    partial class Form1
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        private void InitializeComponent()
        {
            this.logOutput = new System.Windows.Forms.TextBox();
            this.btnSecureMessage = new System.Windows.Forms.Button();
```

```
        this.btnGetPIN = new System.Windows.Forms.Button();
        this.btnFirmwareVersion = new System.Windows.Forms.Button();
        this.tbOutput = new System.Windows.Forms.TextBox();
        this.rbCOMM = new System.Windows.Forms.RadioButton();
        this.rbUSB = new System.Windows.Forms.RadioButton();
        this.tbCOMPort = new System.Windows.Forms.TextBox();
        this.cbBaud = new System.Windows.Forms.ComboBox();
        this.label1 = new System.Windows.Forms.Label();
        this.btnUnsecureMessage = new System.Windows.Forms.Button();
        this.SuspendLayout();
        //
        // logOutput
        //
        this.logOutput.Anchor = ((System.Windows.Forms.AnchorStyles)((((
System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
        | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.logOutput.BackColor = System.Drawing.SystemColors.HighlightText;
        this.logOutput.Location = new System.Drawing.Point(9, 330);
        this.logOutput.Multiline = true;
        this.logOutput.Name = "logOutput";
        this.logOutput.ReadOnly = true;
        this.logOutput.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
        this.logOutput.Size = new System.Drawing.Size(343, 171);
        this.logOutput.TabIndex = 124;
        //
        // btnSecureMessage
        //
        this.btnSecureMessage.Anchor = ((System.Windows.Forms.AnchorStyles)(((
System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.btnSecureMessage.Location = new System.Drawing.Point(9, 104);
        this.btnSecureMessage.Name = "btnSecureMessage";
        this.btnSecureMessage.Size = new System.Drawing.Size(348, 23);
        this.btnSecureMessage.TabIndex = 123;
        this.btnSecureMessage.Text = "Secure Message Prompt";
        this.btnSecureMessage.UseVisualStyleBackColor = true;
        this.btnSecureMessage.Click += new System.EventHandler(this.btnSecureMessage_Click);
        //
        // btnGetPIN
        //
        this.btnGetPIN.Anchor = ((System.Windows.Forms.AnchorStyles)(((
System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.btnGetPIN.Location = new System.Drawing.Point(9, 75);
        this.btnGetPIN.Name = "btnGetPIN";
        this.btnGetPIN.Size = new System.Drawing.Size(348, 23);
        this.btnGetPIN.TabIndex = 122;
        this.btnGetPIN.Text = "Get PIN Entry";
        this.btnGetPIN.UseVisualStyleBackColor = true;
        this.btnGetPIN.Click += new System.EventHandler(this.btnGetPIN_Click);
        //
        // btnFirmwareVersion
        //
        this.btnFirmwareVersion.Anchor = ((System.Windows.Forms.AnchorStyles)(((
System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.btnFirmwareVersion.Location = new System.Drawing.Point(9, 46);
        this.btnFirmwareVersion.Name = "btnFirmwareVersion";
        this.btnFirmwareVersion.Size = new System.Drawing.Size(348, 23);
        this.btnFirmwareVersion.TabIndex = 121;
        this.btnFirmwareVersion.Text = "Firmware Version";
        this.btnFirmwareVersion.UseVisualStyleBackColor = true;
        this.btnFirmwareVersion.Click += new System.EventHandler(this.btnFirmwareVersion_Click);
        //
        // tbOutput
        //
        this.tbOutput.Anchor = ((System.Windows.Forms.AnchorStyles)(((
System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.tbOutput.BackColor = System.Drawing.SystemColors.HighlightText;
        this.tbOutput.Location = new System.Drawing.Point(9, 162);
        this.tbOutput.Multiline = true;
        this.tbOutput.Name = "tbOutput";
        this.tbOutput.ReadOnly = true;
        this.tbOutput.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
        this.tbOutput.Size = new System.Drawing.Size(343, 162);
        this.tbOutput.TabIndex = 120;
        //
        // rbCOMM
        //
        this.rbCOMM.AutoSize = true;
        this.rbCOMM.Location = new System.Drawing.Point(74, 11);
        this.rbCOMM.Name = "rbCOMM";
        this.rbCOMM.Size = new System.Drawing.Size(52, 17);
        this.rbCOMM.TabIndex = 119;
        this.rbCOMM.Text = "COM:";
```

```
            this.rbCOMM.UseVisualStyleBackColor = true;
            //
            // rbUSB
            //
            this.rbUSB.AutoSize = true;
            this.rbUSB.Location = new System.Drawing.Point(9, 11);
            this.rbUSB.Name = "rbUSB";
            this.rbUSB.Size = new System.Drawing.Size(47, 17);
            this.rbUSB.TabIndex = 117;
            this.rbUSB.Text = "USB";
            this.rbUSB.UseVisualStyleBackColor = true;
            this.rbUSB.CheckedChanged += new System.EventHandler(this.rbUSB_CheckedChanged);
            //
            // tbCOMPort
            //
            this.tbCOMPort.Location = new System.Drawing.Point(127, 10);
            this.tbCOMPort.Name = "tbCOMPort";
            this.tbCOMPort.Size = new System.Drawing.Size(45, 20);
            this.tbCOMPort.TabIndex = 118;
            this.tbCOMPort.Text = "1";
            //
            // cbBaud
            //
            this.cbBaud.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.cbBaud.FormattingEnabled = true;
            this.cbBaud.Items.AddRange(new object[] {
            "Default",
            "2400",
            "4800",
            "9600",
            "19200",
            "38400",
            "115000"});
            this.cbBaud.Location = new System.Drawing.Point(263, 12);
            this.cbBaud.Name = "cbBaud";
            this.cbBaud.Size = new System.Drawing.Size(60, 21);
            this.cbBaud.TabIndex = 82;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(225, 17);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(32, 13);
            this.label1.TabIndex = 83;
            this.label1.Text = "Baud";
            //
            // btnUnsecureMessage
            //
            this.btnUnsecureMessage.Anchor = ((System.Windows.Forms.AnchorStyles)(((
        System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.btnUnsecureMessage.Location = new System.Drawing.Point(9, 133);
            this.btnUnsecureMessage.Name = "btnUnsecureMessage";
            this.btnUnsecureMessage.Size = new System.Drawing.Size(348, 23);
            this.btnUnsecureMessage.TabIndex = 125;
            this.btnUnsecureMessage.Text = "Display Unsecure Message";
            this.btnUnsecureMessage.UseVisualStyleBackColor = true;
            this.btnUnsecureMessage.Click += new System.EventHandler(this.btnUnsecureMessage_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(367, 511);
            this.Controls.Add(this.btnUnsecureMessage);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.logOutput);
            this.Controls.Add(this.cbBaud);
            this.Controls.Add(this.btnSecureMessage);
            this.Controls.Add(this.btnGetPIN);
            this.Controls.Add(this.btnFirmwareVersion);
            this.Controls.Add(this.tbOutput);
            this.Controls.Add(this.rbCOMM);
            this.Controls.Add(this.rbUSB);
            this.Controls.Add(this.tbCOMPort);
            this.Name = "Form1";
            this.Text = "L100";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.TextBox tbCOMPort;
        private System.Windows.Forms.RadioButton rbUSB;
```

```
        private System.Windows.Forms.RadioButton rbCOMM;
        private System.Windows.Forms.TextBox tbOutput;
        private System.Windows.Forms.Button btnFirmwareVersion;
        private System.Windows.Forms.Button btnStartEMVTransaction;
        private System.Windows.Forms.Button btnCancelEMVTransaction;
        private System.Windows.Forms.TextBox logOutput;
        private System.Windows.Forms.Button btnSecureMessage;
        private System.Windows.Forms.Button btnGetPIN;
        private System.Windows.Forms.ComboBox cbBaud;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button btnUnsecureMessage;
    }
}
```

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 L100 singleton object in the class initializer. Reference: Initialize SmartPIN L100
- Implement the radio button methods CheckChanged handlers to set the proper interface for SDK communications

```
private void rbUSB_CheckedChanged(object sender, EventArgs e)
{
    if (rbUSB.Checked)
    {
        IDT_L100.useUSB();
        return;
    }
    if (cbBaud.SelectedIndex == 0) {
        IDT_L100.useSerialPort(Convert.ToInt32(tbCOMPort.Text), false);
    }
    else
    {
        IDT_L100.useSerialPort(Convert.ToInt32(tbCOMPort.Text), Convert.ToInt32(cbBaud.SelectedItem), false
      );
    }
}
```

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

```
private void btnFirmwareVersion_Click(object sender, EventArgs e)
{
    string firmwareVersion = "";
    RETURN_CODE rt = IDT_L100.SharedController.device_getFirmwareVersion(ref firmwareVersion);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Firmware Ver: " + firmwareVersion + "\r\n");
        System.Diagnostics.Debug.WriteLine("Firmware Ver: " + firmwareVersion);
    }
    else
    {
        tbOutput.AppendText("Get Firmware Fail Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) +
      ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Get Firmware Fail Error Code: " + "0x" + String.Format("
      {0:X}", (ushort)rt));
    }
}

private void btnGetPIN_Click(object sender, EventArgs e)
{
    RETURN_CODE rt = IDT_L100.SharedController.pin_getEncryptedPIN(1, "4111111111111111", "ENTER PIN", 45);
    if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
    {
        tbOutput.AppendText("Enter Pin Executed Successful\r\n");
        System.Diagnostics.Debug.WriteLine("Enter Pin Executed Successful");
    }
    else
    {
        tbOutput.AppendText("Enter Pin Executed failed Error Code: " + "0x" + String.Format("{0:X}", (
      ushort)rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        System.Diagnostics.Debug.WriteLine("Enter Pin Executed failed Error Code: " + "0x" + String.
      Format("{0:X}", (ushort)rt));
    }
```

```
    }

    private void btnSecureMessage_Click(object sender, EventArgs e)
    {
        RETURN_CODE rt = IDT_L100.SharedController.pin_promptForKeyInput(Convert.ToInt32(tbMessageID.Text),
          Convert.ToInt32(tbLanguageID.Text), cbMaskInput.Checked, Convert.ToInt32(tbMinLength.Text), Convert.ToInt32(
          tbMaxLength.Text), Convert.ToInt32(tbPinTimout.Text));
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            tbOutput.AppendText("Prompt for Key Entry Command Successful\r\n");
        }
        else
        {
            tbOutput.AppendText("Prompt for Key Entry Command Error Code: " + "0x" + String.Format("{0:X}", (
          ushort)rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
        }
    }

    private void btnUnsecureMessage_Click(object sender, EventArgs e)
    {
        RETURN_CODE rt = IDT_Device.SharedController.lcd_displayMessage(1, "SmartPIN L100 Ready");
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            tbOutput.AppendText("Display Message Successful\r\n");
            System.Diagnostics.Debug.WriteLine("Display Message Successful");
        }
        else
        {
            tbOutput.AppendText("Display Message failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)
          rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Display Message 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:
            SetOutputText("To connect\r\n");
            break;
        case DeviceState.DefaultDeviceTypeChange:
            SetOutputText("DefaultDeviceTypeChange\r\n");
            break;
        case DeviceState.Connected:
            SetOutputText("Connected\r\n");
            break;
        case DeviceState.Disconnected:
            SetOutputText("Disconnected\r\n");
            break;
        case DeviceState.ConnectionFailed:
            SetOutputText("Connection Failed\r\n");
            break;
        case DeviceState.Notification:
            SetOutputText("Notification\r\n");
            break;
        case DeviceState.TransactionData:
            if (cardData == null) break;
            //output parsed card data
            SetOutputText("Return Code: " + transactionResultCode.ToString() + "\r\n");
            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\nKey Entry: " + cardData.pin_KeyEntry + "\r\n");
                return;

