Android SDK Guide

for NEO2

#80152505-001

Rev. A

Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;NEO2;SDK;Android	7/19/2017

Contents

Chapter 1

IDTech Android SDK Reference Guide for NEO2

Universal_SDK_X.XX.XXX.jar is an Android library that will be provided by IDTech as the main interface between Android applications, the NEO2 and payment processing solutions.

The purpose of this document is to describe the requirements of the library as well as the interface definitions and requirements needed for any Android applications wishing to deploy with the payment application.

Chapter 2

Connecting with NEO2

The NEO2 connects through Bluetooth Low Energy or USB on Android.

2.1 Connect with USB

The NEO2 will be recognized as a Human Interface Device once plugged into the Android USB port. The Android must be running firmware 3.1 or greater, and it will need an Android USB host adapter cable, usually referred to as OTG. Please see your manufacturers documentation for the correct part to use. Use a standard USB-miniUSB plug to attach the NEO2 (mini-USB port on left side) to the Android host adapter cable.

2.2 Connect with BLE

- Bluetooth Low Energy module is available on NEO2 (Android 5.1.1 or above is required). BLE scanning can be enabled on the Android device. Reference: Activate Bluetooth Low Energy

Chapter 3

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.

3.1 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.

3.2 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.

3.3 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.

3.4 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.

3.5 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 4

NEO2 Main Transaction Commands

The methods below are provided as a reference to the main commands needed to execute a contact or contactless EMV transaction, or collect MSR information from a swipe or tap.

4.1 Transaction Methods

Start Transaction

com.idtechproducts.device.IDT_NEO2.device_startTransaction()
com.idtechproducts.device.IDT_NEO2.emv_startTransaction()
com.idtechproducts.device.IDT_NEO2.ctls_startTransaction()
com.idtechproducts.device.IDT_NEO2.msr_startMSRSwipe()

- device = start a transaction on any interface
- emv = start a transaction contact interface ONLY
- ctls = start a transaction on contactless/MSR interfaces ONLY
- msr = start a transaction on contactless/MSR interfaces ONLY

com.idtechproducts.device.IDT_NEO2.emv_authenticateTransaction()

By default, auto-authenticate is ON and this step does not need to be performed. If auto-authenticate is OFF (com.idtechproducts.device.IDT_NEO2.emv_setAutoAuthenticateTransaction()), when the results come back as com.idtechproducts.device.IDTEMVData.START_TRANS_SUCCESS, this method must be called to continue the EMV transaction.

Complete Online EMV Transaction

com.idtechproducts.device.IDT_NEO2.emv_completeTransaction()

After receiving a host response, pass host response (minimum Authorization Response Code) through the methods parameter. EMV tags can be parsed returned pointer.

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

Terminal Configuration

com.idtechproducts.device.IDT_NEO2.emv_retrieveTerminalData()
com.idtechproducts.device.IDT_NEO2.emv_removeTerminalData()
com.idtechproducts.device.IDT_NEO2.emv_setTerminalData()

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

AID Management

com.idtechproducts.device.IDT_NEO2.emv_retrieveApplicationData:response()
com.idtechproducts.device.IDT_NEO2.emv_removeApplicationData()
com.idtechproducts.device.IDT_NEO2.emv_setApplicationData:configData()
com.idtechproducts.device.IDT_NEO2.emv_retrieveAidList()

Methods for AID management on Contact EMV. When setting the AID, you pass tags in TLV format. When retrieving AID, you can receive the results as tags in TLV format. When retrieving the AID list, the list of AID Names/length can be retrieved from the String[] response

CAPK Management

com.idtechproducts.device.IDT_NEO2.emv_retrieveCAPK)
com.idtechproducts.device.IDT_NEO2.emv_removeCAPK()
com.idtechproducts.device.IDT_NEO2.emv_setCAPK()
com.idtechproducts.device.IDT_NEO2.emv_retrieveCAPKList()

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

CRL Management

com.idtechproducts.device.IDT_NEO2.emv_retrieveCRL()
com.idtechproducts.device.IDT_NEO2.emv_removeCRL()
com.idtechproducts.device.IDT_NEO2.emv_setCRL()

Methods for Certificate Revocation List management.

APDU Communication

com.idtechproducts.device.IDT_NEO2.icc_passthroughOnICC()
com.idtechproducts.device.IDT_NEO2.icc_passthroughOffICC()
com.idtechproducts.device.IDT_NEO2.icc_powerOnICC:()
com.idtechproducts.device.IDT_NEO2.icc_powerOffICC:()

com.idtechproducts.device.IDT_NEO2.icc_exchangeAPDU:response:()
Allows the direct sending of APDU packets to ICC. Pass through mode must first be enabled. Then Power On needs to complete successfully. Then APDU packet exchange can take place

4.2 MSR/CTLS

Request Swipe or Tap

com.idtechproducts.device.IDT_NEO2.msr_startMSRSwipe()
com.idtechproducts.device.IDT_NEO2.ctls_startTransaction()
Both methods perform identical operation. Enables MSR to receive Swipe and CTLS to receive tap.

Cancel Swipe

com.idtechproducts.device.IDT_NEO2.msr_cancelMSRSwipe()
com.idtechproducts.device.IDT_NEO2.ctls_cancelTransaction()
Both methods perform identical operation. Cancels the Swipe/Tap request.

Chapter 5

EMV Callback

During an EMV transaction, without a built-in LCD display on the NEO2, LCD Display messages will be returned as an EMV Callback.

The callback for LCD messages is com.idtechproducts.device.OnReceiverListener.lcdDisplay

Once this listener is implemented, if an EMV transaction requires information to be displayed on what would normally be an LCD display controlled by the Kernel, this data is returned with the display message type, and message string(s).

To evaluate what kind of LCD message, you interpret the mode as follows:
1- LCD_DISPLAY_MODE_MENU: Menu selection, response required with selected menu index #, or 0 to cancel
2- LCD_DISPLAY_MODE_PROMPT: Message Prompt, response required 'E' for Enter/Accept, or 'C' for cancel
3- LCD_DISPLAY_MODE_MESSAGE: Display Message, no response required
8- LCD_DISPLAY_MODE_LANGUAGE_SELECT: Language selection, response required with selected language index #
16- LCD_DISPLAY_MODE_CLEAR_SCREEN: Request to clear LCD screen of information
If the mode is LCD_DISPLAY_MODE_MESSAGE or LCD_DISPLAY_MODE_CLEAR_SCREEN, these do not pause the EMV transaction. These two modes are for displaying a message (no response required), or for clearing the screen.

If the mode is LCD_DISPLAY_MODE_MENU, LCD_DISPLAY_MODE_PROMPT, or LCD_DISPLAY_MODE_LA↩ NGUAGE_SELECT, the provided message must be displayed, and then the EMV transaction pauses until a response is sent to IDT_NEO2::emv_callbackResponseLCD:selection:().

The message to display is returned as and array of strings (String[]). This contains either Message String, or a Message retrieved from the LCD Foreign Language Mapping Table (Foreign Language Mapping Table).

Chapter 6

Sending Direct Commands

The main purpose of Android library for NEO2 is to expedite integration to the device by providing the connectivity and communication protocols. It also provides the main functions to get device info, perform contact EMV Transactions, perform swipe/Tap transactions, and to modify contact EMV data files.

The NEO2 has an extensive and powerful command set based on the NEO platform. A NEO command consists of a Command, a Sub Command, and optionally data. To access these commands, please reference the NEO/IDG Command document included as a separate item in the SDK. Please note that Protocol 1 commands have been deprecated, and any existing Protocol 1 commands relevant to NEO2 can be accomplished by a Protocol 2 command. The Android .jar uses the following the command to send Protocol 2 commands to NEO2:

com.idtechproducts.device.IDT_NEO2.device_sendCommand()

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

Chapter 7

Core Implementation NEO2: Android

Universal_SDK_X.XX.XXX.jar includes class libraries to interface with the NEO2. This guide assume a fair understanding of Eclipse IDE and general Android programming knowledge.

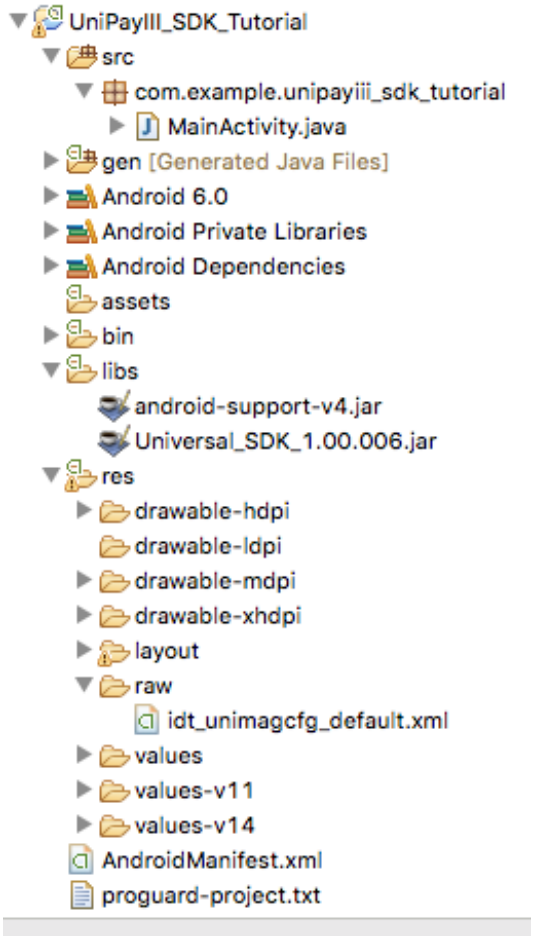
7.1 Integrating with Android SDK

- Import the necessary libraries
- Add Import statements to utilize libraries
- Implement OnReceiverListener for the activity
- Enable permissions for the application
- Allocate/initialize NEO2 objects
- Activate Bluetooth Low Energy (if available on device)
- Sample Project Tutorial Eclipse
- Sample Project Tutorial Android Studio

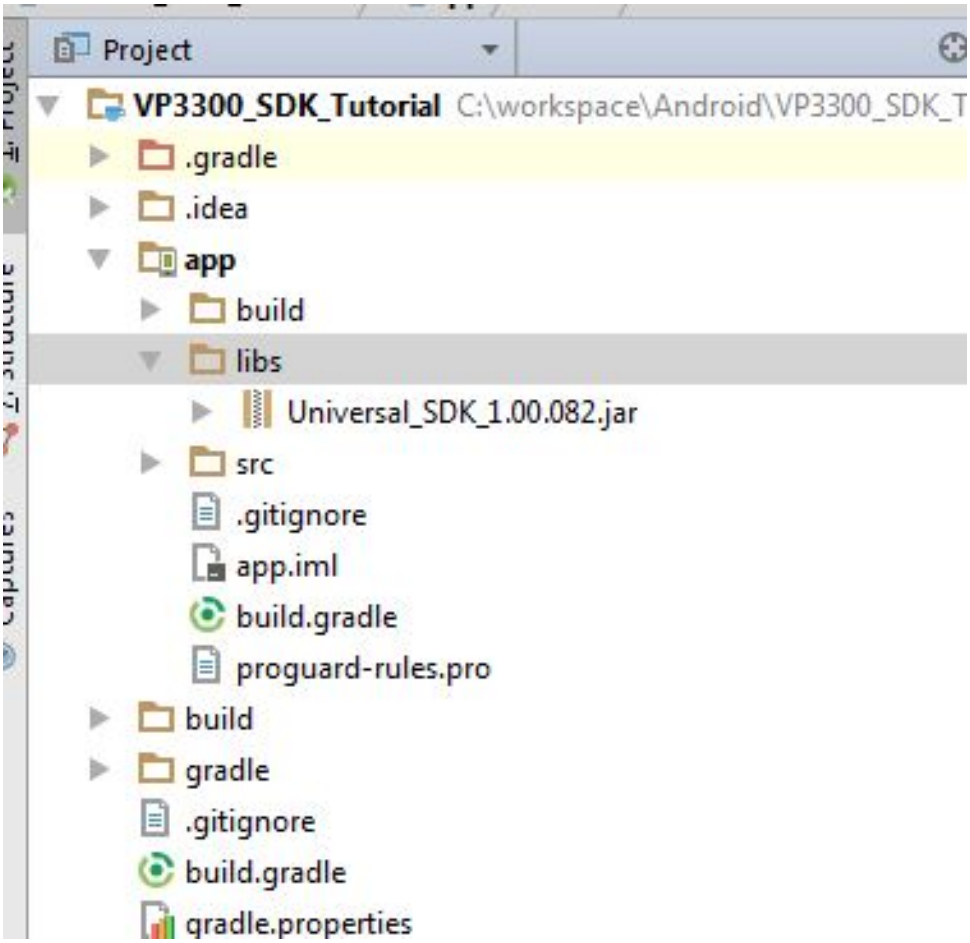
7.2 Import the necessary libraries

Communicating with NEO2 requires the Universal_SDK_X.XX.XXX.jar file be added to the project. While build paths can be created pointing to the Universal_SDK_X.XX.XXX.jar file, the simplest solution is to add the jar file to the projects libs folder.