            }
            break;
        case DeviceState.DataReceived:
            SetOutputTextLog(" IN: " + Common.getHexStringFromBytes(data)+ "\r\n");
            break;
        case DeviceState.DataSent:
            SetOutputTextLog(" OUT: " + Common.getHexStringFromBytes(data)+ "\r\n");
            break;
        case DeviceState.CommandTimeout:
            SetOutputText("Command Timeout\r\n");
            break;
        case DeviceState.ToSwipe:
```

```
            SetOutputText("To Swipe\r\n");
            break;
        case DeviceState.MSRDecodeError:
            SetOutputText("MSR Decode Error\r\n");
            break;
        case DeviceState.ToTap:
            SetOutputText("To Tap\r\n");
            break;
        case DeviceState.SwipeTimeout:
            SetOutputText("Swipe Timeout\r\n");
            break;
        case DeviceState.TransactionCancelled:
            System.Diagnostics.Debug.WriteLine("TransactionCancelled\r\n");
            break;
        case DeviceState.DeviceTimeout:
            SetOutputText("Device Timeout\r\n");
            break;
        case DeviceState.TransactionFailed:

            SetOutputText("Transaction Failed: " + IDTechSDK.errorCode.getErrorString(
      transactionResultCode) + "\r\n");
            break;
        case DeviceState.EMVCallback:
            SetOutputText("EMV Callback\r\n");
            break;
        case DeviceState.PINpadKeypress:
            SetOutputText("PINPad Key was pressed\r\n");
            break;
        default:
            break;

    }
}
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);
        }

    }
}
```

Chapter 6

Core Implementation: UWP

IDTechSDK_UWP.dll includes class libraries to interface with the SmartPIN L100. This guide assume 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 SmartPIN L100
- 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 not on the destination system.

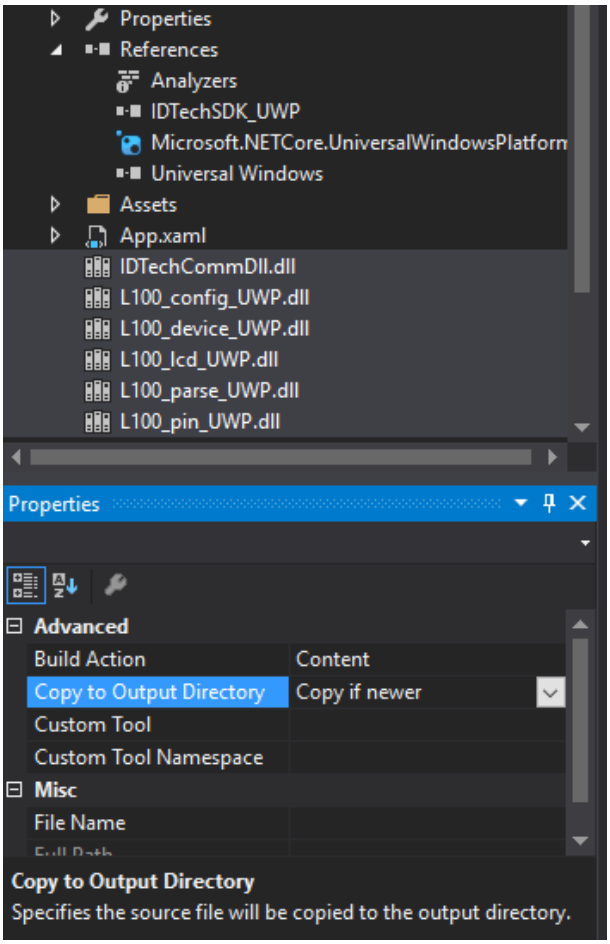
In addition, the following libraries need to be added to project folder and included in with the application distribution:

- IDTechCommDll.dll
- ssleay32.dll
- libeay32.dll
- L100_config_UWP.dll
- L100_device_UWP.dll
- L100_lcd_UWP.dll

- L100_parse_UWP.dll
- L100_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.

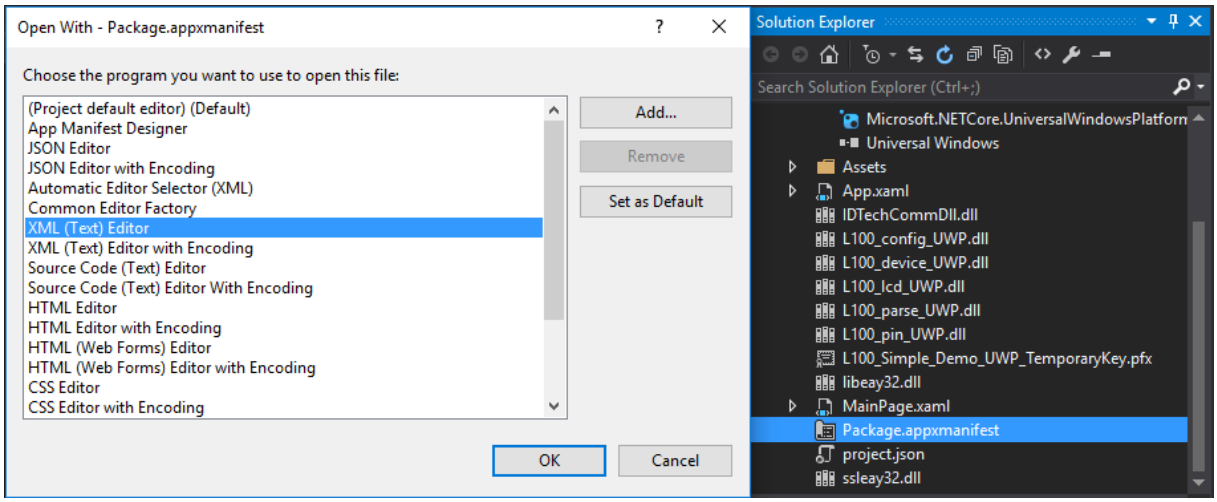
Once all three items are added, set their properties to "Copy if newer" so they will be included in the final applications destination folder.



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">
  <!-- L100 -->
  <Device Id="vidpid:0ACD 1050 usb">
    <Function Type="usage:0002 0026"/>
  </Device>

  <!-- L100 (Alternate) -->
  <Device Id="vidpid:0ACD 0850 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 SmartPIN L100 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="3d59fbde-6761-4aa0-a2a6-519f7c1da11f"
    Publisher="CN=davidt"
    Version="1.0.0.0" />

  <mp:PhoneIdentity PhoneProductId="3d59fbde-6761-4aa0-a2a6-519f7c1da11f" PhonePublisherId="
      00000000-0000-0000-0000-000000000000"/>

  <Properties>
    <DisplayName>L100_Simple_Demo_UWP</DisplayName>
    <PublisherDisplayName>davidt</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="L100_Simple_Demo_UWP.App">
      <uap:VisualElements
        DisplayName="L100_Simple_Demo_UWP"
        Square150x150Logo="Assets\Square150x150Logo.png"
        Square44x44Logo="Assets\Square44x44Logo.png"
        Description="L100_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">
      <!-- L100 -->
      <Device Id="vidpid:0ACD 1050 usb">
        <Function Type="usage:0002 0026"/>
      </Device>

      <!-- L100 (Alternate) -->
      <Device Id="vidpid:0ACD 0850 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 occured 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 beign 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:
            //Device timeout
            break;
        case DeviceState.TransactionFailed:
            //Transaction failed to complete
            break;
        case DeviceState.EMVCallback:
            //Callback during EMV transaction retrieved from EMV_Callback emvCallback
             break;

       case DeviceState.PINpadKeypress:
            //Callback during PINPAD keypress
            break;
        default:
            break;
    }
}
```

6.6 Initialize SmartPIN L100

A Singleton instance has been established in the IDT_L100 class. Establish the callback, and then set a connection to L100 through either USB or Serial.

```
public MainPage()
{
    this.InitializeComponent();
    IDT_L100.setCallback(MessageCallBack);

    //USB Connection:
    Task.Run(() => IDT_L100.useUSB());

    //Serial Connection:
```

```
    //Task.Run(() => IDT_L100.useSerialPort(1, false));
}
```

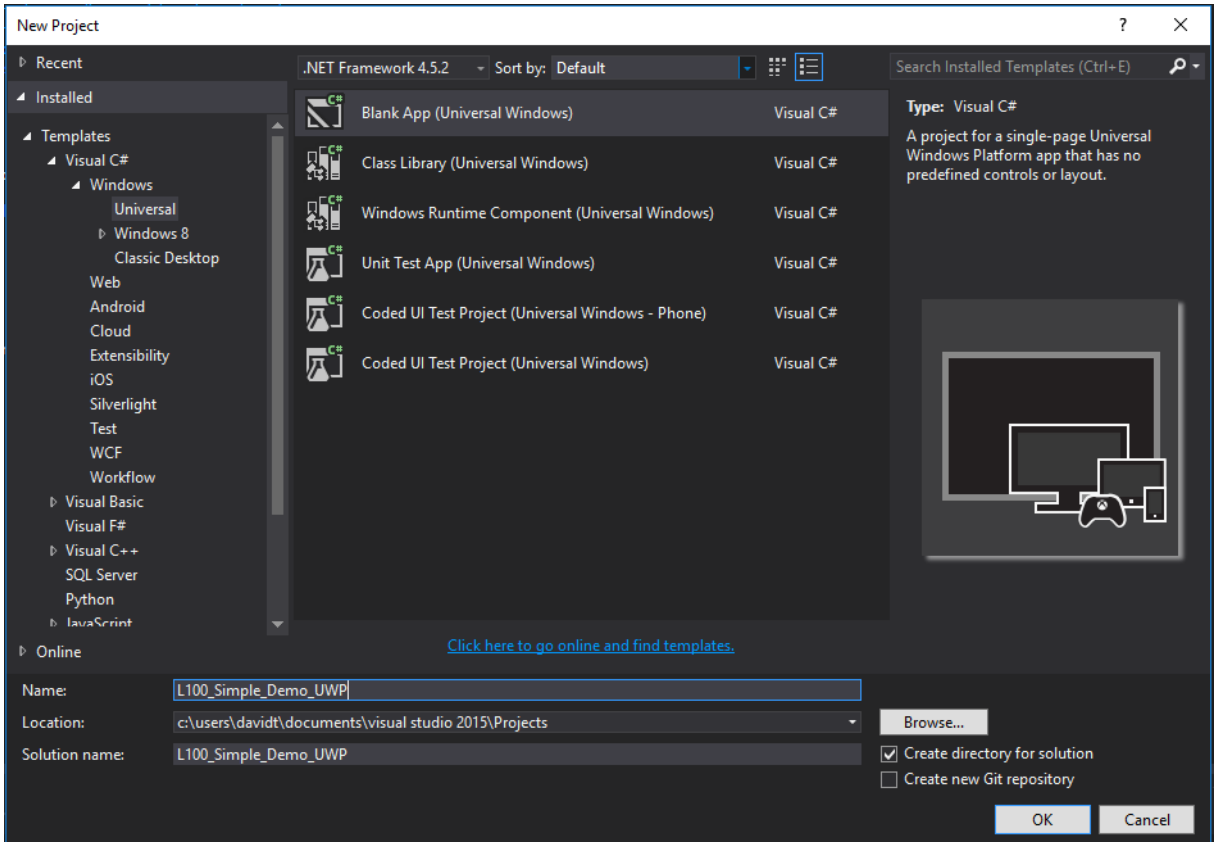
6.7 Sample Project Tutorial

Using Visual Studio 2015, we will create a C# sample project that will interface with the SmartPin L100 and will perform the following activities:

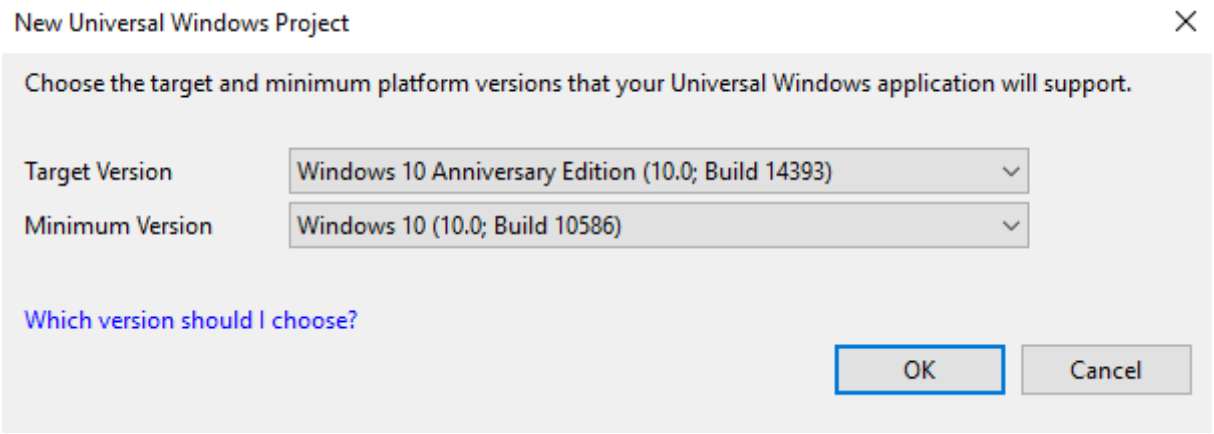
- Get firmware version
- Prompt for PIN Entry and return data
- Display a secure message and prompt for entry
- Display a non-secure message
- Connect to either USB or Serial Versions of L100.
- Show log of all data going to/from L100

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 L100_↩ 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

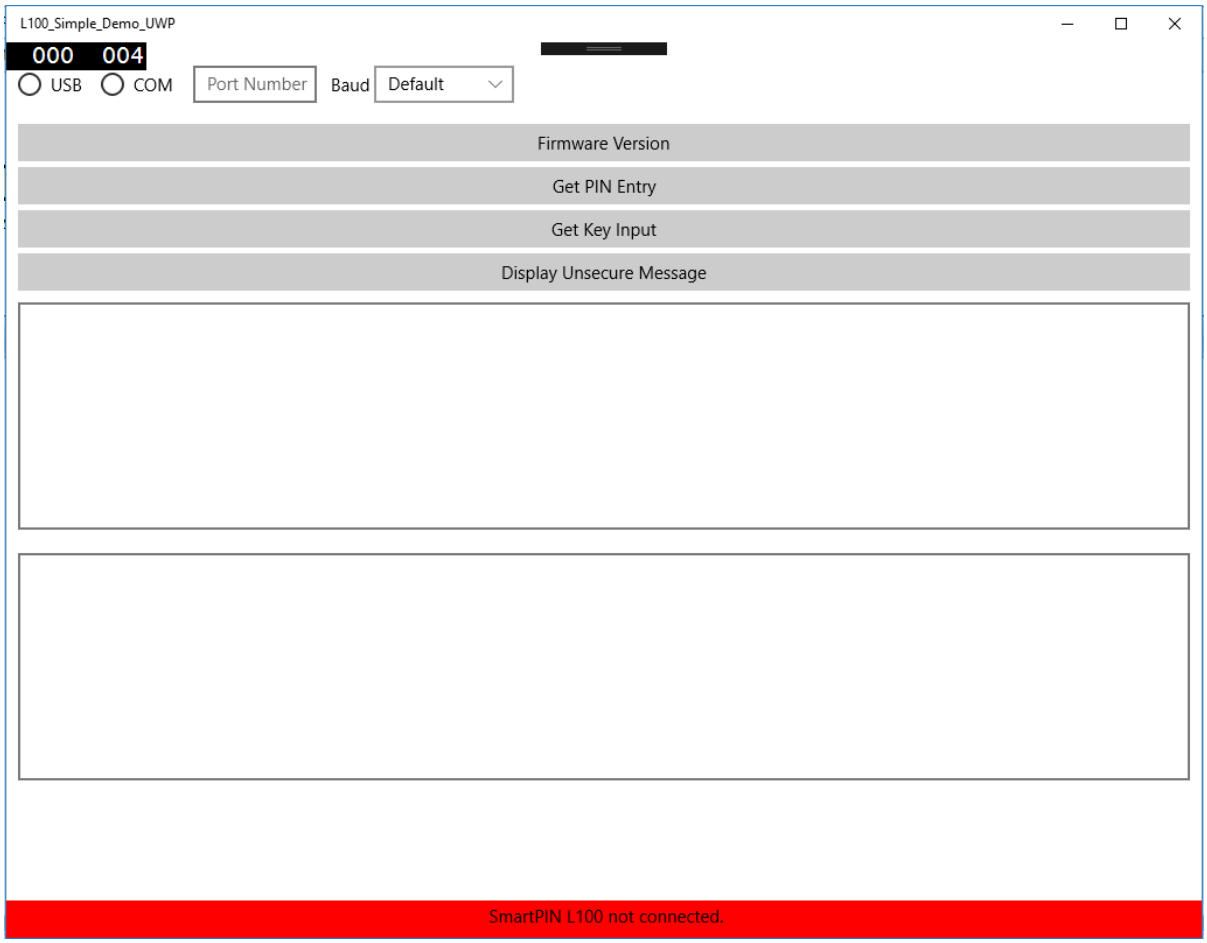
6.7.3 Step 3: Add Device Details

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 a drop-down for baud rate
- Add buttons to execute the following functions:
  - Get Firmware
  - Get PIN Entry
  - Secure Message Prompt
  - Unsecure Message Prompt
- Add a text box to communicate data from the L100.
- Add a text box for the log of L100.
- Add a text box to display the connection status.




```
<Page
    x:Class="L100_Simple_Demo_UWP.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:L100_Simple_Demo_UWP"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <ScrollViewer VerticalScrollBarVisibility="Auto">
            <Grid Height="673" Margin="0,10,0,0" VerticalAlignment="Top">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="*" />
                </Grid.RowDefinitions>
                <Grid Grid.Row="0">
                    <Button x:Name="btnGetFirmwareVersion" Content="Firmware Version" HorizontalAlignment="
      Stretch" Margin="10,60,10,0" VerticalAlignment="Top" Click="btnGetFirmwareVersion_Click"/>
                    <Button x:Name="btnGetPINEntry" Content="Get PIN Entry" HorizontalAlignment="Stretch"
      Margin="10,97,10,0" VerticalAlignment="Top" Click="btnGetPINEntry_Click"/>
                    <Button x:Name="btnGetKeyInput" Content="Get Key Input" HorizontalAlignment="Stretch"
      Margin="10,134,10,0" VerticalAlignment="Top" Click="btnGetKeyInput_Click"/>
                    <Button x:Name="btnDisplayUnsecureMessage" Content="Display Unsecure Message"
      HorizontalAlignment="Stretch" Margin="10,171,10,0" VerticalAlignment="Top" Click="btnDisplayUnsecureMessage_Click"/>
                    <RadioButton x:Name="rbUSB" Content="USB" HorizontalAlignment="Left" Margin="10,10,0,0"
       VerticalAlignment="Top" Click="rbUSB_Click"/>
                    <RadioButton x:Name="rbSerial" Content="COM" HorizontalAlignment="Left" Margin="
      81,10,0,0" VerticalAlignment="Top" Click="rbSerial_Click"/>
                    <TextBox x:Name="tbCOMPort" HorizontalAlignment="Left" Margin="160,10,0,0" TextWrapping
      ="Wrap" VerticalAlignment="Top" PlaceholderText="Port Number"/>
                    <TextBlock x:Name="textBlock" HorizontalAlignment="Left" Margin="278,16,0,0"
      TextWrapping="Wrap" Text="Baud" VerticalAlignment="Top"/>
                    <ComboBox x:Name="comboBoxBaudRate" HorizontalAlignment="Left" Margin="315,10,0,0"
      VerticalAlignment="Top" Width="120" ItemsSource="{Binding}"/>
                </Grid>
```

```
                <Grid Grid.Row="1" Margin="0,0,0,40"  MinHeight="200">
                    <Grid.RowDefinitions>
                        <RowDefinition/>
                        <RowDefinition/>
                    </Grid.RowDefinitions>
                    <TextBox x:Name="tbOutput" Margin="10" TextWrapping="Wrap" ScrollViewer.
      VerticalScrollMode="Auto" ScrollViewer.VerticalScrollBarVisibility="Auto" IsReadOnly="True" Grid.Row="0"/>
                    <TextBox x:Name="logOutput" Margin="10" TextWrapping="Wrap" ScrollViewer.
      VerticalScrollMode="Auto" ScrollViewer.VerticalScrollBarVisibility="Auto" IsReadOnly="True" Grid.Row="1"/>
                </Grid>
            </Grid>
        </ScrollViewer>
        <TextBox x:Name="connectionStatus" TextAlignment="Center" Background="Red" TextWrapping="Wrap" Text
      ="SmartPIN L100 not connected." VerticalAlignment="Bottom" BorderThickness="0" IsHitTestVisible="False"/>
    </Grid>
</Page>
```

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 L100 singleton object in the class initializer. Reference: Initialize SmartPIN L100
- 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_L100.useUSB());
}

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

    if (comboBoxBaudRate.SelectedIndex == 0)
    {
        Task.Run(() => IDT_L100.useSerialPort(portNumber, false));
    }
    else
    {
        Int32 baudRate = Convert.ToInt32(comboBoxBaudRate.SelectedItem);

        Task.Run(() => IDT_L100.useSerialPort(portNumber, baudRate, false));
    }
}
```

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

```
private void btnGetFirmwareVersion_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        string firmwareVersion = "";
        RETURN_CODE rt = IDT_L100.SharedController.device_getFirmwareVersion(ref firmwareVersion);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Firmware Ver: " + firmwareVersion + "\r\n");
            System.Diagnostics.Debug.WriteLine("Firmware Ver: " + firmwareVersion);
        }
        else
        {
            SetOutputText("Get Firmware Fail Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt) + ":
       " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Get Firmware Fail Error Code: " + "0x" + String.
```