Eclispe:



Android Studio:



7.3 Add Import statements to utilize libraries

In the header files of the java activity that will access NEO2, use import statement utilize the library:

```
import com.idtechproducts.device.*;
```

```
1  package com.example.unipayiii_sdk_tutorial;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import com.idtechproducts.device.*;
6
```

The complete import list is as follows:

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.os.Bundle;
```

```
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.idtechproducts.device.*;
import com.idtechproducts.device.bluetooth.BluetoothLEController;
```

7.4 Implement OnReceiverListener for the activity:

In the class that will be a delegate of NEO2, implement OnReceiverListener. Add the implemented methods to eliminate any error messages.

```
public class MainActivity extends Activity implements  OnReceiverListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public void ICCNotifyInfo(byte[] arg0, String arg1) {
        // TODO Auto-generated method stub

    }

    @Override
    public void LoadXMLConfigFailureInfo(int arg0, String arg1) {
        // TODO Auto-generated method stub

    }

    @Override
    public void autoConfigCompleted(StructConfigParameters arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void autoConfigProgress(int arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void deviceConnected() {
        // TODO Auto-generated method stub

    }

    @Override
    public void deviceDisconnected() {
        // TODO Auto-generated method stub

    }

    @Override
    public void emvTransactionData(IDTEMVData arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void lcdDisplay(int arg0, String[] arg1, int arg2) {
        // TODO Auto-generated method stub

    }

    @Override
    public void msgAudioVolumeAjustFailed() {
        // TODO Auto-generated method stub

    }

    @Override
    public void msgRKICompleted(String arg0) {
        // TODO Auto-generated method stub

    }
```

```
    @Override
    public void msgToConnectDevice() {
        // TODO Auto-generated method stub

    }

    @Override
    public void swipeMSRData(IDTMSRData arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void timeout(int arg0) {
        // TODO Auto-generated method stub

    }
}
```

7.5 Enable permissions for the application:

```
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-feature android:name="android.hardware.usb.host" />
```

7.6 Allocate/initialize NEO2 objects:

Initialize IDT_Device object by passing context and OnReceiverListener delegate.

```
// declaring the instance of the NEO2Reader;
private IDT_NEO2 myNEO2Reader = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    if(myNEO2Reader!=null){
        myNEO2Reader.unregisterListen();
        myNEO2Reader.release();
        myNEO2Reader = null;
    }
    myNEO2Reader = new IDT_NEO2(this,this,getActivity());
    myNEO2Reader.device_setDeviceType(DEVICE_TYPE.DEVICE_NEO2_BT);
    myNEO2Reader.registerListen();
    BLE_Id = "BLE_NAME";  //Change this to the BLE device name to match during a search
    scanLeDevice(true, BLE_ScanTimeout);
}
```

7.7 Activate Bluetooth Low Energy (if available on device):

If the desired connection is BLE (Bluetooth Low Energy), a minimum SDK level of 18 is required. All communication is handled by the BluetoothLEController.

```
import com.idtechproducts.device.bluetooth.BluetoothLEController;
import android.bluetooth.BluetoothAdapter;
```

Define a BluetoothAdapter for the project

```
private BluetoothAdapter mBtAdapter = null;
```

Define the routine that will start scanning for BLE devices, and stop after a timeout value is reached. You need to pass a callback to the BluetoothAdapter that will execute when a BLE device is found.

```
private void scanLeDevice(final boolean enable, long timeout) {

    if (enable) {
        handler.postDelayed(new Runnable() {
            public void run() {
                mBtAdapter.stopLeScan(mLeScanCallback);
            }
        }, timeout);
        mBtAdapter.startLeScan(mLeScanCallback);
    } else {
        mBtAdapter.stopLeScan(mLeScanCallback);
    }
}
```

Define the callback. When a device is found, a match can be determined by the device name or the device MAC address. If found, execute setBluetoothDevice on that device.

```
private String BLE_Id = "BLE_NAME";  //Chang this to the BLE device name to match during a search
private boolean btleDeviceRegistered = false;
private String btleDeviceAddress = "";
private final long BLE_ScanTimeout = 5000;  //in milliseconds

private BluetoothAdapter.LeScanCallback mLeScanCallback =
        new BluetoothAdapter.LeScanCallback() {
            public void onLeScan(final BluetoothDevice btledevice, int rssi,
                                 byte[] scanRecord) {
                if (BLE_Id != null)
                {
                    if (BLE_Id.length() == 17 && BLE_Id.charAt(2) == ':' && BLE_Id.charAt(5) == ':' &&
      BLE_Id.charAt(8) == ':' &&
                        BLE_Id.charAt(11) == ':' && BLE_Id.charAt(14) == ':')  //search by address
                    {
                        String deviceAddress = btledevice.getAddress();
                        if (deviceAddress != null && deviceAddress.equalsIgnoreCase(BLE_Id))  //found the
       device by address
                        {
                            BluetoothLEController.setBluetoothDevice(btledevice);
                            btleDeviceAddress = deviceAddress;
                            if (!btleDeviceRegistered)
                            {
                                myNEO2Reader.registerListen();
                                btleDeviceRegistered = true;
                            }
                        }
                    }
                    else  //search by name
                    {
                        String deviceName = btledevice.getName();
                        if (deviceName != null && deviceName.startsWith(BLE_Id))  //found the device by
       name
                        {
                            BluetoothLEController.setBluetoothDevice(btledevice);
                            btleDeviceAddress = btledevice.getAddress();
                            if (!btleDeviceRegistered)
                            {
                                myNEO2Reader.registerListen();
                                btleDeviceRegistered = true;
                            }
                        }
                    }
                }
            }
        };
```

7.8 Sample Project Tutorial Eclipse

Using Eclipse, we will create a sample project that will interface with the NEO2 and will perform the following activities:

- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for CTLS/MSR (tap/swipe)
- Start/Cancel EMV Transaction

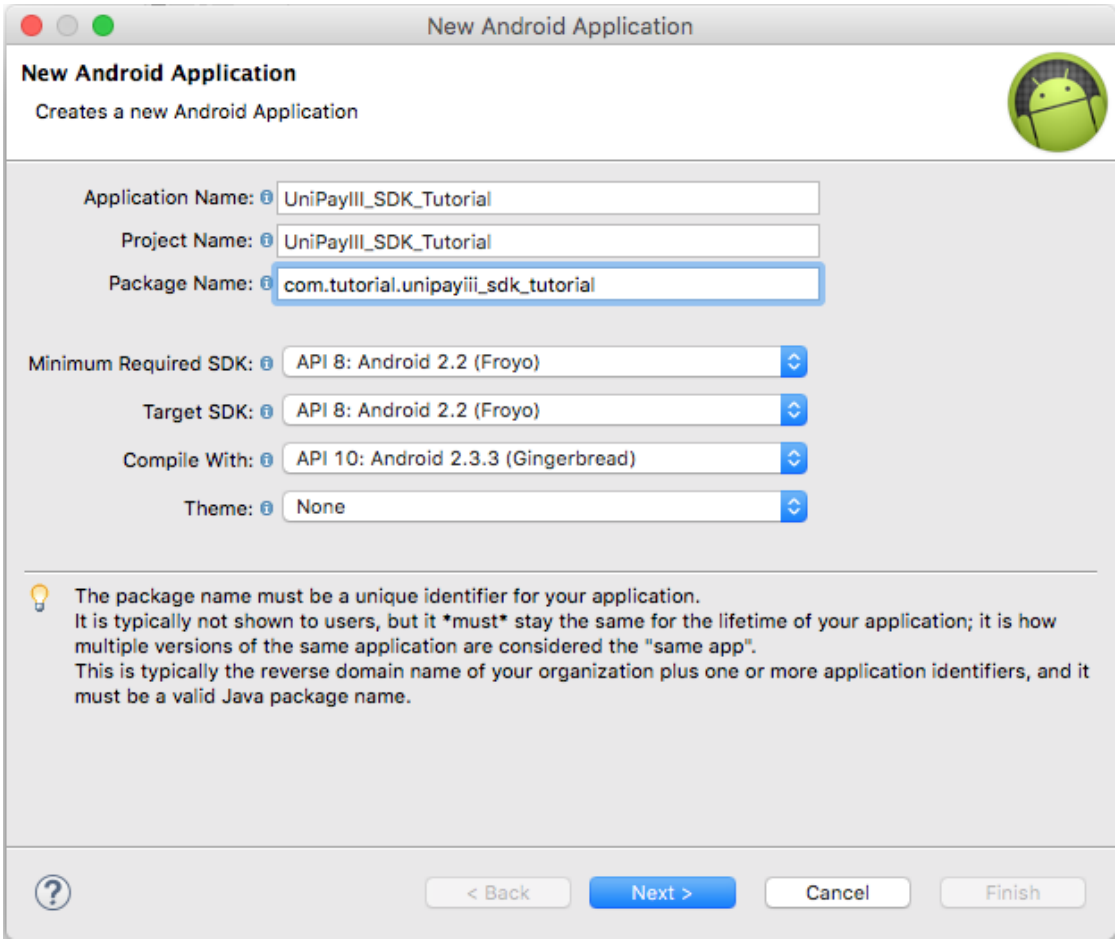
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

Listeners:

- Listener to receive card swipes
- Listener to detect device connected
- Listener to detect device disconnected
- Listener to receive EMV/CTLS tag data
- Listener to receive LCD messages

7.8.1 Step 1: Create New Project

Create a new Android Application project in Eclipse as an Empty Activity



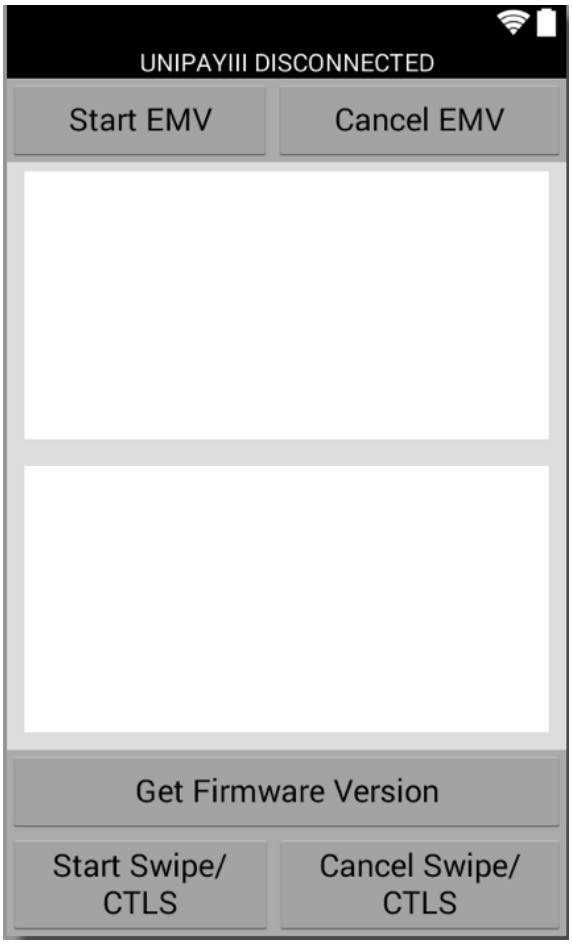
7.8.2 Step 2: Import IDTechSDK for NEO2

Import the necessary libraries

7.8.3 Step 3: Design Interface

Design the User Interface by editing the main layout XML file
Open your layout and add items to so it contains the following buttons/fields (sample code provide at end of section):

- Add a TextView to the top that will signify connection/disconnection status.
- Add two TextViews to communicate data from the NEO2 and for EMV LCD display information. Remove the Editable behavior if you don't want the keyboard to pop up if you accidentally select it.
- Add buttons to execute the following functions:
  - Get Firmware
  - Start MSR/ CTLS
  - Start ICC EMV
  - Complete ICC EMV
  - Cancel Transaction



7.8.4 Step 4: Configure Activity File

In the activity file, perform the following:

- Add Import statements to utilize libraries
- Implement OnReceiverListener for the activity

- Enable permissions for the application
- Define the association for all the elements on the layout: The connection label, the two text views, and the 5 buttons

Layout Source Code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#aaaaaa"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/status_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#000000"
        android:gravity="center_vertical|center_horizontal"
        android:text="NEO2 DISCONNECTED"
        android:textColor="#FFFFFF" />
    <LinearLayout
        android:id="@+id/linearLayoutBottom2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <Button
            android:id="@+id/btn_StartEMV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:gravity="center_vertical|center_horizontal"
            android:text="Start EMV" />
        <Button
            android:id="@+id/btn_CancelEMV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Cancel EMV" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayoutEditText"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="#dddddd"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical" >
        <ScrollView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_marginBottom="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginTop="5dp"
            android:background="#ffffff" >
            <TextView
                android:id="@+id/lcdLog"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:text=""
                android:textColor="#000000"
                android:textSize="12sp"
                android:typeface="monospace" >
            </TextView>
        </ScrollView>
    </LinearLayout>
        <LinearLayout
        android:id="@+id/linearLayoutEditText2"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="#dddddd"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical" >
        <ScrollView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_marginBottom="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginTop="5dp"
```

```
            android:background="#ffffff" >
            <TextView
                android:id="@+id/textLog"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:text=""
                android:textColor="#000000"
                android:textSize="12sp"
                android:typeface="monospace" >
            </TextView>
        </ScrollView>
    </LinearLayout>
        <LinearLayout
        android:id="@+id/linearLayoutBottom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <Button
            android:id="@+id/btn_getFirmware"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:gravity="center_vertical|center_horizontal"
            android:text="Get Firmware Version" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayoutBottom4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >

        <Button
            android:id="@+id/btn_startSwipe"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Start Swipe/CTLS" />
        <Button
            android:id="@+id/btn_cancelSwipe"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Cancel Swipe/CTLS" />
    </LinearLayout>
</LinearLayout>
```