```
      Format("{0:X}", (ushort)rt));
        }
    });
}

private void btnGetPINEntry_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_L100.SharedController.pin_getEncryptedPIN(1, "4111111111111111", "ENTER PIN",
      45);
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Enter Pin Executed Successful\r\n");
            System.Diagnostics.Debug.WriteLine("Enter Pin Executed Successful");
        }
        else
        {
            SetOutputText("Enter Pin Executed failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)
      rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Enter Pin Executed failed Error Code: " + "0x" +
      String.Format("{0:X}", (ushort)rt));
        }
    });
}

private void btnGetKeyInput_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_L100.SharedController.pin_promptForKeyInput(Convert.ToInt32(tbMessageID.Text),
       Convert.ToInt32(tbLanguageID.Text), cbMaskInput.Checked, Convert.ToInt32(tbMinLength.Text), Convert.ToInt32
      (tbMaxLength.Text), Convert.ToInt32(tbPinTimout.Text));
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Prompt for Key Entry Command Successful\r\n");
            System.Diagnostics.Debug.WriteLine("Prompt for Key Entry Command Successful");
        }
        else
        {
            SetOutputText("Prompt for Key Entry Command Error Code: " + "0x" + String.Format("{0:X}", (
      ushort)rt) + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Prompt for Key Entry Command Error Code: " + "0x" +
      String.Format("{0:X}", (ushort)rt));
        }
    });
}

private void btnDisplayUnsecureMessage_Click(object sender, RoutedEventArgs e)
{
    Task.Run(() =>
    {
        RETURN_CODE rt = IDT_Device.SharedController.lcd_displayMessage(1, "SmartPIN L100 Ready");
        if (rt == RETURN_CODE.RETURN_CODE_DO_SUCCESS)
        {
            SetOutputText("Display Message Successful\r\n");
            System.Diagnostics.Debug.WriteLine("Display Message Successful");
        }
        else
        {
            SetOutputText("Display Message failed Error Code: " + "0x" + String.Format("{0:X}", (ushort)rt)
       + ": " + IDTechSDK.errorCode.getErrorString(rt) + "\r\n");
            System.Diagnostics.Debug.WriteLine("Display Message 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:
            SetOutputText("To connect\r\n");
            break;
        case DeviceState.DefaultDeviceTypeChange:
            //The SDK is changing the default device to IDT_DEVICE_TYPES type
            SetOutputText("Callback:DefaultDeviceTypeChange\n");
            connectionStatus.Text = "SmartPIN L100 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 = "SmartPIN L100 connected.";
            connectionStatus.Background = new SolidColorBrush(Windows.UI.Colors.Green);
            break;
        case DeviceState.Disconnected:
            //A disconnection has occured with IDT_DEVICE_TYPES type
            SetOutputText("Callback:Disconnected\n");
            connectionStatus.Text = "SmartPIN L100 not connected.";
            connectionStatus.Background = new SolidColorBrush(Windows.UI.Colors.Red);
            break;
        case DeviceState.ConnectionFailed:
            SetOutputText("Connection Failed\r\n");
            break;
        case DeviceState.Notification:
            SetOutputText("Notification\r\n");
            break;
        case DeviceState.TransactionData:
            if (cardData == null) break;
            //output parsed card data
            SetOutputText("Return Code: " + transactionResultCode.ToString() + "\r\n");
            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\nKey Entry: " + cardData.pin_KeyEntry + "\r\n");
                return;
            }
            break;
        case DeviceState.DataReceived:
            SetOutputTextLog(" IN: " + Common.getHexStringFromBytes(data) + "\r\n");
            break;
        case DeviceState.DataSent:
            SetOutputTextLog(" OUT: " + Common.getHexStringFromBytes(data) + "\r\n");
            break;
        case DeviceState.CommandTimeout:
            SetOutputText("Command Timeout\r\n");
            break;
        case DeviceState.ToSwipe:
            SetOutputText("To Swipe\r\n");
            break;
        case DeviceState.MSRDecodeError:
            SetOutputText("MSR Decode Error\r\n");
            break;
        case DeviceState.ToTap:
            SetOutputText("To Tap\r\n");
            break;
        case DeviceState.SwipeTimeout:
            SetOutputText("Swipe Timeout\r\n");
            break;
        case DeviceState.TransactionCancelled:
            System.Diagnostics.Debug.WriteLine("TransactionCancelled\r\n");
            break;
        case DeviceState.DeviceTimeout:
            SetOutputText("Device Timeout\r\n");
            break;
        case DeviceState.TransactionFailed:
            SetOutputText("Transaction Failed: " + IDTechSDK.errorCode.getErrorString(
      transactionResultCode) + "\r\n");
            break;
        case DeviceState.EMVCallback:
            SetOutputText("EMV Callback\r\n");
            break;
        case DeviceState.PINpadKeypress:
            SetOutputText("PINPad Key was pressed\r\n");
            break;
        default:
            break;
    }
}

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

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 8

General Message Table

Secure messages to be used with General Prompts commands

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
1	ENTER	ENTER	INGRESE	ENTREZ
2	REENTER	RE-INTRODUZIR	REINGRESE	RE-ENTREZ
3	ENTER YOUR	INTRODUZIR O SEU	INGRESE SU	ENTREZ VOTRE
4	REENTER YOUR	RE-INTRODUZIR O SEU	REINGRESE SU	RE-ENTREZ VOTRE
5	PLEASE ENTER	POR FAVOR DIGITE	POR FAVOR INGRE↩SE	SVP ENTREZ
6	PLEASE REENTER	POR FAVO REENT↩RAR	POR FAVO REINGR↩ESE	SVP RE-ENTREZ
7	PO NUMBER	NÚMERO PO	NUMERO PO	No COMMANDE
8	DRIVER ID	LICENÇA	LICENCIA	ID CONDUCTEUR
9	ODOMETER	ODOMETER	ODOMETRO	ODOMETRE
10	ID NUMBER	NÚMERO ID	NUMERO ID	No IDENT
11	EQUIP CODE	EQUIP CODE	CODIGO EQUIP	CODE EQUIPEMENT
12	DRIVERS ID	DRIVER ID	ID CONDUCTOR	ID CONDUCTEUR
13	JOB NUMBER	EMP NÚMERO	NUMERO EMP	No TRAVAIL
14	WORK ORDER	TRABALHO ORDEM	ORDEN TRABAJO	FICHE TRAVAIL
15	VEHICLE ID	ID VEÍCULO	ID VEHICULO	ID VEHICULE
16	ENTER DRIVER	ENTER DRIVER	INGRESE CONDUC↩TOR	ENTR CONDUCTEUR
17	ENTER DEPT	ENTER DEPT	INGRESE DEPT	ENTR DEPARTEMNT
18	ENTER PHONE	ADICIONAR PHONE	INGRESE TELEFONO	ENTR No TELEPH
19	ENTER ROUTE	ROUTE ADD	INGRESE RUTA	ENTREZ ROUTE
20	ENTER FLEET	ENTER FROTA	INGRESE FLOTA	ENTREZ PARC AUTO
21	ENTER JOB ID	ENTER JOB ID	INGRESE ID TRAB↩AJO	ENTR ID TRAVAIL
22	ROUTE NUMBER	NÚMERO PATH	RUTA NUMERO	No ROUTE
23	ENTER USER ID	ENTER USER ID	INGRESE ID USUA↩RIO	ID UTILISATEUR
24	FLEET NUMBER	NÚMERO DE FROTA	FLOTA NUMERO	No PARC AUTO
25	ENTER PRODUCT	ADICIONAR PROD↩UTO	INGRESE PRODUC↩TO	ENTREZ PRODUIT
26	DRIVER NUMBER	NÚMERO DRIVER	CONDUCTOR NUM↩ERO	No CONDUCTEUR
27	ENTER LICENSE	ENTER LICENÇA	INGRESE LICENCIA	ENTREZ PERMIS

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
28	ENTER FLEET NO	ENTER NRO FROTA	INGRESE NRO FLOTA	ENT No PARC AUTO
29	ENTER CAR WASH	WASH ENTER	INGRESE LAVADO	ENTREZ LAVE-AUTO
30	ENTER VEHICLE	ENTER VEÍCULO	INGRESE VEHICULO	ENTREZ VEHICULE
31	ENTER TRAILER	TRAILER ENTER	INGRESE TRAILER	ENTREZ REMORQUE
32	ENTER ODOMETER	ENTER ODOMETER	INGRESE ODOMETRO	ENTREZ ODOMETRE
33	DRIVER LICENSE	CARTEIRA DE MOTORISTA	LICENCIA CONDUCTOR	PERMIS CONDUIRE
34	ENTER CUSTOMER	ENTER CLIENTE	INGRESE CLIENTE	ENTREZ CLIENT
35	VEHICLE NUMBER	NÚMERO DO VEÍCULO	VEHICULO NUMERO	No VEHICULE
36	ENTER CUST DATA	ENTER CLIENTE INFO	INGRESE INFO CLIENTE	INFO CLIENT
37	REENTER DRIVID	REENTRAR DRIVER ID	REINGRESE ID CHOFER	RE-ENTR ID COND
38	ENTER USER DATA	ENTER INFO USUÁRIO	INGRESE INFO USUARIO	INFO UTILISATEUR
39	ENTER CUST CODE	ENTER CODE. CLIENTE	INGRESE COD. CLIENTE	ENTR CODE CLIENT
40	ENTER EMPLOYEE	ENTER FUNCIONÁRIO	INGRESE EMPLEADO	ENTREZ EMPLOYE
41	ENTER ID NUMBER	ENTER NÚMERO ID	INGRESE NUMERO ID	ENTREZ No ID
42	ENTER DRIVER ID	ENTER ID DRIVER	INGRESE ID CONDUCTOR	No CONDUCTEUR
43	ENTER FLEET PIN	ENTER PIN FROTA	INGRESE PIN DE FLOTA	NIP PARC AUTO
44	ODOMETER NUMBER	NÚMERO ODOMETER	ODOMETRO NUMERO	No ODOMETRE
45	ENTER DRIVER LIC	ENTER DRIVER LIC	INGRESE LIC CONDUCTOR	PERMIS CONDUIRE
46	ENTER TRAILER NO	NRO TRAILER ENTER	INGRESE NRO TRAILER	ENT No REMORQUE
47	REENTER VEHICLE	REENTRAR VEÍCULO	REINGRESE VEHICULO	RE-ENTR VEHICULE
48	ENTER VEHICLE ID	ENTER VEÍCULO ID	INGRESE ID VEHICULO	ENTR ID VEHICULE
49	ENTER BIRTH DATE	INSERIR DATA NAC	INGRESE FECHA NAC	ENT DT NAISSANCE
50	ENTER DOB MMDDYY	ENTER FDN MMDDYY	INGRESE FDN MMDDAA	NAISSANCE MMJJAA
51	ENTER FLEET DATA	ENTER FROTA INFO	INGRESE INFO DE FLOTA	INFO PARC AUTO
52	ENTER REFERENCE	ENTER REFERÊNCIA	INGRESE REFERENCIA	ENTREZ REFERENCE
53	ENTER AUTH NUMBR	ENTER NÚMERO AUT	INGRESE NUMERO AUT	No AUTORISATION
54	ENTER HUB NUMBER	ENTER HUB NRO	INGRESE NRO HUB	ENTREZ No NOYAU
55	ENTER HUBOMETER	MEDIDA PARA ENTRAR HUB	INGRESE MEDIDO DE HUB	COMPTEUR NOYAU
56	ENTER TRAILER ID	TRAILER ENTER ID	INGRESE ID TRAILER	ENT ID REMORQUE