7.8.5 Step 5: Configure Method File

In the activity file, perform the following:

- set delegate and initialize NEO2 object in the onCreate method. Reference: Allocate/initialize NEO2 objects
- set correct device type.

```
// declaring the instance of the NEO2Reader;
private IDT_NEO2 myNEO2Reader = null;
private TextView connectStatusTextView;
private TextView textLog;
private TextView lcdLog;
private Button btnGetFirmware;
private Button btnStartSwipe;
private Button btnCancelSwipe;
private Button btnStartEMV;
private Button btnCancelEMV;
private Handler handler = new Handler();
private boolean isReaderConnected = false;
private String info = "";
private String detail = "";
private BluetoothAdapter mBtAdapter = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    handler = new Handler();
    btnStartEMV = (Button)findViewById(R.id.btn_StartEMV);
    btnCancelEMV = (Button)findViewById(R.id.btn_CancelEMV);
    btnGetFirmware = (Button)findViewById(R.id.btn_getFirmware);
    btnStartSwipe = (Button)findViewById(R.id.btn_startSwipe);
    btnCancelSwipe = (Button)findViewById(R.id.btn_cancelSwipe);
    textLog = (TextView)findViewById(R.id.textLog);
    lcdLog = (TextView)findViewById(R.id.lcdLog);
```

```
    connectStatusTextView = (TextView)findViewById(R.id.status_text);
    if(myNEO2Reader!=null){
        myNEO2Reader.unregisterListen();
        myNEO2Reader.release();
        myNEO2Reader = null;
    }
    myNEO2Reader = new IDT_NEO2(this,this,getActivity());
    myNEO2Reader.device_setDeviceType(DEVICE_TYPE.DEVICE_NEO2_USB);
    myNEO2Reader.registerListen();
}
```

- If available on device, activate BLE scanning. Reference: Activate Bluetooth Low Energy
  - Implement protocol delegate com.idtechproducts.device.OnReceiverListener.deviceConnected() and com.idtechproducts.device.OnReceiverListener.deviceDisconnected() to monitor connect/disconnect events and modify our connection label upon change. Reference: Implement OnReceiverListener for the activity
    Note: This notification may come back on a thread different that the UI thread, so we want to make sure to use a handler to send to main UI thread.

```
private Runnable doUpdateLabel = new Runnable()
{
    public void run()
    {
        if(!isReaderConnected){
            connectStatusTextView.setText("NEO2 DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("NEO2 CONNECTED");
        }
    }
};
@Override
public void deviceConnected() {
    isReaderConnected = true;
    handler.post(doUpdateLabel);
}

@Override
public void deviceDisconnected() {
    isReaderConnected = false;
    handler.post(doUpdateLabel);
}
```

-Implement protocol delegate com.idtechproducts.device.OnReceiverListener.swipeMSRData() to receive unsolicited card swipe data.

```
private Runnable doUpdateStatus = new Runnable()
{
    public void run()
    {
        lcdLog.setText(info);
        textLog.setText(detail);
    }
};
@Override
public void swipeMSRData(IDTMSRData card) {
    if (card.cardData[0] != (byte)0x01 && card.track1Length == 0 && card.track2Length == 0 && card.
      track3Length == 0)
        info = "Swipe/Tap data didn't read correctly";
    else
        info = "Swipe/Tap Read Successfully";
    detail = Common.parse_MSRData(myNEO2Reader.device_getDeviceType(), card);
    handler.post(doUpdateStatus);

}
```

- Implement protocol delegate com.idtechproducts.device.OnReceiverListener.emvTransactionData() to report EMV transaction results

```
    public void emvTransactionData(IDTEMVData emvData) {

        detail += Common.emvErrorCodes(emvData.result);
        detail += "\r\n";
        if (emvData.result == IDTEMVData.START_TRANS_SUCCESS)
            detail += "Start transaction response:\r\n";
        else if (emvData.result == IDTEMVData.GO_ONLINE)
```

```
            detail += "\r\nAuthentication response:\r\n";
        else
            detail += "\r\nComplete Transaction response:\r\n";
        if (!emvData.unencryptedTags.isEmpty())
        {
            detail += "Unencrypted Tags:\r\n";
            Set<String> keys = emvData.unencryptedTags.keySet();
            for(String key: keys){
                detail += key + ": ";
                byte[] data = emvData.unencryptedTags.get(key);
                detail += Common.getHexStringFromBytes(data) + "\r\n";
            }
        }
        if (!emvData.maskedTags.isEmpty())
        {
            detail += "Masked Tags:\r\n";
            Set<String> keys = emvData.maskedTags.keySet();
            for(String key: keys){
                detail += key + ": ";
                byte[] data = emvData.maskedTags.get(key);
                detail += Common.getHexStringFromBytes(data) + "\r\n";
            }
        }
        if (!emvData.encryptedTags.isEmpty())
        {
            detail += "Encrypted Tags:\r\n";
            Set<String> keys = emvData.encryptedTags.keySet();
            for(String key: keys){
                detail += key + ": ";
                byte[] data = emvData.encryptedTags.get(key);
                detail += Common.getHexStringFromBytes(data) + "\r\n";
            }
        }
        handler.post(doUpdateStatus);
         if (emvData.result == IDTEMVData.GO_ONLINE){
            //Auto Complete
             byte[] response = new byte[]{0x30,0x30};
             myNEO2Reader.emv_completeTransaction(false, response, null, null,null);
         }
         else if (emvData.result == IDTEMVData.START_TRANS_SUCCESS){
                //Auto Authenticate
             myNEO2Reader.emv_authenticateTransaction(null);
            }

    }
```

- Implement protocol delegate com.idtechproducts.device.OnReceiverListener.emvTransactionData() to receive LCD messages, and automatically select 1st menu item/language when presented with choices. Normal operation would require a choice be made by card holder.

```
public void lcdDisplay(int mode, String[] lines, int timeout) {

            if (mode == 0x01) //Menu Display
            {
                //automatically select 1st application
                myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else if (mode == 0x08) //Language Menu Display
            {
                //automatically select first language
                myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else{
                ResDataStruct toData = new ResDataStruct();
                info = lines[0];
                handler.post(doUpdateStatus);
            }
}
```

- Implement the button press methods

```
btnGetFirmware.setOnClickListener(new Button.OnClickListener(){
    public void onClick(View v) {
        info = "Getting Firmware\n";
        detail = "";
        handler.post(doUpdateStatus);
        StringBuilder sb = new StringBuilder();
        int ret = myNEO2Reader.device_getFirmwareVersion(sb);
        if (ret == ErrorCode.SUCCESS) {
            info += "Firmware Version: " + sb.toString();
            detail = "";
            handler.post(doUpdateStatus);
        }
```

```
            else {
                info += "GetFirmwareVersion: Failed\n";
                info += "Status: "+ myNEO2Reader.device_getResponseCodeString(ret)+"";
                detail = "";
                handler.post(doUpdateStatus);
                }
        }
    });

    btnStartEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {;
        detail = "";
        info = "Starting EMV Transaction\n";
        handler.post(doUpdateStatus);
            IDT_NEO2.emv_allowFallback(true);
            myNEO2Reader.emv_startTransaction(1.00, 0.00, 0, 30, null, false);
        }
    });

    btnCancelEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Canceling EMV Transaction\n";
            handler.post(doUpdateStatus);
            ResDataStruct resData = new ResDataStruct();
            myNEO2Reader.emv_cancelEMVTransaction(resData);
        }
    });

    btnStartSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Starting Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_startMSRSwipe();
        }
    });

    btnCancelSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Cancelling Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_cancelMSRSwipe();
        }
    });
```