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
57	ODOMETER READI↩ NG	QUILOMETRAGEM	LECTURA ODOME↩ TRO	LECTURE ODOME↩ TRE
58	REENTER ODOME↩ TER	REENTRAR ODOM↩ ETER	REINGRESE ODOM↩ ETRO	RE-ENT ODOMETRE
59	REENTER DRIV. ID	REENTRAR DRIVER ID	REINGRESE ID CH↩ OFER	RE-ENT ID CONDUC
60	ENTER CUSTOMER ID	ENTER CLIENTE ID	INGRESE ID CLIEN↩ TE	ENTREZ ID CLIENT
61	ENTER CUST. ID	ENTER CLIENTE ID	INGRESE ID CLIEN↩ TE	ENTREZ ID CLIENT
62	ENTER ROUTE NUM	ENTER NUM ROUTE	INGRESE NUM RUTA	ENT No ROUTE
63	ENTER FLEET NUM	FROTA ENTER NUM	INGRESE NUM FLO↩ TA	ENT No PARC AUTO
64	FLEET PIN	FROTA PIN	PIN DE FLOTA	NIP PARC AUTO
65	DRIVER #	DRIVER #	CONDUCTOR #	CONDUCTEUR
66	ENTER DRIVER #	ENTER DRIVER #	INGRESE CONDUC↩ TOR #	ENT # CONDUCTEUR
67	VEHICLE #	VEÍCULO #	VEHICULO #	# VEHICULE
68	ENTER VEHICLE #	ENTER VEÍCULO #	INGRESE VEHICULO #	ENT # VEHICULE
69	JOB #	TRABALHO #	TRABAJO #	# TRAVAIL
70	ENTER JOB #	ENTER JOB #	INGRESE TRABAJO #	ENTREZ # TRAVAIL
71	DEPT NUMBER	NÚMERO DEPT	NUMERO DEPTO	No DEPARTEMENT
72	DEPARTMENT #	DEPARTAMENTO #	DEPARTAMENTO #	DEPARTEMENT
73	ENTER DEPT #	ENTER DEPT #	INGRESE DEPTO #	ENT# DEPARTEME↩ NT
74	LICENSE NUMBER	NÚMERO DE LICE↩ NÇA	NUMERO LICENCIA	No PERMIS
75	LICENSE #	LICENÇA #	LICENCIA #	# PERMIS
76	ENTER LICENSE #	ENTER LICENÇA #	INGRESE LICENCIA #	ENTREZ # PERMIS
77	DATA	INFO	INFO	INFO
78	ENTER DATA	ENTER INFO	INGRESE INFO	ENTREZ INFO
79	CUSTOMER DATA	CLIENTE INFO	INFO CLIENTE	INFO CLIENT
80	ID #	ID #	ID #	# ID
81	ENTER ID #	ENTER ID #	INGRESE ID #	ENTREZ # ID
82	USER ID	USER ID	ID USUARIO	ID UTILISATEUR
83	ROUTE #	ROUTE #	RUTA #	# ROUTE
84	ENTER ROUTE #	ADD ROUTE #	INGRESE RUTA #	ENTREZ # ROUTE
85	ENTER CARD NUM	ENTER NÚMERO DE CARTÃO	INGRESE NUM TAR↩ JETA	ENTREZ NO CARTE
86	EXP DATE(YYMM)	VALIDADE VAL (AA↩ MM)	FECHA EXP (AAMM)	DATE EXPIR(AAMM)
87	PHONE NUMBER	TELEFONE	NUMERO TELEFONO	NO TEL
88	CVV START DATE	CVV DATA DE INÍCIO	CVV FECHA INICIO	CVV DATE DE DEB↩ UT
89	ISSUE NUMBER	NÚMERO DE EMIS↩ SÃO	NUMERO DE EMISI↩ ON	NO DEMISSION
90	START DATE (MMYY)	DATA DE INÍCIO (A↩ AMM)	FECHA INICIO (AA↩ MM)	DATE DE DEBUT-A↩ AMM

Chapter 9

Namespace Index

9.1 Packages

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

Chapter 10

Class Index

10.1 Class List

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

Chapter 11

Namespace Documentation

11.1 IDTechSDK Namespace Reference

Classes

- class EMV_Callback
- class IDT_L100
- class IDTTransactionData

Enumerations

- enum EMV_ENCRYPTION_MODE { **EMV_ENCRYPTION_MODE_UNKNOWN** = 0, **EMV_ENCRYPTION_MODE_TDES** = 1, **EMV_ENCRYPTION_MODE_AES** = 2 }
- 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_AMEX** = 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_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_CANCELED** = 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_↩ 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_CRYPTOGRAM_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_PRESENCED = 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_PRESENCED** = 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 }

11.1.1 Enumeration Type Documentation

11.1.1.1 enum IDTechSDK.CTLS_APPLICATION [strong]

Define CTLS_APPLICATION.

11.1.1.2 enum IDTechSDK.EMV_CALLBACK_TYPE [strong]

Define EMV_CALLBACK_TYPES.

11.1.1.3 enum IDTechSDK.EMV_ENCRYPTION_MODE [strong]

Define EMV_CALLBACK_TYPES.

11.1.1.4 enum IDTechSDK.EMV_LCD_DISPLAY_MODE [strong]

Define EMV_LCD_DISPLAY_MODE.

11.1.1.5 enum IDTechSDK.EMV_PIN_MODE [strong]

Define EMV_PIN_MODE.

11.1.1.6 enum IDTechSDK.EMV_RESULT_CODE [strong]

Define EMV_PIN_MODE.

Chapter 12

Class Documentation

12.1 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

12.1.1 Detailed Description

Class for LCD Message

12.1.2 Member Data Documentation

12.1.2.1 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

12.1.2.2 byte [ ] IDTechSDK.EMV_Callback.language

Message Language

2 Bytes

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

12.1.2.3 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

12.1.2.4 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_LANG↩ UAGE_SELECT: Language selection, response required with selected language index # 16 - LCD_DISPLAY_M↩ ODE_CLEAR_SCREEN: Request to clear LCD screen of information

12.1.2.5 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.

12.1.2.6 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.

12.1.2.7 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

12.1.2.8 bool IDTechSDK.EMV_Callback.maskEntry

Mask Entry

If True, keypad entry should be masked with '∗'

12.1.2.9 int IDTechSDK.EMV_Callback.msr_displayMessage

MSR Message

Message to display during swipe request

12.1.2.10 int IDTechSDK.EMV_Callback.msr_swipeTimeout

Swipe Timeout

Timeout value waiting for MSR Swipe

12.1.2.11 int IDTechSDK.EMV_Callback.pin_entryInterval

PIN Entry Interval Timeout value of interval between input each PIN

12.1.2.12 int IDTechSDK.EMV_Callback.pin_entryStartTimeout

PIN Entry Start Timeout

Timeout value waiting for PIN entry to start

12.1.2.13 byte [ ] IDTechSDK.EMV_Callback.pin_KSN

PIN KSN

Pairing DUKPT KSN

12.1.2.14 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_O↩
NLINE_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

12.1.2.15 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

12.2 IDTechSDK.IDT_L100 Class Reference

Public Member Functions

- RETURN_CODE device_getFirmwareVersion (ref string response)
- RETURN_CODE device_sendDataCommand (string cmd, bool calcLRC, ref byte[ ] response)
- RETURN_CODE config_getModelNumber (ref string response)
- RETURN_CODE config_getSerialNumber (ref string response)

- bool config_setCmdTimeOutDuration (int newTimeOut)
- RETURN_CODE pin_getEncryptedPIN (int keyType, string PAN, string message, int timeout)
- RETURN_CODE pin_getFunctionKey (int timeout)
- RETURN_CODE pin_sendBeep (int frequency, int duration)
- RETURN_CODE device_rebootDevice ()
- RETURN_CODE pin_cancelPINEntry ()
- RETURN_CODE device_getKeyStatus (ref byte[] status)
- RETURN_CODE pin_promptForKeyInput (int messageID, int languageID, bool maskInput, int minLen, int maxLen, int timeout)
- RETURN_CODE pin_promptForAmountInput (int messageID, int languageID, int minLen, int maxLen, int timeout)
- RETURN_CODE device_setSleepModeTime (int time)
- RETURN_CODE device_startRKI ()
- RETURN_CODE device_enterStopMode ()
- RETURN_CODE device_setDateTime ()
- RETURN_CODE device_getDateTime (ref byte[] dateTime)
- RETURN_CODE lcd_clearDisplay (int lineNumber)
- RETURN_CODE lcd_clearAllLines ()
- RETURN_CODE lcd_savePrompt (int promptNumber, string prompt)
- RETURN_CODE lcd_displayPrompt (int promptNumber, int lineNumber)
- RETURN_CODE lcd_displayMessage (int lineNumber, string message)
- RETURN_CODE lcd_enableBacklight (bool enable)
- RETURN_CODE lcd_getBacklightStatus (ref bool enabled)
- RETURN_CODE config_setBaudRate (int baud)
- RETURN_CODE config_getBaudRate (ref int baud)

Static Public Member Functions

- static bool useSerialPort (int port, bool isSRED)
- static bool useSerialPort (int port, int baud, bool isSRED)
- static bool useUSB ()
- static bool closeSerialPort ()
- static bool closeUSB ()
- static void setCallback (CallBack my_Callback)
- static void setCallback (IntPtr my_Callback, SynchronizationContext context)
- static String SDK_Version ()

Properties

- static IDT_L100 SharedController `[get]`

12.2.1 Member Function Documentation

12.2.1.1 static bool IDTechSDK.IDT_L100.closeSerialPort ( ) `[static]`

Close Serial Port Interface

Instructs SDK to close the Serial Port if connected to L100

Returns

bool TRUE=successful, FALSE=failure

12.2.1.2 static bool IDTechSDK.IDT_L100.closeUSB ( ) [static]

Close USB

Instructs SDK to close the USB if connected to L100

Returns

bool TRUE=successful, FALSE=failure

12.2.1.3 RETURN_CODE IDTechSDK.IDT_L100.config_getBaudRate (ref int *baud*)

Get Baud Rate

Gets the buad rate for RS-232 communication.

Parameters

baud	• 2 = 2400 • 3 = 4800 • 4 = 9600 • 6 = 19200 • 7 = 38400 • 9 = 115200

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.4 RETURN_CODE IDTechSDK.IDT_L100.config_getModelNumber (ref string *response*)

Polls device for Model Number

Parameters

response	Returns Model Number

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.5 RETURN_CODE IDTechSDK.IDT_L100.config_getSerialNumber (ref string *response*)

Polls device for Serial Number

Parameters

response	Returns Serial Number

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.6 RETURN_CODE IDTechSDK.IDT_L100.config_setBaudRate (int *baud*)

Set Baud Rate

Sets the buad rate for RS-232 communication.

Parameters

baud	• 2 = 2400 • 3 = 4800 • 4 = 9600 • 6 = 19200 • 7 = 38400 • 9 = 115200

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.7 bool IDTechSDK.IDT_L100.config_setCmdTimeOutDuration (int *newTimeOut*)

Command Acknowledgement Timout

Sets the amount of seconds to wait for an {ACK} to a command before a timeout. Responses should normally be received under one second. Default is 3 seconds

Parameters

newTimeOut	Timout value. Valid range 1 - 60 seconds

Returns

Success flag. Determines if value was set and in range.

12.2.1.8 RETURN_CODE IDTechSDK.IDT_L100.device_enterStopMode ( )

Enter Stop Mode

Set device enter to stio mode. In stop mode, LCD display and backlight is off. Stop mode reduces power consumption to the lowest possible level. A unit in stop mode can only be woken up by a physical key press.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.9 RETURN_CODE IDTechSDK.IDT_L100.device_getDateTime (ref byte[] *dateTime*)

Get Date Time

Gets current system date and time of the device.

Parameters

dateTime	The date time returned as follows: • byte 0: Year 00-99 • byte 1: Month 01-12 • byte 2: Date 01-31 • byte 3: Hour 00-23 • byte 4: Minute 00-59 • byte 5: Second 00-59

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.10 RETURN_CODE IDTechSDK.IDT_L100.device_getFirmwareVersion (ref string *response*)

Polls device for Firmware Version

Parameters

response	Response returned of Firmware Version

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.11 RETURN_CODE IDTechSDK.IDT_L100.device_getKeyStatus (ref byte[] *status*)

Get Key Status

Gets the status of loaded keys

Parameters

status	byte 0: PIN DUKPT Key, 1 Exists, 0 None, 0xFF STOP byte 1: PIN Master Key, 1 Exists, 0 None byte 2: PIN Session Key, 1 Exists, 0 None byte 3: Account/MSR DUKPT Key, Does not support, always 0 byte 4: Account/ICC DUKPT Key, Does not support, always 0 byte 5: Admin DUKPT Key, 1 Exists, 0 None, 0xFF STOP

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.12 RETURN_CODE IDTechSDK.IDT_L100.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

12.2.1.13 RETURN_CODE IDTechSDK.IDT_L100.device_sendDataCommand (string *cmd,* bool *calcLRC,* ref byte[ ] *response*)

Send a NSData object to device

Sends a command represented by the provide NSData object to the device through the accessory protocol.

Parameters

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

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.14 RETURN_CODE IDTechSDK.IDT_L100.device_setDateTime ( )

Set Date Time

Set current system date and time to the device.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.15 RETURN_CODE IDTechSDK.IDT_L100.device_setSleepModeTime (int *time*)

Set Sleep Mode Timer

Set device enter to sleep mode after the given time. In sleep mode, LCD display and backlight is off. Sleep mode reduces power consumption to the lowest possible level. A unit in Sleep mode can only be woken up by a physical key press.

Parameters

time	Enter sleep time value, in second.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.16 RETURN_CODE IDTechSDK.IDT_L100.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

12.2.1.17 RETURN_CODE IDTechSDK.IDT_L100.lcd_clearAllLines ( )

Clear LCD Display

Clears all lines of the LCD Display.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.18 RETURN_CODE IDTechSDK.IDT_L100.lcd_clearDisplay (int *lineNumber*)

Clear LCD Display Line

Clears the line number of the LCD Display.

Parameters

lineNumber	Line number to clear (1-4)

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.19 RETURN_CODE IDTechSDK.IDT_L100.lcd_displayMessage (int *lineNumber,* string *message*)

Display Message on Line

Displays a message on a display line.

Parameters

lineNumber	Line number to display message on (1-4)
message	Message to display

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.20 RETURN_CODE IDTechSDK.IDT_L100.lcd_displayPrompt (int *promptNumber,* int *lineNumber*)

Display Prompt on Line

Displays a message prompt from L100 memory.

Parameters

promptNumber	Prompt number (0-9)
lineNumber	Line number to display message prompt (1-4)

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.21 RETURN_CODE IDTechSDK.IDT_L100.lcd_enableBacklight (bool *enable*)

Enable/Disable LCD Backlight

Turns on/off the LCD back lighting.

Parameters

enable	TRUE = turn ON backlight, FALSE = turn OFF backlight

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.22 RETURN_CODE IDTechSDK.IDT_L100.lcd_getBacklightStatus (ref bool *enabled*)

Get Backlight Status

Returns the status of the LCD back lighting.

Parameters

enabled	TRUE = Backlight is ON, FALSE = Backlight is OFF

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.23 RETURN_CODE IDTechSDK.IDT_L100.lcd_savePrompt (int *promptNumber,* string *prompt*)

Save Prompt

Saves a message prompt to L100 memory.

Parameters

promptNumber	Prompt number (0-9)
prompt	Prompt string (up to 20 characters)

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.24 RETURN_CODE IDTechSDK.IDT_L100.pin_cancelPINEntry ( )

Cancel PIN Entry

Cancel “Get Function Key” & “Get Encrypted PIN” & “Get Numeric” & “Get Amount”

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.25 RETURN_CODE IDTechSDK.IDT_L100.pin_getEncryptedPIN (int *keyType,* string *PAN,* string *message,* int *timeout*)

Get Encrypted PIN

Requests PIN Entry

Parameters

keyType	• 0x00- MKSK-TDES: External Plaintext PAN • 0x01- DUKPT-TDES: External Plaintext PAN • 0x10 MKSK-TDES: External Ciphertext PAN • 0x11 DUKPT-TDES: External Ciphertext PAN
PAN	Account Number
message	Message to display
timeout	PIN entry timout

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.26 RETURN_CODE IDTechSDK.IDT_L100.pin_getFunctionKey (int *timeout*)

Get Function Key

Captures a function key

- Backspace = B
- Cancel = C
- Enter = E
- * = *
- # = #
- Help = ?
- Function Key 1 = F1

- Function Key 2 = F2
- Function Key 3 = F3

Parameters

timeout	Timeout, in seconds

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.2.1.27 RETURN_CODE IDTechSDK.IDT_L100.pin_promptForAmountInput (int *messageID,* int *languageID,* int *minLen,* int *maxLen,* int *timeout*)

Prompt for Amount Input

Prompts for amount input using the secure message according to the following table

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
1	ENTER	ENTER	INGRESE	ENTREZ
2	REENTER	RE-INTRODUZIR	REINGRESE	RE-ENTREZ
3	ENTER YOUR	INTRODUZIR O SEU	INGRESE SU	ENTREZ VOTRE
4	REENTER YOUR	RE-INTRODUZIR O SEU	REINGRESE SU	RE-ENTREZ VOTRE
5	PLEASE ENTER	POR FAVOR DIGITE	POR FAVOR INGRESE	SVP ENTREZ
6	PLEASE REENTER	POR FAVO REENTRAR	POR FAVO REINGRESE	SVP RE-ENTREZ
7	PO NUMBER	NÚMERO PO	NUMERO PO	No COMMANDE
8	DRIVER ID	LICENÇA	LICENCIA	ID CONDUCTEUR
9	ODOMETER	ODOMETER	ODOMETRO	ODOMETRE
10	ID NUMBER	NÚMERO ID	NUMERO ID	No IDENT
11	EQUIP CODE	EQUIP CODE	CODIGO EQUIP	CODE EQUIPEMENT
12	DRIVERS ID	DRIVER ID	ID CONDUCTOR	ID CONDUCTEUR
13	JOB NUMBER	EMP NÚMERO	NUMERO EMP	No TRAVAIL
14	WORK ORDER	TRABALHO ORDEM	ORDEN TRABAJO	FICHE TRAVAIL
15	VEHICLE ID	ID VEÍCULO	ID VEHICULO	ID VEHICULE
16	ENTER DRIVER	ENTER DRIVER	INGRESE CONDUCTOR	ENTR CONDUCTEUR
17	ENTER DEPT	ENTER DEPT	INGRESE DEPT	ENTR DEPARTEMNT
18	ENTER PHONE	ADICIONAR PHONE	INGRESE TELEFONO	ENTR No TELEPH
19	ENTER ROUTE	ROUTE ADD	INGRESE RUTA	ENTREZ ROUTE
20	ENTER FLEET	ENTER FROTA	INGRESE FLOTA	ENTREZ PARC AUTO
21	ENTER JOB ID	ENTER JOB ID	INGRESE ID TRABAJO	ENTR ID TRAVAIL
22	ROUTE NUMBER	NÚMERO PATH	RUTA NUMERO	No ROUTE
23	ENTER USER ID	ENTER USER ID	INGRESE ID USUARIO	ID UTILISATEUR
24	FLEET NUMBER	NÚMERO DE FROTA	FLOTA NUMERO	No PARC AUTO
25	ENTER PRODUCT	ADICIONAR PRODUTO	INGRESE PRODUCTO	ENTREZ PRODUIT
26	DRIVER NUMBER	NÚMERO DRIVER	CONDUCTOR NUMERO	No CONDUCTEUR

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
27	ENTER LICENSE	ENTER LICENÇA	INGRESE LICENCIA	ENTREZ PERMIS
28	ENTER FLEET NO	ENTER NRO FROTA	INGRESE NRO FLO↩ TA	ENT No PARC AUTO
29	ENTER CAR WASH	WASH ENTER	INGRESE LAVADO	ENTREZ LAVE-AUTO
30	ENTER VEHICLE	ENTER VEÍCULO	INGRESE VEHICULO	ENTREZ VEHICULE
31	ENTER TRAILER	TRAILER ENTER	INGRESE TRAILER	ENTREZ REMORQUE
32	ENTER ODOMETER	ENTER ODOMETER	INGRESE ODOMET↩ RO	ENTREZ ODOMETRE
33	DRIVER LICENSE	CARTEIRA DE MOT↩ ORISTA	LICENCIA CONDUC↩ TOR	PERMIS CONDUIRE
34	ENTER CUSTOMER	ENTER CLIENTE	INGRESE CLIENTE	ENTREZ CLIENT
35	VEHICLE NUMBER	NÚMERO DO VEÍC↩ ULO	VEHICULO NUMERO	No VEHICULE
36	ENTER CUST DATA	ENTER CLIENTE IN↩ FO	INGRESE INFO CLI↩ ENTE	INFO CLIENT
37	REENTER DRIVID	REENTRAR DRIVER ID	REINGRESE ID CH↩ OFER	RE-ENTR ID COND
38	ENTER USER DATA	ENTER INFO USUÁ↩ RIO	INGRESE INFO US↩ UARIO	INFO UTILISATEUR
39	ENTER CUST CODE	ENTER CODE. CLIE↩ NTE	INGRESE COD. CLI↩ ENTE	ENTR CODE CLIENT
40	ENTER EMPLOYEE	ENTER FUNCIONÁ↩ RIO	INGRESE EMPLEA↩ DO	ENTREZ EMPLOYE
41	ENTER ID NUMBER	ENTER NÚMERO ID	INGRESE NUMERO ID	ENTREZ No ID
42	ENTER DRIVER ID	ENTER ID DRIVER	INGRESE ID COND↩ UCTOR	No CONDUCTEUR
43	ENTER FLEET PIN	ENTER PIN FROTA	INGRESE PIN DE F↩ LOTA	NIP PARC AUTO
44	ODOMETER NUMB↩ ER	NÚMERO ODOMET↩ ER	ODOMETRO NUME↩ RO	No ODOMETRE
45	ENTER DRIVER LIC	ENTER DRIVER LIC	INGRESE LIC CON↩ DUCTOR	PERMIS CONDUIRE
46	ENTER TRAILER NO	NRO TRAILER ENT↩ ER	INGRESE NRO TRA↩ ILER	ENT No REMORQUE
47	REENTER VEHICLE	REENTRAR VEÍCULO	REINGRESE VEHIC↩ ULO	RE-ENTR VEHICULE
48	ENTER VEHICLE ID	ENTER VEÍCULO ID	INGRESE ID VEHIC↩ ULO	ENTR ID VEHICULE
49	ENTER BIRTH DATE	INSERIR DATA NAC	INGRESE FECHA N↩ AC	ENT DT NAISSANCE
50	ENTER DOB MMDD↩ YY	ENTER FDN MMDD↩ YY	INGRESE FDN MM↩ DDAA	NAISSANCE MMJJAA
51	ENTER FLEET DATA	ENTER FROTA INFO	INGRESE INFO DE FLOTA	INFO PARC AUTO
52	ENTER REFERENCE	ENTER REFERÊNCIA	INGRESE REFERE↩ NCIA	ENTREZ REFEREN↩ CE
53	ENTER AUTH NUM↩ BR	ENTER NÚMERO A↩ UT	INGRESE NUMERO AUT	No AUTORISATION
54	ENTER HUB NUMB↩ ER	ENTER HUB NRO	INGRESE NRO HUB	ENTREZ No NOYAU
55	ENTER HUBOMETER	MEDIDA PARA ENT↩ RAR HUB	INGRESE MEDIDO DE HUB	COMPTEUR NOYAU