7.8.6 Complete code listing

```
package com.example.neo2_sdk_tutorial;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.Set;

import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

import com.idtechproducts.device.*;
import com.idtechproducts.device.ReaderInfo.DEVICE_TYPE;
import com.idtechproducts.device.ReaderInfo.
      CAPTURE_ENCODE_TYPE;
import com.idtechproducts.device.ReaderInfo.
      CAPTURE_ENCRYPT_TYPE;
import com.idtechproducts.device.ReaderInfo.
      EVENT_MSR_Types;

public class MainActivity extends Activity implements  OnReceiverListener{

    // declaring the instance of the NEO2Reader;
    private IDT_NEO2 myNEO2Reader = null;
    private TextView connectStatusTextView;
    private TextView textLog;
```

```
private TextView lcdLog;
private Button btnGetFirmware;
private Button btnStartSwipe;
private Button btnCancelSwipe;
private Button btnStartEMV;
private Button btnCancelEMV;
private Handler handler = new Handler();
private boolean isReaderConnected = false;
private String info = "";
private String detail = "";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    handler = new Handler();
    btnStartEMV = (Button)findViewById(R.id.btn_StartEMV);
    btnCancelEMV = (Button)findViewById(R.id.btn_CancelEMV);
    btnGetFirmware = (Button)findViewById(R.id.btn_getFirmware);
    btnStartSwipe = (Button)findViewById(R.id.btn_startSwipe);
    btnCancelSwipe = (Button)findViewById(R.id.btn_cancelSwipe);
    textLog = (TextView)findViewById(R.id.textLog);
    lcdLog = (TextView)findViewById(R.id.lcdLog);
    connectStatusTextView = (TextView)findViewById(R.id.status_text);
    if(myNEO2Reader!=null){
        myNEO2Reader.unregisterListen();
        myNEO2Reader.release();
        myNEO2Reader = null;
    }
    myNEO2Reader = new IDT_NEO2(this,this);
    myNEO2Reader.device_setDeviceType(DEVICE_TYPE.DEVICE_NEO2_USB);
    myNEO2Reader.registerListen();
    loadXMLfile();


    btnGetFirmware.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            info = "Getting Firmware\n";
            detail = "";
            handler.post(doUpdateStatus);
            StringBuilder sb = new StringBuilder();
            int ret = myNEO2Reader.device_getFirmwareVersion(sb);
            if (ret == ErrorCode.SUCCESS) {
                info += "Firmware Version: " + sb.toString();
                detail = "";
                handler.post(doUpdateStatus);
            }
            else {
                info += "GetFirmwareVersion: Failed\n";
                info += "Status: "+ myNEO2Reader.device_getResponseCodeString(ret)+"";
                detail = "";
                handler.post(doUpdateStatus);
                }
        }
    });

    btnStartEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {;
        detail = "";
        info = "Starting EMV Transaction\n";
        handler.post(doUpdateStatus);
            IDT_NEO2.emv_allowFallback(true);
            myNEO2Reader.emv_startTransaction(1.00, 0.00, 0, 30, null, false);
        }
    });

    btnCancelEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Canceling EMV Transaction\n";
            handler.post(doUpdateStatus);
            ResDataStruct resData = new ResDataStruct();
            myNEO2Reader.emv_cancelEMVTransaction(resData);
        }
    });

    btnStartSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Starting Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_startMSRSwipe();
        }
    });

    btnCancelSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
```

```
            info = "Cancelling Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_cancelMSRSwipe();
        }
    });

}

@Override
public void ICCNotifyInfo(byte[] arg0, String arg1) {
    // TODO Auto-generated method stub

}

@Override
public void LoadXMLConfigFailureInfo(int arg0, String arg1) {
    // TODO Auto-generated method stub

}

@Override
public void autoConfigCompleted(StructConfigParameters arg0) {
    // TODO Auto-generated method stub

}

@Override
public void autoConfigProgress(int arg0) {
    // TODO Auto-generated method stub

}

private Runnable doUpdateLabel = new Runnable()
{
    public void run()
    {
        if(!isReaderConnected){
            connectStatusTextView.setText("NEO2 DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("NEO2 CONNECTED");
        }
    }
};
@Override
public void deviceConnected() {
    isReaderConnected = true;
    handler.post(doUpdateLabel);
}

@Override
public void deviceDisconnected() {
    isReaderConnected = false;
    handler.post(doUpdateLabel);
}

private void printTags(IDTEMVData emvData)
{

}

@Override
public void emvTransactionData(IDTEMVData emvData) {

    detail += Common.emvErrorCodes(emvData.result);
    detail += "\r\n";
    if (emvData.result == IDTEMVData.START_TRANS_SUCCESS)
        detail += "Start transaction response:\r\n";
    else if (emvData.result == IDTEMVData.GO_ONLINE)
        detail += "\r\nAuthentication response:\r\n";
    else
        detail += "\r\nComplete Transaction response:\r\n";
    if (!emvData.unencryptedTags.isEmpty())
    {
        detail += "Unencrypted Tags:\r\n";
        Set<String> keys = emvData.unencryptedTags.keySet();
        for(String key: keys){
            detail += key + ": ";
            byte[] data = emvData.unencryptedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    if (!emvData.maskedTags.isEmpty())
    {
        detail += "Masked Tags:\r\n";
        Set<String> keys = emvData.maskedTags.keySet();
        for(String key: keys){
```

```
            detail += key + ": ";
            byte[] data = emvData.maskedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    if (!emvData.encryptedTags.isEmpty())
    {
        detail += "Encrypted Tags:\r\n";
        Set<String> keys = emvData.encryptedTags.keySet();
        for(String key: keys){
            detail += key + ": ";
            byte[] data = emvData.encryptedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    handler.post(doUpdateStatus);
     if (emvData.result == IDTEMVData.GO_ONLINE){
        //Auto Complete
         byte[] response = new byte[]{0x30,0x30};
         myNEO2Reader.emv_completeTransaction(false, response, null, null,null);
     }
     else if (emvData.result == IDTEMVData.START_TRANS_SUCCESS){
            //Auto Authenticate
         myNEO2Reader.emv_authenticateTransaction(null);
        }

}

public void lcdDisplay(int mode, String[] lines, int timeout) {

            if (mode == 0x01) //Menu Display
            {
                //automatically select 1st application
                myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else if (mode == 0x08) //Language Menu Display
            {
                //automatically select first language
                myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else{
                ResDataStruct toData = new ResDataStruct();
                info = lines[0];
                handler.post(doUpdateStatus);
            }
}


@Override
public void msgAudioVolumeAjustFailed() {
    // TODO Auto-generated method stub

}

@Override
public void msgRKICompleted(String arg0) {
    // TODO Auto-generated method stub

}

@Override
public void msgToConnectDevice() {
    // TODO Auto-generated method stub

}

private Runnable doUpdateStatus = new Runnable()
{
    public void run()
    {
        lcdLog.setText(info);
        textLog.setText(detail);
    }
};
@Override
public void swipeMSRData(IDTMSRData card) {
    if (card.cardData[0] != (byte)0x01 && card.track1Length == 0 && card.track2Length == 0 && card.
  track3Length == 0)
        info = "Swipe/Tap data didn't read correctly";
    else
        info = "Swipe/Tap Read Successfully";
    detail = Common.parse_MSRData(myNEO2Reader.device_getDeviceType(), card);
    handler.post(doUpdateStatus);

}

@Override
```

```
    public void timeout(int arg0) {
        // TODO Auto-generated method stub

    }

    private String getXMLFileFromRaw(String fileName ,int res){
        //the target filename in the application path
        String fileNameWithPath = null;
        fileNameWithPath = fileName;
        String newFilename = fileName;

        try {
            InputStream in = getResources().openRawResource(res);
            int length = in.available();
            byte [] buffer = new byte[length];
            in.read(buffer);
            in.close();
            deleteFile(fileNameWithPath);
            FileOutputStream fout = openFileOutput(fileNameWithPath, MODE_PRIVATE);
            fout.write(buffer);
            fout.close();

            // to refer to the application path
            File fileDir = this.getFilesDir();
            fileNameWithPath = fileDir.getParent() + java.io.File.separator + fileDir.getName();
            fileNameWithPath += java.io.File.separator+newFilename;

        } catch(Exception e){
            e.printStackTrace();
            fileNameWithPath = null;
        }
        return fileNameWithPath;
    }

    private String getConfigurationFileFromRaw( ){
        return getXMLFileFromRaw("idt_unimagcfg_default.xml",R.raw.idt_unimagcfg_default);
        }
    private boolean isFileExist(String path) {
        if(path==null)
            return false;
        File file = new File(path);
        if (!file.exists()) {
          return false ;
        }
        return true;
    }
    private void loadXMLfile(){

        //load the XML configuration file
        String fileNameWithPath = getConfigurationFileFromRaw();
        if(!isFileExist(fileNameWithPath)) {
            fileNameWithPath = null;
        }
        // Network operation is prohibited in the UI Thread if target API is 11 or above.
        // If target API is 11 or above, please use AsyncTask to avoid errors.
        myVP3600Reader.config_setXMLFileNameWithPath(fileNameWithPath);
        Log.d("Demo Info >>>>>","loadingConfigurationXMLFile begin.");
        myNEO2Reader.config_loadingConfigurationXMLFile(true);
    }




}
```

7.9 Sample Project Tutorial Android Studio

Using Android Studio, we will create a sample project that will interface with the NEO2 and will perform the following activities:

- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for CTLS/MSR (tap/swipe)
- Start/Cancel EMV Transaction

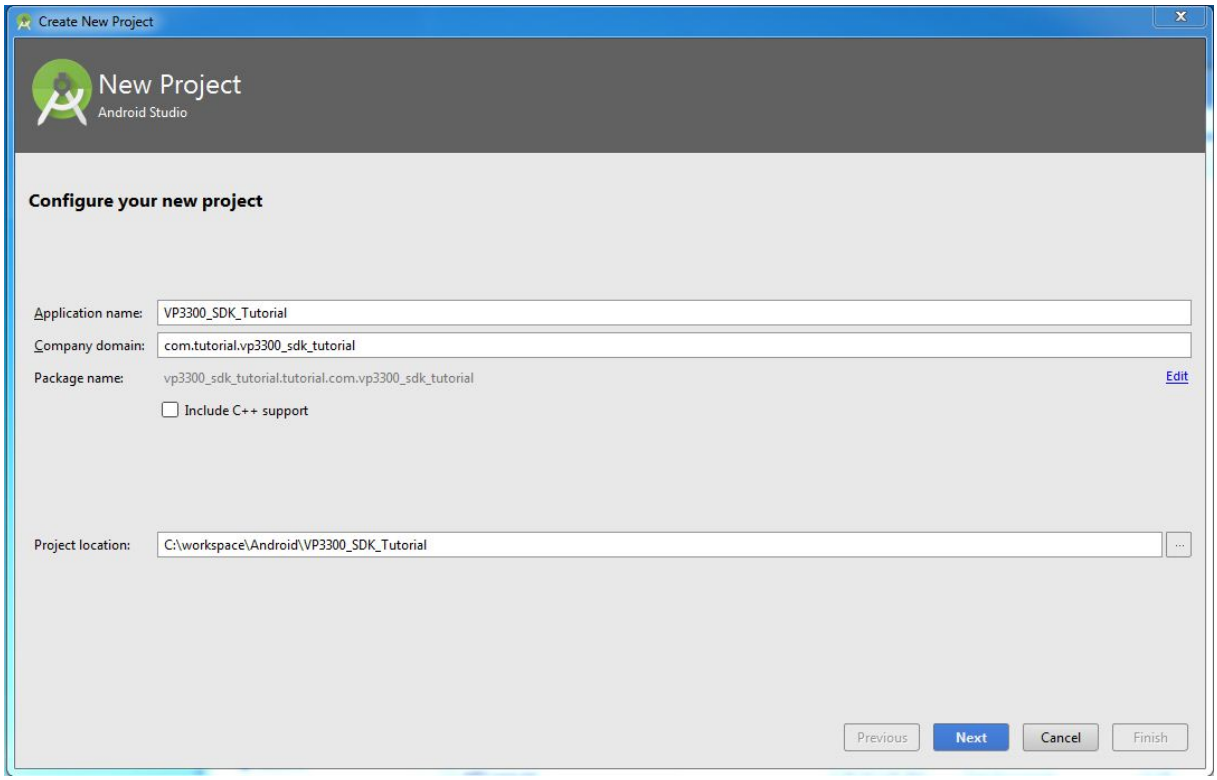
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

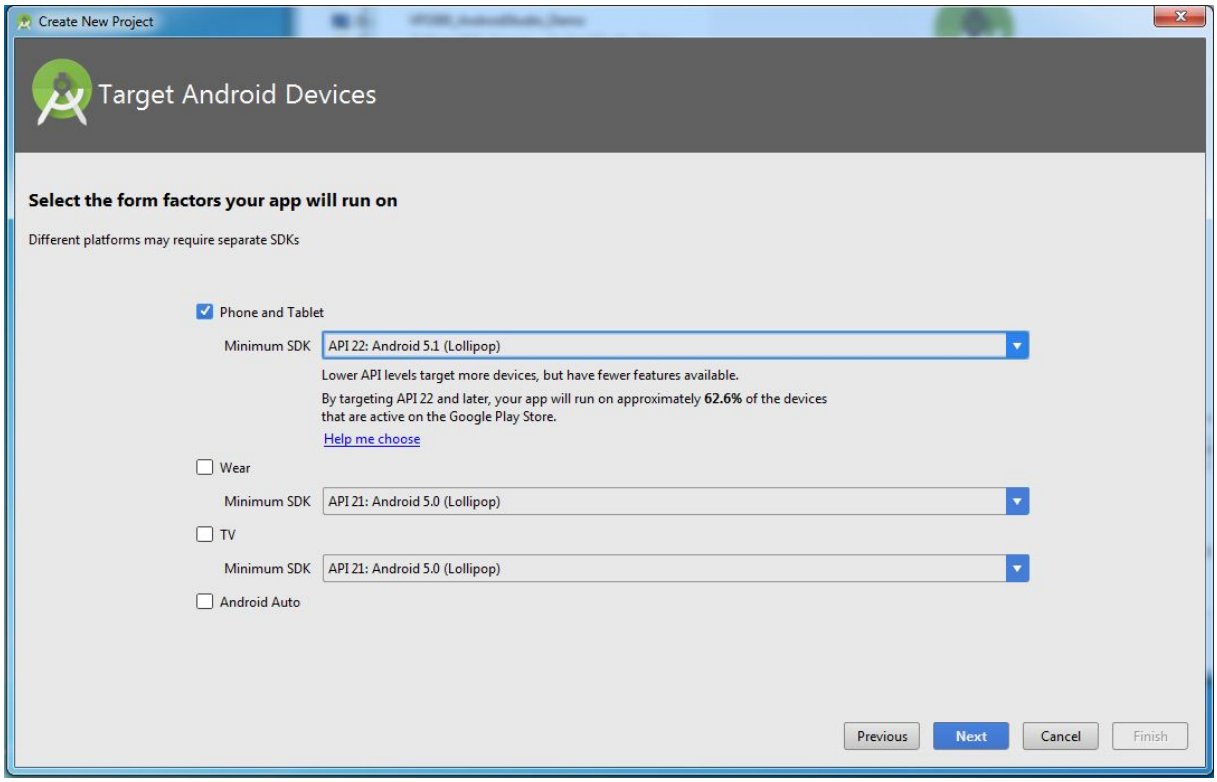
Listeners:

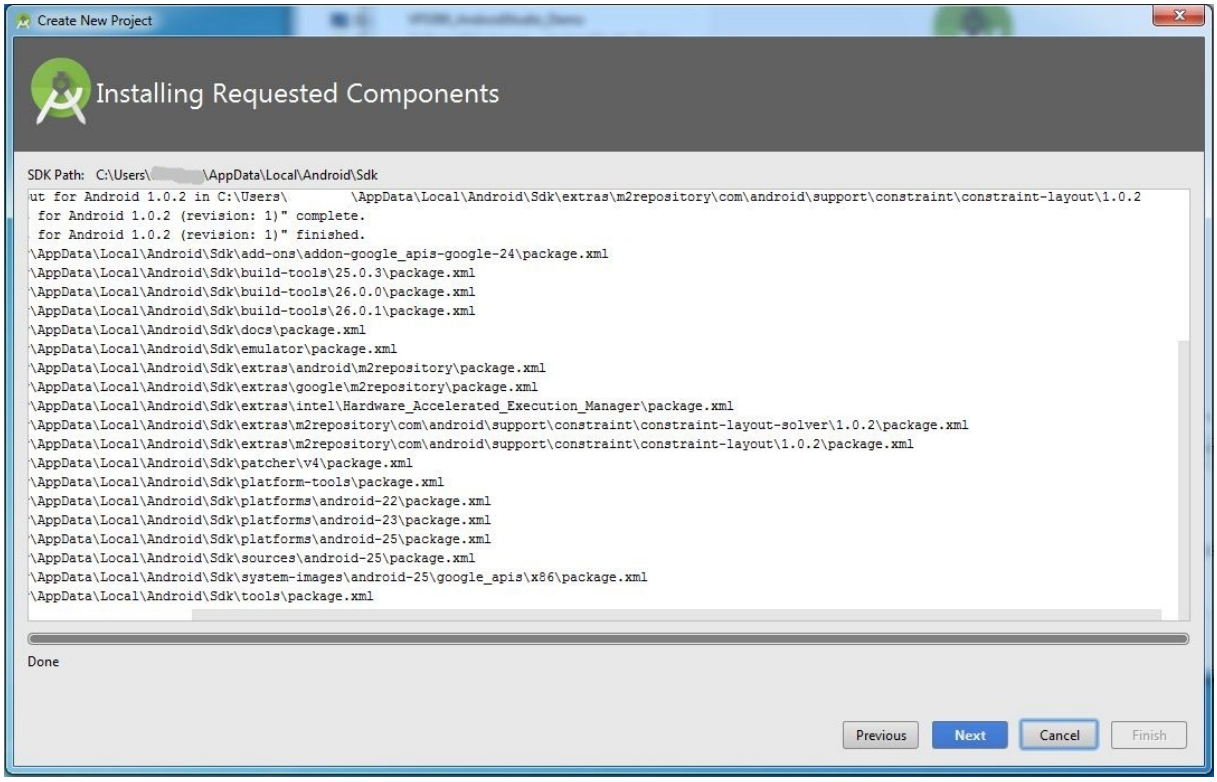
- Listener to receive card swipes
- Listener to detect device connected
- Listener to detect device disconnected
- Listener to receive EMV/CTLS tag data
- Listener to receive LCD messages

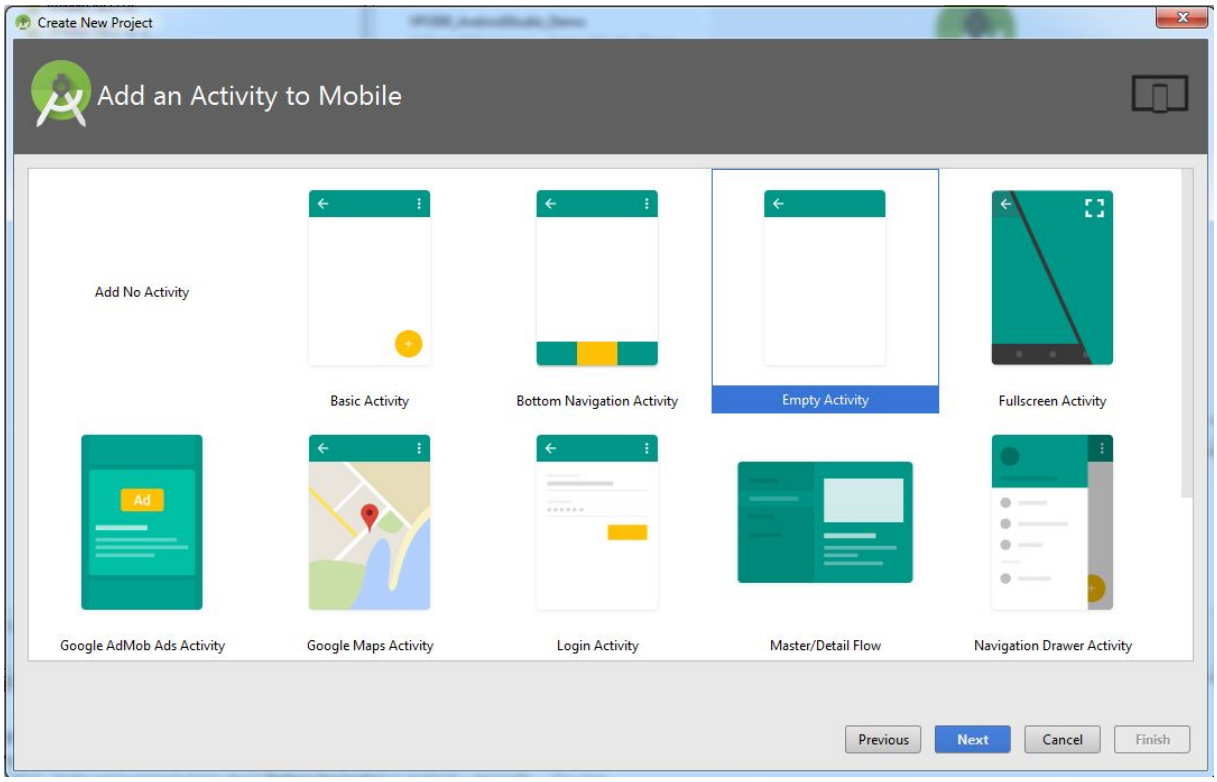
7.9.1 Step 1: Create New Project

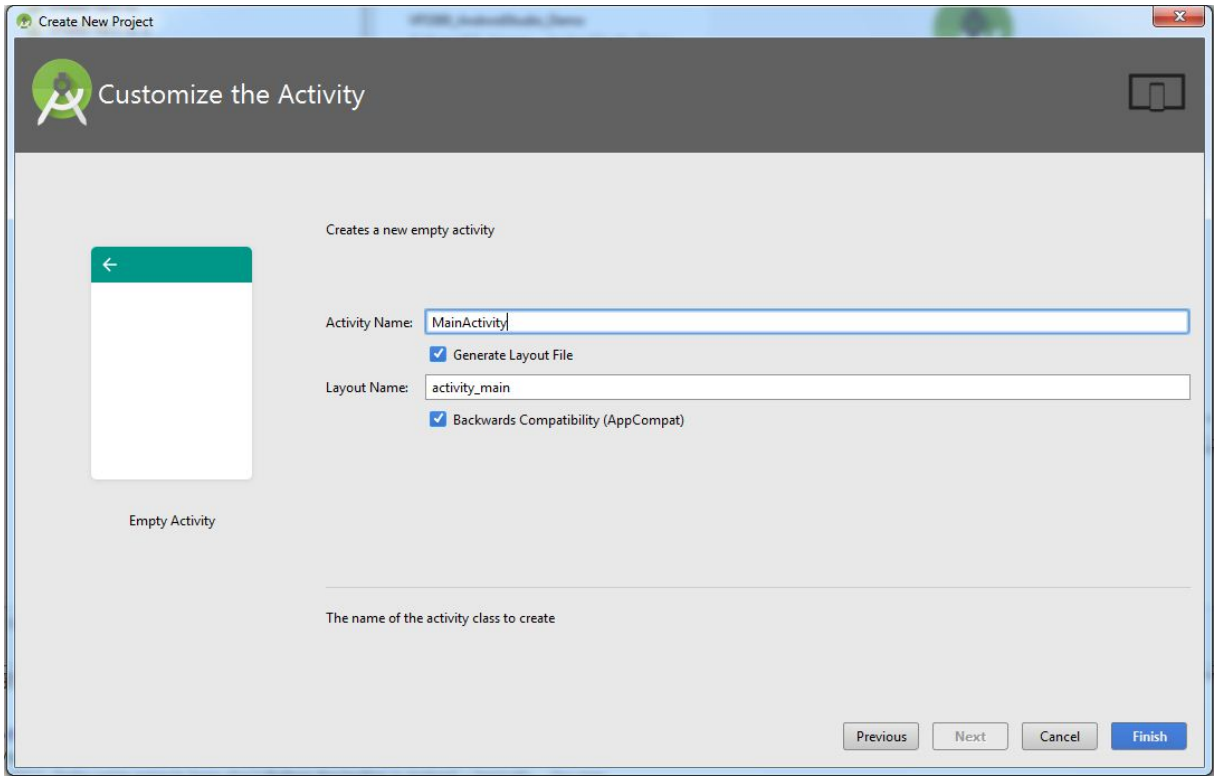
Create a new Android Application project in Android Studio as an Empty Activity




Create New Project
Target Android Devices
Select the form factors your app will run on
Different platforms may require separate SDKs
Phone and Tablet
Minimum SDK
API 22: Android 5.1 (Lollipop)
Lower API levels target more devices, but have fewer features available.
By targeting API 22 and later, your app will run on approximately 62.6% of the devices that are active on the Google Play Store.
Help me choose
Wear
Minimum SDK
API 21: Android 5.0 (Lollipop)
TV
Minimum SDK
API 21: Android 5.0 (Lollipop)
Android Auto
Previous
Next
Cancel
Finish


Create New Project
Installing Requested Components
SDK Path: C:\Users\ \AppData\Local\Android\Sdk
ut for Android 1.0.2 in C:\Users\ \AppData\Local\Android\Sdk\extras\m2repository\com\android\support\constraint\constraint-layout\1.0.2
for Android 1.0.2 (revision: 1)" complete.
for Android 1.0.2 (revision: 1)" finished.
\AppData\Local\Android\Sdk\add-ons\addon-google_apis-google-24\package.xml
\AppData\Local\Android\Sdk\build-tools\25.0.3\package.xml
\AppData\Local\Android\Sdk\build-tools\26.0.0\package.xml
\AppData\Local\Android\Sdk\build-tools\26.0.1\package.xml
\AppData\Local\Android\Sdk\docs\package.xml
\AppData\Local\Android\Sdk\emulator\package.xml
\AppData\Local\Android\Sdk\extras\android\m2repository\package.xml
\AppData\Local\Android\Sdk\extras\google\m2repository\package.xml
\AppData\Local\Android\Sdk\extras\intel\Hardware_Accelerated_Execution_Manager\package.xml
\AppData\Local\Android\Sdk\extras\m2repository\com\android\support\constraint\constraint-layout-solver\1.0.2\package.xml
\AppData\Local\Android\Sdk\extras\m2repository\com\android\support\constraint\constraint-layout\1.0.2\package.xml
\AppData\Local\Android\Sdk\patcher\v4\package.xml
\AppData\Local\Android\Sdk\platform-tools\package.xml
\AppData\Local\Android\Sdk\platforms\android-22\package.xml
\AppData\Local\Android\Sdk\platforms\android-23\package.xml
\AppData\Local\Android\Sdk\platforms\android-25\package.xml
\AppData\Local\Android\Sdk\sources\android-25\package.xml
\AppData\Local\Android\Sdk\system-images\android-25\google_apis\x86\package.xml
\AppData\Local\Android\Sdk\tools\package.xml
Done
Previous
Next
Cancel
Finish