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
56	ENTER TRAILER ID	TRAILER ENTER ID	INGRESE ID TRAIL↩ ER	ENT ID REMORQUE
57	ODOMETER READI↩ NG	QUILOMETRAGEM	LECTURA ODOME↩ TRO	LECTURE ODOME↩ TRE
58	REENTER ODOME↩ TER	REENTRAR ODOM↩ ETER	REINGRESE ODOM↩ ETRO	RE-ENT ODOMETRE
59	REENTER DRIV. ID	REENTRAR DRIVER ID	REINGRESE ID CH↩ OFER	RE-ENT ID CONDUC
60	ENTER CUSTOMER ID	ENTER CLIENTE ID	INGRESE ID CLIEN↩ TE	ENTREZ ID CLIENT
61	ENTER CUST. ID	ENTER CLIENTE ID	INGRESE ID CLIEN↩ TE	ENTREZ ID CLIENT
62	ENTER ROUTE NUM	ENTER NUM ROUTE	INGRESE NUM RUTA	ENT No ROUTE
63	ENTER FLEET NUM	FROTA ENTER NUM	INGRESE NUM FLO↩ TA	ENT No PARC AUTO
64	FLEET PIN	FROTA PIN	PIN DE FLOTA	NIP PARC AUTO
65	DRIVER #	DRIVER #	CONDUCTOR #	CONDUCTEUR
66	ENTER DRIVER #	ENTER DRIVER #	INGRESE CONDUC↩ TOR #	ENT # CONDUCTEUR
67	VEHICLE #	VEÍCULO #	VEHICULO #	# VEHICULE
68	ENTER VEHICLE #	ENTER VEÍCULO #	INGRESE VEHICULO #	ENT # VEHICULE
69	JOB #	TRABALHO #	TRABAJO #	# TRAVAIL
70	ENTER JOB #	ENTER JOB #	INGRESE TRABAJO #	ENTREZ # TRAVAIL
71	DEPT NUMBER	NÚMERO DEPT	NUMERO DEPTO	No DEPARTEMENT
72	DEPARTMENT #	DEPARTAMENTO #	DEPARTAMENTO #	DEPARTEMENT
73	ENTER DEPT #	ENTER DEPT #	INGRESE DEPTO #	ENT# DEPARTEME↩ NT
74	LICENSE NUMBER	NÚMERO DE LICE↩ NÇA	NUMERO LICENCIA	No PERMIS
75	LICENSE #	LICENÇA #	LICENCIA #	# PERMIS
76	ENTER LICENSE #	ENTER LICENÇA #	INGRESE LICENCIA #	ENTREZ # PERMIS
77	DATA	INFO	INFO	INFO
78	ENTER DATA	ENTER INFO	INGRESE INFO	ENTREZ INFO
79	CUSTOMER DATA	CLIENTE INFO	INFO CLIENTE	INFO CLIENT
80	ID #	ID #	ID #	# ID
81	ENTER ID #	ENTER ID #	INGRESE ID #	ENTREZ # ID
82	USER ID	USER ID	ID USUARIO	ID UTILISATEUR
83	ROUTE #	ROUTE #	RUTA #	# ROUTE
84	ENTER ROUTE #	ADD ROUTE #	INGRESE RUTA #	ENTREZ # ROUTE
85	ENTER CARD NUM	ENTER NÚMERO DE CARTÃO	INGRESE NUM TAR↩ JETA	ENTREZ NO CARTE
86	EXP DATE(YYMM)	VALIDADE VAL (AA↩ MM)	FECHA EXP (AAMM)	DATE EXPIR(AAMM)
87	PHONE NUMBER	TELEFONE	NUMERO TELEFONO	NO TEL
88	CVV START DATE	CVV DATA DE INÍCIO	CVV FECHA INICIO	CVV DATE DE DEB↩ UT
89	ISSUE NUMBER	NÚMERO DE EMIS↩ SÃO	NUMERO DE EMISI↩ ON	NO DEMISSION
90	START DATE (MMYY)	DATA DE INÍCIO (A↩ AMM)	FECHA INICIO (AA↩ MM)	DATE DE DEBUT-A↩ AMM

```
@param messageID  Message (1-90)
```

```
@param languageID 0=English Prompt, 1=Portuguese Prompt, 2=Spanish Prompt, 3=French Prompt
@param minLen  Minimum input length.  Cannot be less than 1
@param maxLen Maximum input length.  Cannot be greater than 15
@param timeout Timout value, in seconds


@return RETURN_CODE:  Values can be parsed with errorCode.getErrorString()
```

12.2.1.28 RETURN_CODE IDTechSDK.IDT_L100.pin_promptForKeyInput (int *messageID,* int *languageID,* bool *maskInput,* int *minLen,* int *maxLen,* int *timeout*)

Prompt for Key Input

Prompts for a numeric key using the secure message according to the following table

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
1	ENTER	ENTER	INGRESE	ENTREZ
2	REENTER	RE-INTRODUZIR	REINGRESE	RE-ENTREZ
3	ENTER YOUR	INTRODUZIR O SEU	INGRESE SU	ENTREZ VOTRE
4	REENTER YOUR	RE-INTRODUZIR O SEU	REINGRESE SU	RE-ENTREZ VOTRE
5	PLEASE ENTER	POR FAVOR DIGITE	POR FAVOR INGRESE	SVP ENTREZ
6	PLEASE REENTER	POR FAVO REENTRAR	POR FAVO REINGRESE	SVP RE-ENTREZ
7	PO NUMBER	NÚMERO PO	NUMERO PO	No COMMANDE
8	DRIVER ID	LICENÇA	LICENCIA	ID CONDUCTEUR
9	ODOMETER	ODOMETER	ODOMETRO	ODOMETRE
10	ID NUMBER	NÚMERO ID	NUMERO ID	No IDENT
11	EQUIP CODE	EQUIP CODE	CODIGO EQUIP	CODE EQUIPEMENT
12	DRIVERS ID	DRIVER ID	ID CONDUCTOR	ID CONDUCTEUR
13	JOB NUMBER	EMP NÚMERO	NUMERO EMP	No TRAVAIL
14	WORK ORDER	TRABALHO ORDEM	ORDEN TRABAJO	FICHE TRAVAIL
15	VEHICLE ID	ID VEÍCULO	ID VEHICULO	ID VEHICULE
16	ENTER DRIVER	ENTER DRIVER	INGRESE CONDUCTOR	ENTR CONDUCTEUR
17	ENTER DEPT	ENTER DEPT	INGRESE DEPT	ENTR DEPARTEMNT
18	ENTER PHONE	ADICIONAR PHONE	INGRESE TELEFONO	ENTR No TELEPH
19	ENTER ROUTE	ROUTE ADD	INGRESE RUTA	ENTREZ ROUTE
20	ENTER FLEET	ENTER FROTA	INGRESE FLOTA	ENTREZ PARC AUTO
21	ENTER JOB ID	ENTER JOB ID	INGRESE ID TRABAJO	ENTR ID TRAVAIL
22	ROUTE NUMBER	NÚMERO PATH	RUTA NUMERO	No ROUTE
23	ENTER USER ID	ENTER USER ID	INGRESE ID USUARIO	ID UTILISATEUR
24	FLEET NUMBER	NÚMERO DE FROTA	FLOTA NUMERO	No PARC AUTO
25	ENTER PRODUCT	ADICIONAR PRODUTO	INGRESE PRODUCTO	ENTREZ PRODUIT
26	DRIVER NUMBER	NÚMERO DRIVER	CONDUCTOR NUMERO	No CONDUCTEUR
27	ENTER LICENSE	ENTER LICENÇA	INGRESE LICENCIA	ENTREZ PERMIS
28	ENTER FLEET NO	ENTER NRO FROTA	INGRESE NRO FLOTA	ENT No PARC AUTO
29	ENTER CAR WASH	WASH ENTER	INGRESE LAVADO	ENTREZ LAVE-AUTO
30	ENTER VEHICLE	ENTER VEÍCULO	INGRESE VEHICULO	ENTREZ VEHICULE

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
31	ENTER TRAILER	TRAILER ENTER	INGRESE TRAILER	ENTREZ REMORQUE
32	ENTER ODOMETER	ENTER ODOMETER	INGRESE ODOMET↩ RO	ENTREZ ODOMETRE
33	DRIVER LICENSE	CARTEIRA DE MOT↩ ORISTA	LICENCIA CONDUC↩ TOR	PERMIS CONDUIRE
34	ENTER CUSTOMER	ENTER CLIENTE	INGRESE CLIENTE	ENTREZ CLIENT
35	VEHICLE NUMBER	NÚMERO DO VEÍC↩ ULO	VEHICULO NUMERO	No VEHICULE
36	ENTER CUST DATA	ENTER CLIENTE IN↩ FO	INGRESE INFO CLI↩ ENTE	INFO CLIENT
37	REENTER DRIVID	REENTRAR DRIVER ID	REINGRESE ID CH↩ OFER	RE-ENTR ID COND
38	ENTER USER DATA	ENTER INFO USUÁ↩ RIO	INGRESE INFO US↩ UARIO	INFO UTILISATEUR
39	ENTER CUST CODE	ENTER CODE. CLIE↩ NTE	INGRESE COD. CLI↩ ENTE	ENTR CODE CLIENT
40	ENTER EMPLOYEE	ENTER FUNCIONÁ↩ RIO	INGRESE EMPLEA↩ DO	ENTREZ EMPLOYE
41	ENTER ID NUMBER	ENTER NÚMERO ID	INGRESE NUMERO ID	ENTREZ No ID
42	ENTER DRIVER ID	ENTER ID DRIVER	INGRESE ID COND↩ UCTOR	No CONDUCTEUR
43	ENTER FLEET PIN	ENTER PIN FROTA	INGRESE PIN DE F↩ LOTA	NIP PARC AUTO
44	ODOMETER NUMB↩ ER	NÚMERO ODOMET↩ ER	ODOMETRO NUME↩ RO	No ODOMETRE
45	ENTER DRIVER LIC	ENTER DRIVER LIC	INGRESE LIC CON↩ DUCTOR	PERMIS CONDUIRE
46	ENTER TRAILER NO	NRO TRAILER ENT↩ ER	INGRESE NRO TRA↩ ILER	ENT No REMORQUE
47	REENTER VEHICLE	REENTRAR VEÍCULO	REINGRESE VEHIC↩ ULO	RE-ENTR VEHICULE
48	ENTER VEHICLE ID	ENTER VEÍCULO ID	INGRESE ID VEHIC↩ ULO	ENTR ID VEHICULE
49	ENTER BIRTH DATE	INSERIR DATA NAC	INGRESE FECHA N↩ AC	ENT DT NAISSANCE
50	ENTER DOB MMDD↩ YY	ENTER FDN MMDD↩ YY	INGRESE FDN MM↩ DDAA	NAISSANCE MMJJAA
51	ENTER FLEET DATA	ENTER FROTA INFO	INGRESE INFO DE FLOTA	INFO PARC AUTO
52	ENTER REFERENCE	ENTER REFERÊNCIA	INGRESE REFERE↩ NCIA	ENTREZ REFEREN↩ CE
53	ENTER AUTH NUM↩ BR	ENTER NÚMERO A↩ UT	INGRESE NUMERO AUT	No AUTORISATION
54	ENTER HUB NUMB↩ ER	ENTER HUB NRO	INGRESE NRO HUB	ENTREZ No NOYAU
55	ENTER HUBOMETER	MEDIDA PARA ENT↩ RAR HUB	INGRESE MEDIDO DE HUB	COMPTEUR NOYAU
56	ENTER TRAILER ID	TRAILER ENTER ID	INGRESE ID TRAIL↩ ER	ENT ID REMORQUE
57	ODOMETER READI↩ NG	QUILOMETRAGEM	LECTURA ODOME↩ TRO	LECTURE ODOME↩ TRE
58	REENTER ODOME↩ TER	REENTRAR ODOM↩ ETER	REINGRESE ODOM↩ ETRO	RE-ENT ODOMETRE

Msg Id	English Prompt	Portuguese Prompt	Spanish Prompt	French Prompt
59	REENTER DRIV. ID	REENTRAR DRIVER ID	REINGRESE ID CH↩ OFER	RE-ENT ID CONDUC
60	ENTER CUSTOMER ID	ENTER CLIENTE ID	INGRESE ID CLIEN↩ TE	ENTREZ ID CLIENT
61	ENTER CUST. ID	ENTER CLIENTE ID	INGRESE ID CLIEN↩ TE	ENTREZ ID CLIENT
62	ENTER ROUTE NUM	ENTER NUM ROUTE	INGRESE NUM RUTA	ENT No ROUTE
63	ENTER FLEET NUM	FROTA ENTER NUM	INGRESE NUM FLO↩ TA	ENT No PARC AUTO
64	FLEET PIN	FROTA PIN	PIN DE FLOTA	NIP PARC AUTO
65	DRIVER #	DRIVER #	CONDUCTOR #	CONDUCTEUR
66	ENTER DRIVER #	ENTER DRIVER #	INGRESE CONDUC↩ TOR #	ENT # CONDUCTEUR
67	VEHICLE #	VEÍCULO #	VEHICULO #	# VEHICULE
68	ENTER VEHICLE #	ENTER VEÍCULO #	INGRESE VEHICULO #	ENT # VEHICULE
69	JOB #	TRABALHO #	TRABAJO #	# TRAVAIL
70	ENTER JOB #	ENTER JOB #	INGRESE TRABAJO #	ENTREZ # TRAVAIL
71	DEPT NUMBER	NÚMERO DEPT	NUMERO DEPTO	No DEPARTEMENT
72	DEPARTMENT #	DEPARTAMENTO #	DEPARTAMENTO #	DEPARTEMENT
73	ENTER DEPT #	ENTER DEPT #	INGRESE DEPTO #	ENT# DEPARTEME↩ NT
74	LICENSE NUMBER	NÚMERO DE LICE↩ NÇA	NUMERO LICENCIA	No PERMIS
75	LICENSE #	LICENÇA #	LICENCIA #	# PERMIS
76	ENTER LICENSE #	ENTER LICENÇA #	INGRESE LICENCIA #	ENTREZ # PERMIS
77	DATA	INFO	INFO	INFO
78	ENTER DATA	ENTER INFO	INGRESE INFO	ENTREZ INFO
79	CUSTOMER DATA	CLIENTE INFO	INFO CLIENTE	INFO CLIENT
80	ID #	ID #	ID #	# ID
81	ENTER ID #	ENTER ID #	INGRESE ID #	ENTREZ # ID
82	USER ID	USER ID	ID USUARIO	ID UTILISATEUR
83	ROUTE #	ROUTE #	RUTA #	# ROUTE
84	ENTER ROUTE #	ADD ROUTE #	INGRESE RUTA #	ENTREZ # ROUTE
85	ENTER CARD NUM	ENTER NÚMERO DE CARTÃO	INGRESE NUM TAR↩ JETA	ENTREZ NO CARTE
86	EXP DATE(YYMM)	VALIDADE VAL (AA↩ MM)	FECHA EXP (AAMM)	DATE EXPIR(AAMM)
87	PHONE NUMBER	TELEFONE	NUMERO TELEFONO	NO TEL
88	CVV START DATE	CVV DATA DE INÍCIO	CVV FECHA INICIO	CVV DATE DE DEB↩ UT
89	ISSUE NUMBER	NÚMERO DE EMIS↩ SÃO	NUMERO DE EMISI↩ ON	NO DEMISSION
90	START DATE (MMYY)	DATA DE INÍCIO (A↩ AMM)	FECHA INICIO (AA↩ MM)	DATE DE DEBUT-A↩ AMM

```
@param messageID  Message (1-90)
@param languageID 0=English Prompt, 1=Portuguese Prompt, 2=Spanish Prompt, 3=French Prompt
@param maskInput  TRUE = entry is masked with '*', FALSE = entry is displayed on keypad
@param minLen  Minimum input length.  Cannot be less than 1
@param maxLen Maximum input length.  Cannot be greater than 16
@param timeout Timout value, in seconds


@return RETURN_CODE:  Values can be parsed with errorCode.getErrorString()
```

12.2.1.29 RETURN_CODE IDTechSDK.IDT_L100.pin_sendBeep (int *frequency,* int *duration*)

Send Beep

Executes a beep request.

Parameters

frequency	Frequency, range 200-20000Hz
duration	Duration, range 16-65535ms

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

12.2.1.30 static String IDTechSDK.IDT_L100.SDK_Version ( ) [static]

SDK Version

- All Devices

Returns the current version of SDK

Returns

Framework version

12.2.1.31 static void IDTechSDK.IDT_L100.setCallback (CallBack *my_Callback*) [static]

Set Callback

```
Sets the class callback
```

12.2.1.32 static void IDTechSDK.IDT_L100.setCallback (IntPtr *my_Callback,* SynchronizationContext *context*) [static]

Set Callback

Sets the class callback

Parameters

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

12.2.1.33 static bool IDTechSDK.IDT_L100.useSerialPort (int *port,* bool *isSRED*) [static]

Use Serial Port Interface

Instructs SDK to attempt to use the Serial Port for communication with L100 using default baud rate

Parameters

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

Returns

bool TRUE=successful, FALSE=failure

12.2.1.34 static bool IDTechSDK.IDT_L100.useSerialPort (int *port,* int *baud,* bool *isSRED*) [static]

Use Serial Port Interface with baud rate L100

Parameters

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

Returns

bool TRUE=successful, FALSE=failure

12.2.1.35 static bool IDTechSDK.IDT_L100.useUSB ( ) [static]

Use USB Interface

Instructs SDK to attempt to use USB for communication with IDT_L100

12.2.2 Property Documentation

12.2.2.1 IDT_L100 IDTechSDK.IDT_L100.SharedController [static],[get]

Singleton Instance

Establishes an singleton instance of IDT_L100 class.

Returns

Instance of IDT_L100

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

- Source/IDT_L100.cs

12.3 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)

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 msr_captureEncryptType
- 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
- EMV_ENCRYPTION_MODE emv_encryptionMode
- 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

12.3.1 Detailed Description

Class for swipe data

12.3.2 Member Function Documentation

12.3.2.1 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

isAES	TRUE = AES, FALSE = TDES
ksn	Key Serial Number
BDK	Base Derivitive Key

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.3.2.2 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

isAES	TRUE = AES, FALSE = TDES
ksn	Key Serial Number
BDK1	First Base Derivitive Key
BDK2	Second Base Derivitive Key

Returns

RETURN_CODE: Values can be parsed with errorCode.getErrorString()

12.3.3 Member Data Documentation

12.3.3.1 CTLS_APPLICATION IDTechSDK.IDTTransactionData.ctlsApplication

CTLS Application

12.3.3.2 String IDTechSDK.IDTTransactionData.device_RSN

Get the Reader Serial Number.

12.3.3.3 byte [ ] IDTechSDK.IDTTransactionData.emv_clearingRecord

clearing record TLV

12.3.3.4 byte [ ] IDTechSDK.IDTTransactionData.emv_encryptedTags

Encrypted Tags TLV

12.3.3.5 EMV_ENCRYPTION_MODE IDTechSDK.IDTTransactionData.emv_encryptionMode

EMV Encryption Mode

12.3.3.6 bool IDTechSDK.IDTTransactionData.emv_hasAdvise

Advise

12.3.3.7 bool IDTechSDK.IDTTransactionData.emv_hasReversal

Reversal

12.3.3.8 byte [ ] IDTechSDK.IDTTransactionData.emv_maskedTags

Masked Tags TLV

12.3.3.9 EMV_RESULT_CODE IDTechSDK.IDTTransactionData.emv_resultCode

EMV Result Code

12.3.3.10 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.

12.3.3.11 byte [ ] IDTechSDK.IDTTransactionData.emv_unencryptedTags

Unencrypted Tags TLV

12.3.3.12 EVENT_TRANSACTION_DATA_Types IDTechSDK.IDTTransactionData.Event

Transaction Data type,please see EVENT_TRANSACTION_DATA_Types for more information.

12.3.3.13 int IDTechSDK.IDTTransactionData.iccPresent

Get the swiped card ICC Status.
0 = Unknown 1 = True 2 = False

12.3.3.14 int IDTechSDK.IDTTransactionData.isCTLS

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

12.3.3.15 byte [ ] IDTechSDK.IDTTransactionData.mac

Message Authentication Code

12.3.3.16 byte [ ] IDTechSDK.IDTTransactionData.macKSN

Message Authentication Code Key Serial Number

12.3.3.17 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;

12.3.3.18 CAPTURE_ENCRYPT_TYPE IDTechSDK.IDTTransactionData.msr_captureEncryptType

Get the swiped card encrypted type,please see CAPTURE_ENCRYPT_TYPE for more information.
CAPTURE_ENCRYPT_TYPE_TDES:TDES;
CAPTURE_ENCRYPT_TYPE_AES:AES;

12.3.3.19 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

12.3.3.20 byte [ ] IDTechSDK.IDTTransactionData.msr_encTrack1

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

12.3.3.21 byte [ ] IDTechSDK.IDTTransactionData.msr_encTrack2

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

12.3.3.22 byte [ ] IDTechSDK.IDTTransactionData.msr_encTrack3

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

12.3.3.23 int IDTechSDK.IDTTransactionData.msr_errorCode

Contains error code when data is not returned

12.3.3.24 byte [ ] IDTechSDK.IDTTransactionData.msr_extendedField

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

12.3.3.25 byte [ ] IDTechSDK.IDTTransactionData.msr_hashTrack1

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

12.3.3.26 byte [ ] IDTechSDK.IDTTransactionData.msr_hashTrack2

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

12.3.3.27 byte [ ] IDTechSDK.IDTTransactionData.msr_hashTrack3

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

12.3.3.28 KEY_VARIANT_TYPE IDTechSDK.IDTTransactionData.msr_keyVariantType

KEY_VARIANT_TYPE_DATA = Data Variant key used
KEY_VARIANT_TYPE_PIN = PIN Variant key used

12.3.3.29 byte [ ] IDTechSDK.IDTTransactionData.msr_KSN

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

12.3.3.30 byte [ ] IDTechSDK.IDTTransactionData.msr_rawData

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

12.3.3.31 byte [ ] IDTechSDK.IDTTransactionData.msr_sessionID

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

12.3.3.32 String IDTechSDK.IDTTransactionData.msr_track1

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

12.3.3.33 int IDTechSDK.IDTTransactionData.msr_track1Length

Get the swiped card length of Track1 data.

12.3.3.34 String IDTechSDK.IDTTransactionData.msr_track2

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

12.3.3.35 int IDTechSDK.IDTTransactionData.msr_track2Length

Get the swiped card length of Track2 data.

12.3.3.36 String IDTechSDK.IDTTransactionData.msr_track3

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

12.3.3.37 int IDTechSDK.IDTTransactionData.msr_track3Length

Get the swiped card length of Track3 data.

12.3.3.38 EVENT_NOTIFICATION_Types IDTechSDK.IDTTransactionData.Notification

Event Notification type,please see EVENT_NOTIFICATION_Types for more information.

12.3.3.39 string IDTechSDK.IDTTransactionData.pin_KeyEntry

KSN for Pinblock

12.3.3.40 string IDTechSDK.IDTTransactionData.pin_KSN

KSN for Pinblock

12.3.3.41 string IDTechSDK.IDTTransactionData.pin_pinblock

PIN block from PINPAD

12.3.3.42 byte IDTechSDK.IDTTransactionData.SW1

SW1

12.3.3.43 byte IDTechSDK.IDTTransactionData.SW2

SW2

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

- Source/IDT_Transactions.cs

Index