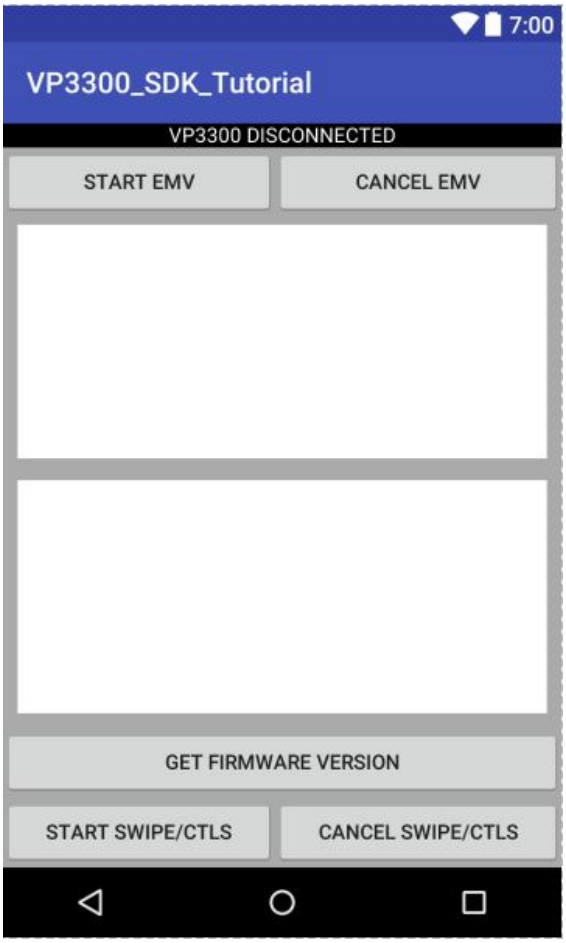
7.9.2 Step 2: Import IDTechSDK for NEO2

Import the necessary libraries

7.9.3 Step 3: Design Interface

Design the User Interface by editing the main layout XML file
Open your layout and add items to so it contains the following buttons/fields (sample code provide at end of section):

- Add a TextView to the top that will signify connection/disconnection status.
- Add two TextViews to communicate data from the NEO2 and for EMV LCD display information. Remove the Editable behavior if you don't want the keyboard to pop up if you accidentally select it.
- Add buttons to execute the following functions:
  - Get Firmware
  - Start MSR/ CTLS
  - Start ICC EMV
  - Complete ICC EMV
  - Cancel Transaction



7.9.4 Step 4: Configure Activity File

In the activity file, perform the following:

- Add Import statements to utilize libraries
- Implement OnReceiverListener for the activity

- Enable permissions for the application
- Define the association for all the elements on the layout: The connection label, the two text views, and the 5 buttons

Layout Source Code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#aaaaaa"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/status_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#000000"
        android:gravity="center_vertical|center_horizontal"
        android:text="NEO2 DISCONNECTED"
        android:textColor="#FFFFFF" />
    <LinearLayout
        android:id="@+id/linearLayoutBottom2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <Button
            android:id="@+id/btn_StartEMV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:gravity="center_vertical|center_horizontal"
            android:text="Start EMV" />
        <Button
            android:id="@+id/btn_CancelEMV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Cancel EMV" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayoutEditText"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="#dddddd"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical" >
        <ScrollView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_marginBottom="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginTop="5dp"
            android:background="#ffffff" >
            <TextView
                android:id="@+id/lcdLog"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:text=""
                android:textColor="#000000"
                android:textSize="12sp"
                android:typeface="monospace" >
            </TextView>
        </ScrollView>
    </LinearLayout>
        <LinearLayout
        android:id="@+id/linearLayoutEditText2"
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="#dddddd"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical" >
        <ScrollView
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_marginBottom="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="10dp"
            android:layout_marginTop="5dp"
```

```
            android:background="#ffffff" >
            <TextView
                android:id="@+id/textLog"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:text=""
                android:textColor="#000000"
                android:textSize="12sp"
                android:typeface="monospace" >
            </TextView>
        </ScrollView>
    </LinearLayout>
        <LinearLayout
        android:id="@+id/linearLayoutBottom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <Button
            android:id="@+id/btn_getFirmware"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:gravity="center_vertical|center_horizontal"
            android:text="Get Firmware Version" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayoutBottom4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >

        <Button
            android:id="@+id/btn_startSwipe"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Start Swipe/CTLS" />
        <Button
            android:id="@+id/btn_cancelSwipe"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Cancel Swipe/CTLS" />
    </LinearLayout>
</LinearLayout>
```

7.9.5 Step 5: Configure Method File

In the activity file, perform the following:

- set delegate and initialize NEO2 object in the onCreate method. Reference: Allocate/initialize NEO2 objects
- set correct device type.

```
// declaring the instance of the NEO2Reader;
private IDT_NEO2 myNEO2Reader = null;
private TextView connectStatusTextView;
private TextView textLog;
private TextView lcdLog;
private Button btnGetFirmware;
private Button btnStartSwipe;
private Button btnCancelSwipe;
private Button btnStartEMV;
private Button btnCancelEMV;
private Handler handler = new Handler();
private boolean isReaderConnected = false;
private String info = "";
private String detail = "";
private BluetoothAdapter mBtAdapter = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    handler = new Handler();
    btnStartEMV = (Button)findViewById(R.id.btn_StartEMV);
    btnCancelEMV = (Button)findViewById(R.id.btn_CancelEMV);
    btnGetFirmware = (Button)findViewById(R.id.btn_getFirmware);
    btnStartSwipe = (Button)findViewById(R.id.btn_startSwipe);
    btnCancelSwipe = (Button)findViewById(R.id.btn_cancelSwipe);
    textLog = (TextView)findViewById(R.id.textLog);
    lcdLog = (TextView)findViewById(R.id.lcdLog);
```

```
    connectStatusTextView = (TextView)findViewById(R.id.status_text);
    if(myNEO2Reader!=null){
        myNEO2Reader.unregisterListen();
        myNEO2Reader.release();
        myNEO2Reader = null;
    }
    myNEO2Reader = new IDT_NEO2(this,this);
    myNEO2Reader.device_setDeviceType(DEVICE_TYPE.DEVICE_NEO2_USB);
    myNEO2Reader.registerListen();

}
```

- If available on device, activate BLE scanning. Reference: Activate Bluetooth Low Energy

  – Implement protocol delegate com.idtechproducts.device.OnReceiverListener.deviceConnected() and com.idtechproducts.device.OnReceiverListener.deviceDisconnected() to monitor connect/disconnect events and modify our connection label upon change. Reference: Implement OnReceiverListener for the activity
  Note: This notification may come back on a thread different that the UI thread, so we want to make sure to use a handler to send to main UI thread.

```
private Runnable doUpdateLabel = new Runnable()
{
    public void run()
    {
        if(!isReaderConnected){
            connectStatusTextView.setText("NEO2 DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("NEO2 CONNECTED");
        }
    }
};
@Override
public void deviceConnected() {
    isReaderConnected = true;
    handler.post(doUpdateLabel);
}

@Override
public void deviceDisconnected() {
    isReaderConnected = false;
    handler.post(doUpdateLabel);
}
```

-Implement protocol delegate com.idtechproducts.device.OnReceiverListener.swipeMSRData() to receive unsolicited card swipe data.

```
private Runnable doUpdateStatus = new Runnable()
{
    public void run()
    {
        lcdLog.setText(info);
        textLog.setText(detail);
    }
};
@Override
public void swipeMSRData(IDTMSRData card) {
    if (card.cardData[0] != (byte)0x01 && card.track1Length == 0 && card.track2Length == 0 && card.
      track3Length == 0)
        info = "Swipe/Tap data didn't read correctly";
    else
        info = "Swipe/Tap Read Successfully";
    detail = Common.parse_MSRData(myNEO2Reader.device_getDeviceType(), card);
    handler.post(doUpdateStatus);

}
```

- Implement protocol delegate com.idtechproducts.device.OnReceiverListener.emvTransactionData() to report EMV transaction results

```
    public void emvTransactionData(IDTEMVData emvData) {

        detail += Common.emvErrorCodes(emvData.result);
        detail += "\r\n";
        if (emvData.result == IDTEMVData.START_TRANS_SUCCESS)
            detail += "Start transaction response:\r\n";
```

```
    else if (emvData.result == IDTEMVData.GO_ONLINE)
        detail += "\r\nAuthentication response:\r\n";
    else
        detail += "\r\nComplete Transaction response:\r\n";
    if (!emvData.unencryptedTags.isEmpty())
    {
        detail += "Unencrypted Tags:\r\n";
        Set<String> keys = emvData.unencryptedTags.keySet();
        for(String key: keys){
            detail += key + ": ";
            byte[] data = emvData.unencryptedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    if (!emvData.maskedTags.isEmpty())
    {
        detail += "Masked Tags:\r\n";
        Set<String> keys = emvData.maskedTags.keySet();
        for(String key: keys){
            detail += key + ": ";
            byte[] data = emvData.maskedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    if (!emvData.encryptedTags.isEmpty())
    {
        detail += "Encrypted Tags:\r\n";
        Set<String> keys = emvData.encryptedTags.keySet();
        for(String key: keys){
            detail += key + ": ";
            byte[] data = emvData.encryptedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    handler.post(doUpdateStatus);
     if (emvData.result == IDTEMVData.GO_ONLINE){
        //Auto Complete
         byte[] response = new byte[]{0x30,0x30};
         myNEO2Reader.emv_completeTransaction(false, response, null, null,null);
     }
     else if (emvData.result == IDTEMVData.START_TRANS_SUCCESS){
            //Auto Authenticate
         myNEO2Reader.emv_authenticateTransaction(null);
        }

}
```

- Implement protocol delegate com.idtechproducts.device.OnReceiverListener.emvTransactionData() to receive LCD messages, and automatically select 1st menu item/language when presented with choices. Normal operation would require a choice be made by card holder.

```
public void lcdDisplay(int mode, String[] lines, int timeout) {

            if (mode == 0x01) //Menu Display
            {
                //automatically select 1st application
                myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else if (mode == 0x08) //Language Menu Display
            {
                //automatically select first language
                myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else{
                ResDataStruct toData = new ResDataStruct();
                info = lines[0];
                handler.post(doUpdateStatus);
            }
}
```

- Implement the button press methods

```
btnGetFirmware.setOnClickListener(new Button.OnClickListener(){
    public void onClick(View v) {
        info = "Getting Firmware\n";
        detail = "";
        handler.post(doUpdateStatus);
        StringBuilder sb = new StringBuilder();
        int ret = myNEO2Reader.device_getFirmwareVersion(sb);
        if (ret == ErrorCode.SUCCESS) {
            info += "Firmware Version: " + sb.toString();
            detail = "";
            handler.post(doUpdateStatus);
```

```
            }
            else {
                info += "GetFirmwareVersion: Failed\n";
                info += "Status: "+ myNEO2Reader.device_getResponseCodeString(ret)+"";
                detail = "";
                handler.post(doUpdateStatus);
                }
        }
    });

    btnStartEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {;
        detail = "";
        info = "Starting EMV Transaction\n";
        handler.post(doUpdateStatus);
            IDT_NEO2.emv_allowFallback(true);
            myNEO2Reader.emv_startTransaction(1.00, 0.00, 0, 30, null, false);
        }
    });

    btnCancelEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Canceling EMV Transaction\n";
            handler.post(doUpdateStatus);
            ResDataStruct resData = new ResDataStruct();
            myNEO2Reader.emv_cancelEMVTransaction(resData);
        }
    });

    btnStartSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Starting Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_startMSRSwipe();
        }
    });

    btnCancelSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Cancelling Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_cancelMSRSwipe();
        }
    });
```

7.9.6 Complete code listing

```
package com.example.neo2_sdk_tutorial;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.Set;

import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

import com.idtechproducts.device.*;
import com.idtechproducts.device.ReaderInfo.DEVICE_TYPE;
import com.idtechproducts.device.ReaderInfo.
      CAPTURE_ENCODE_TYPE;
import com.idtechproducts.device.ReaderInfo.
      CAPTURE_ENCRYPT_TYPE;
import com.idtechproducts.device.ReaderInfo.
      EVENT_MSR_Types;

public class MainActivity extends Activity implements  OnReceiverListener{

    // declaring the instance of the NEO2Reader;
    private IDT_NEO2 myNEO2Reader = null;
    private TextView connectStatusTextView;
```

```
private TextView textLog;
private TextView lcdLog;
private Button btnGetFirmware;
private Button btnStartSwipe;
private Button btnCancelSwipe;
private Button btnStartEMV;
private Button btnCancelEMV;
private Handler handler = new Handler();
private boolean isReaderConnected = false;
private String info = "";
private String detail = "";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    handler = new Handler();
    btnStartEMV = (Button)findViewById(R.id.btn_StartEMV);
    btnCancelEMV = (Button)findViewById(R.id.btn_CancelEMV);
    btnGetFirmware = (Button)findViewById(R.id.btn_getFirmware);
    btnStartSwipe = (Button)findViewById(R.id.btn_startSwipe);
    btnCancelSwipe = (Button)findViewById(R.id.btn_cancelSwipe);
    textLog = (TextView)findViewById(R.id.textLog);
    lcdLog = (TextView)findViewById(R.id.lcdLog);
    connectStatusTextView = (TextView)findViewById(R.id.status_text);
    if(myNEO2Reader!=null){
        myNEO2Reader.unregisterListen();
        myNEO2Reader.release();
        myNEO2Reader = null;
    }
    myNEO2Reader = new IDT_NEO2(this,this);
    myNEO2Reader.device_setDeviceType(DEVICE_TYPE.DEVICE_NEO2_AJ);
    myNEO2Reader.registerListen();
    loadXMLfile();


    btnGetFirmware.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            info = "Getting Firmware\n";
            detail = "";
            handler.post(doUpdateStatus);
            StringBuilder sb = new StringBuilder();
            int ret = myNEO2Reader.device_getFirmwareVersion(sb);
            if (ret == ErrorCode.SUCCESS) {
                info += "Firmware Version: " + sb.toString();
                detail = "";
                handler.post(doUpdateStatus);
            }
            else {
                info += "GetFirmwareVersion: Failed\n";
                info += "Status: "+ myNEO2Reader.device_getResponseCodeString(ret)+"";
                detail = "";
                handler.post(doUpdateStatus);
                }
        }
    });

    btnStartEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {;
        detail = "";
        info = "Starting EMV Transaction\n";
        handler.post(doUpdateStatus);
            IDT_NEO2.emv_allowFallback(true);
            myNEO2Reader.emv_startTransaction(1.00, 0.00, 0, 30, null, false);
        }
    });

    btnCancelEMV.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Canceling EMV Transaction\n";
            handler.post(doUpdateStatus);
            ResDataStruct resData = new ResDataStruct();
            myNEO2Reader.emv_cancelEMVTransaction(resData);
        }
    });

    btnStartSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
            detail = "";
            info = "Starting Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_startMSRSwipe();
        }
    });

    btnCancelSwipe.setOnClickListener(new Button.OnClickListener(){
        public void onClick(View v) {
```

```
            detail = "";
            info = "Cancelling Swipe/Tap Transaction\n";
            handler.post(doUpdateStatus);
            myNEO2Reader.msr_cancelMSRSwipe();
        }
    });

}

@Override
public void ICCNotifyInfo(byte[] arg0, String arg1) {
    // TODO Auto-generated method stub

}

@Override
public void LoadXMLConfigFailureInfo(int arg0, String arg1) {
    // TODO Auto-generated method stub

}

@Override
public void autoConfigCompleted(StructConfigParameters arg0) {
    // TODO Auto-generated method stub

}

@Override
public void autoConfigProgress(int arg0) {
    // TODO Auto-generated method stub

}

private Runnable doUpdateLabel = new Runnable()
{
    public void run()
    {
        if(!isReaderConnected){
            connectStatusTextView.setText("NEO2 DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("NEO2 CONNECTED");
        }
    }
};
@Override
public void deviceConnected() {
    isReaderConnected = true;
    handler.post(doUpdateLabel);
}

@Override
public void deviceDisconnected() {
    isReaderConnected = false;
    handler.post(doUpdateLabel);
}

private void printTags(IDTEMVData emvData)
{

}

@Override
public void emvTransactionData(IDTEMVData emvData) {

    detail += Common.emvErrorCodes(emvData.result);
    detail += "\r\n";
    if (emvData.result == IDTEMVData.START_TRANS_SUCCESS)
        detail += "Start transaction response:\r\n";
    else if (emvData.result == IDTEMVData.GO_ONLINE)
        detail += "\r\nAuthentication response:\r\n";
    else
        detail += "\r\nComplete Transaction response:\r\n";
    if (!emvData.unencryptedTags.isEmpty())
    {
        detail += "Unencrypted Tags:\r\n";
        Set<String> keys = emvData.unencryptedTags.keySet();
        for(String key: keys){
            detail += key + ": ";
            byte[] data = emvData.unencryptedTags.get(key);
            detail += Common.getHexStringFromBytes(data) + "\r\n";
        }
    }
    if (!emvData.maskedTags.isEmpty())
    {
        detail += "Masked Tags:\r\n";
        Set<String> keys = emvData.maskedTags.keySet();
```

```
            for(String key: keys){
                detail += key + ": ";
                byte[] data = emvData.maskedTags.get(key);
                detail += Common.getHexStringFromBytes(data) + "\r\n";
            }
        }
        if (!emvData.encryptedTags.isEmpty())
        {
            detail += "Encrypted Tags:\r\n";
            Set<String> keys = emvData.encryptedTags.keySet();
            for(String key: keys){
                detail += key + ": ";
                byte[] data = emvData.encryptedTags.get(key);
                detail += Common.getHexStringFromBytes(data) + "\r\n";
            }
        }
        handler.post(doUpdateStatus);
         if (emvData.result == IDTEMVData.GO_ONLINE){
            //Auto Complete
             byte[] response = new byte[]{0x30,0x30};
             myNEO2Reader.emv_completeTransaction(false, response, null, null,null);
         }
         else if (emvData.result == IDTEMVData.START_TRANS_SUCCESS){
                //Auto Authenticate
             myNEO2Reader.emv_authenticateTransaction(null);
            }

    }

    public void lcdDisplay(int mode, String[] lines, int timeout) {

                if (mode == 0x01) //Menu Display
                {
                    //automatically select 1st application
                    myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
                }
                else if (mode == 0x08) //Language Menu Display
                {
                    //automatically select first language
                    myNEO2Reader.emv_lcdControlResponse((byte)mode, (byte)0x01);
                }
                else{
                    ResDataStruct toData = new ResDataStruct();
                    info = lines[0];
                    handler.post(doUpdateStatus);
                }
    }


    @Override
    public void msgAudioVolumeAjustFailed() {
        // TODO Auto-generated method stub

    }

    @Override
    public void msgRKICompleted(String arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void msgToConnectDevice() {
        // TODO Auto-generated method stub

    }

    private Runnable doUpdateStatus = new Runnable()
    {
        public void run()
        {
            lcdLog.setText(info);
            textLog.setText(detail);
        }
    };
    @Override
    public void swipeMSRData(IDTMSRData card) {
        if (card.cardData[0] != (byte)0x01 && card.track1Length == 0 && card.track2Length == 0 && card.
      track3Length == 0)
            info = "Swipe/Tap data didn't read correctly";
        else
            info = "Swipe/Tap Read Successfully";
        detail = Common.parse_MSRData(myNEO2Reader.device_getDeviceType(), card);
        handler.post(doUpdateStatus);

    }
```

```
    @Override
    public void timeout(int arg0) {
        // TODO Auto-generated method stub

    }

    private String getXMLFileFromRaw(String fileName ,int res){
        //the target filename in the application path
        String fileNameWithPath = null;
        fileNameWithPath = fileName;
        String newFilename = fileName;

        try {
            InputStream in = getResources().openRawResource(res);
            int length = in.available();
            byte [] buffer = new byte[length];
            in.read(buffer);
            in.close();
            deleteFile(fileNameWithPath);
            FileOutputStream fout = openFileOutput(fileNameWithPath, MODE_PRIVATE);
            fout.write(buffer);
            fout.close();

            // to refer to the application path
            File fileDir = this.getFilesDir();
            fileNameWithPath = fileDir.getParent() + java.io.File.separator + fileDir.getName();
            fileNameWithPath += java.io.File.separator+newFilename;

        } catch(Exception e){
            e.printStackTrace();
            fileNameWithPath = null;
        }
        return fileNameWithPath;
    }

    private String getConfigurationFileFromRaw( ){
        return getXMLFileFromRaw("idt_unimagcfg_default.xml",R.raw.idt_unimagcfg_default);
        }
    private boolean isFileExist(String path) {
        if(path==null)
            return false;
        File file = new File(path);
        if (!file.exists()) {
          return false ;
        }
        return true;
    }
    private void loadXMLfile(){

        //load the XML configuration file
        String fileNameWithPath = getConfigurationFileFromRaw();
        if(!isFileExist(fileNameWithPath)) {
            fileNameWithPath = null;
        }
        // Network operation is prohibited in the UI Thread if target API is 11 or above.
        // If target API is 11 or above, please use AsyncTask to avoid errors.
        myNEO2Reader.config_setXMLFileNameWithPath(fileNameWithPath);
        Log.d("Demo Info >>>>>","loadingConfigurationXMLFile begin.");
        myNEO2Reader.config_loadingConfigurationXMLFile(true);
    }




}
```

Chapter 8

NEO2 Error Code Reference

```
0000    OK
0001    Incorrect Header Tag
0002    Unknown Command
0003    Unknown Sub-Command
0004    CRC Error in Frame
0005    Incorrect Parameter
0006    Parameter Not Supported
0007    Mal-formatted Data
0008    Timeout
000A    Failed / NACK
000B    Command not Allowed
000C    Sub-Command not Allowed
000D    Buffer Overflow (Data Length too large for reader buffer)
000E    User Interface Event
0011    Communication type not supported, VT-1, burst, etc.
0012    Secure interface is not functional or is in an intermediate state.
0013    Data field is not mod 8
0014    Pad 0x80 not found where expected
0015    Specified key type is invalid
0016    Could not retrieve key from the SAM (InitSecureComm)
0017    Hash code problem
0018    Could not store the key into the SAM (InstallKey)
0019    Frame is too large
001A    Unit powered up in authentication state but POS must resend the InitSecureComm command
001B    The EEPROM may not be initialized because SecCommInterface does not make sense
001C    Problem encoding APDU
0020    Unsupported Index (ILM) SAM Transceiver error - problem communicating with the SAM (Key Mgr)
0021    Unexpected Sequence Counter in multiple frames for single bitmap (ILM) Length error in data
      returned from the SAM (Key Mgr)
0022    Improper bit map (ILM)
0023    Request Online Authorization
0024    ViVOCard3 raw data read successful
0025    Message index not available (ILM) ViVOcomm activate transaction card type (ViVOcomm)
0026    Version Information Mismatch (ILM)
0027    Not sending commands in correct index message index (ILM)
0028    Time out or next expected message not received (ILM)
0029    ILM languages not available for viewing (ILM)
002A    Other language not supported (ILM)
0050    Auto-Switch OK
0051    Auto-Switch failed
0060    Data not exist
0061    Data Full
0062    Write Flash Error
0063    Ok and Have Next Command
0090    Account DUKPT Key not exist
0091    Account DUKPT Key KSN exhausted
EE00    OK
EE01    Incorrect Header Tag
EE02    Unknown Command
EE03    Unknown Sub-Command
EE04    CRC Error in Frame
EE05    Incorrect Parameter
EE06    Parameter Not Supported
EE07    Mal-formatted Data
EE08    Timeout
EE0A    Failed / NACK
EE0B    Command not Allowed
EE0C    Sub-Command not Allowed
EE0D    Buffer Overflow (Data Length too large for reader buffer)
```

```
EE0E    User Interface Event
EE11    Communication type not supported, VT-1, burst, etc.
EE12    Secure interface is not functional or is in an intermediate state.
EE13    Data field is not mod 8
EE14    Pad 0x80 not found where expected
EE15    Specified key type is invalid
EE16    Could not retrieve key from the SAM (InitSecureComm)
EE17    Hash code problem
EE18    Could not store the key into the SAM (InstallKey)
EE19    Frame is too large
EE1A    Unit powered up in authentication state but POS must resend the InitSecureComm command
EE1B    The EEPROM may not be initialized because SecCommInterface does not make sense
EE1C    Problem encoding APDU
EE20    Unsupported Index (ILM) SAM Transceiver error - problem communicating with the SAM (Key Mgr)
EE21    Unexpected Sequence Counter in multiple frames for single bitmap (ILM) Length error in data
      returned from the SAM (Key Mgr)
EE22    Improper bit map (ILM)
EE23    Request Online Authorization
EE24    ViVOCard3 raw data read successful
EE25    Message index not available (ILM) ViVOcomm activate transaction card type (ViVOcomm)
EE26    Version Information Mismatch (ILM)
EE27    Not sending commands in correct index message index (ILM)
EE28    Time out or next expected message not received (ILM)
EE29    ILM languages not available for viewing (ILM)
EE2A    Other language not supported (ILM)
EE50    Auto-Switch OK
EE51    Auto-Switch failed
EE60    Data not exist
EE61    Data Full
EE62    Write Flash Error
EE63    Ok and Have Next Command
EE90    Account DUKPT Key not exist
EE91    Account DUKPT Key KSN exhausted
```

Chapter 9

Enumeration Reference

IDTMSRData

```
typedef enum _CAPTURE_ENCODE_TYPE{
    CAPTURE_ENCODE_TYPE_ISOABA=0,
    CAPTURE_ENCODE_TYPE_AAMVA=1,
    CAPTURE_ENCODE_TYPE_Other=3,
    CAPTURE_ENCODE_TYPE_Raw=4,
    CAPTURE_ENCODE_TYPE_JIS_II=5,
    CAPTURE_ENCODE_TYPE_JIS_I=6,
    CAPTURE_ENCODE_TYPE_MANUAL_ENTRY=7
} CAPTURE_ENCODE_TYPE;
```

```
typedef enum{
    CAPTURE_ENCRYPT_TYPE_TDES=0,
    CAPTURE_ENCRYPT_TYPE_AES=1
} CAPTURE_ENCRYPT_TYPE;
```

IDTCommon

```
typedef enum{
    POWER_ON_OPTION_IFS_FLAG=1,
    POWER_ON_OPTION_EXPLICIT_PPS_FLAG=2,
    POWER_ON_OPTION_AUTO_PPS_FLAG=64,
    POWER_ON_OPTION_IFS_RESPONSE_CHECK_FLAG=128
}POWER_ON_OPTION;
```

```
typedef enum{
    LANGUAGE_TYPE_ENGLISH=1,
    LANGUAGE_TYPE_PORTUGUESE,
    LANGUAGE_TYPE_SPANISH,
    LANGUAGE_TYPE_FRENCH
}LANGUAGE_TYPE;
```

```
typedef enum{
    PIN_KEY_TDES_MKSK_extp=0x00,
    PIN_KEY_TDES_DUKPT_extp=0x01,
    PIN_KEY_TDES_MKSK_intl=0x10,
    PIN_KEY_TDES_DUKPT_intl=0x11,
}PIN_KEY_Types;
```

```
typedef enum{
    EVENT_PINPAD_UNKNOWN = 11,
    EVENT_PINPAD_ENCRYPTED_PIN,
    EVENT_PINPAD_NUMERIC,
    EVENT_PINPAD_AMOUNT,
    EVENT_PINPAD_ACCOUNT,
    EVENT_PINPAD_ENCRYPTED_DATA,
    EVENT_PINPAD_CANCEL,
    EVENT_PINPAD_TIMEOUT,
    EVENT_PINPAD_FUNCTION_KEY,
    EVENT_PINPAD_DATA_ERROR
}EVENT_PINPAD_Types;
```

```
typedef enum{
    IDT_DEVICE_BTPAY_IOS = 0,
    IDT_DEVICE_BTPAY_OSX_BT,
    IDT_DEVICE_BTPAY_OSX_USB,
    IDT_DEVICE_UNIPAY_IOS,
    IDT_DEVICE_UNIPAY_OSX_USB,
    IDT_DEVICE_NEO2_IOS,
    IDT_DEVICE_NEO2_OSX_USB,
    IDT_DEVICE_IMAG_IOS,
    IDT_DEVICE_VENDI_MOBILE
}IDT_DEVICE_Types;
```

```
typedef enum{
    EVENT_MSR_UNKNOWN = 31,
    EVENT_MSR_CARD_DATA,
    EVENT_MSR_CANCEL_KEY,
    EVENT_MSR_BACKSPACE_KEY,
    EVENT_MSR_ENTER_KEY,
    EVENT_MSR_DATA_ERROR,
    EVENT_MSR_ICC_START,
    EVENT_BTPAY_CARD_DATA,
    EVENT_NEO2_EMV_NO_ICC_MSR_DATA,
    EVENT_NEO2_EMV_FALLBACK_DATA
}EVENT_MSR_Types;
```

```
typedef enum{
    EVENT_ACTIVE_TRANSACTION = 51
}EVENT_CTLS_Types;
```

```
typedef enum {
    RETURN_CODE_DO_SUCCESS = 0,
    RETURN_CODE_ERR_DISCONNECT,
    RETURN_CODE_ERR_CMD_RESPONSE,
    RETURN_CODE_ERR_TIMEDOUT,
    RETURN_CODE_ERR_INVALID_PARAMETER,
    RETURN_CODE_SDK_BUSY_MSR,
    RETURN_CODE_SDK_BUSY_PINPAD,
    RETURN_CODE_SDK_BUSY_CTLS,
    RETURN_CODE_ERR_OTHER,
    RETURN_CODE_FAILED,
    RETURN_CODE_NOT_ATTACHED,
    RETURN_CODE_MONO_AUDIO,
    RETURN_CODE_CONNECTED,
    RETURN_CODE_LOW_VOLUME,
    RETURN_CODE_CANCELED,


    RETURN_CODE_EMV_AUTHORIZATION_ACCEPTED = 0x0E00,
    RETURN_CODE_EMV_AUTHORIZATION_UNABLE_TO_GO_ONLINE = 0x0E01,
    RETURN_CODE_EMV_AUTHORIZATION_TECHNICAL_ISSUE = 0x0E02,
    RETURN_CODE_EMV_AUTHORIZATION_DECLINED = 0x0E03,
    RETURN_CODE_EMV_AUTHORIZATION_ISSUER_REFERRAL = 0x0E04,
```

```
    RETURN_CODE_EMV_APPROVED = 0x0F00,ction
    RETURN_CODE_EMV_DECLINED = 0x0F01,
    RETURN_CODE_EMV_GO_ONLINE = 0x0F02,
    RETURN_CODE_EMV_FAILED = 0x0F03,
    RETURN_CODE_EMV_SYSTEM_ERROR = 0x0F05,
    RETURN_CODE_EMV_NOT_ACCEPTED = 0x0F07,
    RETURN_CODE_EMV_FALLBACK = 0x0F0A,
    RETURN_CODE_EMV_CANCEL = 0x0F0C,
    RETURN_CODE_EMV_TIMEOUT = 0x0F0D,
    RETURN_CODE_EMV_OTHER_ERROR = 0x0F0F,
    RETURN_CODE_EMV_OFFLINE_APPROVED = 0x0F10,
    RETURN_CODE_EMV_OFFLINE_DECLINED = 0x0F11,



    RETURN_CODE_EMV_NEW_SELECTION = 0x0F21,
    RETURN_CODE_EMV_NO_AVAILABLE_APPS = 0x0F22,
    RETURN_CODE_EMV_NO_TERMINAL_FILE = 0x0F23,
    RETURN_CODE_EMV_NO_CAPK_FILE = 0x0F24,
    RETURN_CODE_EMV_NO_CRL_ENTRY = 0x0F25,
    RETURN_CODE_BLOCKING_DISABLED = 0x0FFE,
    RETURN_CODE_COMMAND_UNAVAILABLE = 0x0FFF


} RETURN_CODE;




typedef enum{
    EMV_RESULT_CODE_V2_APPROVED_OFFLINE = 0x0000,
    EMV_RESULT_CODE_V2_DECLINED_OFFLINE = 0x0001,
    EMV_RESULT_CODE_V2_APPROVED = 0x0002,
    EMV_RESULT_CODE_V2_DECLINED = 0x0003,
    EMV_RESULT_CODE_V2_GO_ONLINE = 0x0004,
    EMV_RESULT_CODE_V2_CALL_YOUR_BANK = 0x0005,
    EMV_RESULT_CODE_V2_NOT_ACCEPTED = 0x0006,
    EMV_RESULT_CODE_V2_USE_MAGSTRIPE = 0x0007,
    EMV_RESULT_CODE_V2_TIME_OUT = 0x0008,
    EMV_RESULT_CODE_V2_START_TRANS_SUCCESS = 0x0010,
    EMV_RESULT_CODE_V2_MSR_SUCCESS = 0x0011,
    EMV_RESULT_CODE_V2_FILE_ARG_INVALID = 0x1001,
    EMV_RESULT_CODE_V2_FILE_OPEN_FAILED = 0x1002,
    EMV_RESULT_CODE_V2_FILE_OPERATION_FAILED = 0X1003,
    EMV_RESULT_CODE_V2_MEMORY_NOT_ENOUGH = 0x2001,
    EMV_RESULT_CODE_V2_SMARTCARD_FAIL = 0x3001,
    EMV_RESULT_CODE_V2_SMARTCARD_INIT_FAILED = 0x3003,
    EMV_RESULT_CODE_V2_FALLBACK_SITUATION = 0x3004,
    EMV_RESULT_CODE_V2_SMARTCARD_ABSENT = 0x3005,
    EMV_RESULT_CODE_V2_SMARTCARD_TIMEOUT = 0x3006,
    EMV_RESULT_CODE_V2_MSR_CARD_ERROR = 0x3007,
    EMV_RESULT_CODE_V2_PARSING_TAGS_FAILED= 0X5001,
    EMV_RESULT_CODE_V2_CARD_DATA_ELEMENT_DUPLICATE = 0X5002,
    EMV_RESULT_CODE_V2_DATA_FORMAT_INCORRECT = 0X5003,
    EMV_RESULT_CODE_V2_APP_NO_TERM = 0X5004,
    EMV_RESULT_CODE_V2_APP_NO_MATCHING = 0X5005,
    EMV_RESULT_CODE_V2_AMANDATORY_OBJECT_MISSING = 0X5006,
    EMV_RESULT_CODE_V2_APP_SELECTION_RETRY = 0X5007,
    EMV_RESULT_CODE_V2_AMOUNT_ERROR_GET = 0X5008,
    EMV_RESULT_CODE_V2_CARD_REJECTED = 0X5009,
    EMV_RESULT_CODE_V2_AIP_NOT_RECEIVED = 0X5010,
    EMV_RESULT_CODE_V2_AFL_NOT_RECEIVEDE = 0X5011,
    EMV_RESULT_CODE_V2_AFL_LEN_OUT_OF_RANGE = 0X5012,
    EMV_RESULT_CODE_V2_SFI_OUT_OF_RANGE = 0X5013,
    EMV_RESULT_CODE_V2_AFL_INCORRECT = 0X5014,
    EMV_RESULT_CODE_V2_EXP_DATE_INCORRECT = 0X5015,
    EMV_RESULT_CODE_V2_EFF_DATE_INCORRECT = 0X5016,
    EMV_RESULT_CODE_V2_ISS_COD_TBL_OUT_OF_RANGE = 0X5017,
    EMV_RESULT_CODE_V2_CRYPTOGRAM_TYPE_INCORRECT = 0X5018,
    EMV_RESULT_CODE_V2_PSE_BY_CARD_NOT_SUPPORTED = 0X5019,
    EMV_RESULT_CODE_V2_USER_LANGUAGE_SELECTED = 0X5020,
    EMV_RESULT_CODE_V2_SERVICE_NOT_ALLOWED = 0X5021,
    EMV_RESULT_CODE_V2_NO_TAG_FOUND = 0X5022,
    EMV_RESULT_CODE_V2_CARD_BLOCKED = 0X5023,
    EMV_RESULT_CODE_V2_LEN_INCORRECT = 0X5024,
    EMV_RESULT_CODE_V2_CARD_COM_ERROR = 0X5025,
    EMV_RESULT_CODE_V2_TSC_NOT_INCREASED = 0X5026,
    EMV_RESULT_CODE_V2_HASH_INCORRECT = 0X5027,
    EMV_RESULT_CODE_V2_ARC_NOT_PRESENCED = 0X5028,
    EMV_RESULT_CODE_V2_ARC_INVALID = 0X5029,
    EMV_RESULT_CODE_V2_COMM_NO_ONLINE = 0X5030,
    EMV_RESULT_CODE_V2_TRAN_TYPE_INCORRECT = 0X5031,
    EMV_RESULT_CODE_V2_APP_NO_SUPPORT = 0X5032,
    EMV_RESULT_CODE_V2_APP_NOT_SELECT = 0X5033,
    EMV_RESULT_CODE_V2_LANG_NOT_SELECT = 0X5034,
    EMV_RESULT_CODE_V2_TERM_DATA_NOT_PRESENCED = 0X5035,
```

```
    EMV_RESULT_CODE_V2_CVM_TYPE_UNKNOWN = 0X6001,
    EMV_RESULT_CODE_V2_CVM_AIP_NOT_SUPPORTED = 0X6002,
    EMV_RESULT_CODE_V2_CVM_TAG_8E_MISSING = 0X6003,
    EMV_RESULT_CODE_V2_CVM_TAG_8E_FORMAT_ERROR = 0X6004,
    EMV_RESULT_CODE_V2_CVM_CODE_IS_NOT_SUPPORTED = 0X6005,
    EMV_RESULT_CODE_V2_CVM_COND_CODE_IS_NOT_SUPPORTED = 0X6006,
    EMV_RESULT_CODE_V2_CVM_NO_MORE = 0X6007,
    EMV_RESULT_CODE_V2_PIN_BYPASSED_BEFORE = 0X6008
} EMV_RESULT_CODE_V2_Types;
```

```
typedef enum{
    EMV_AUTHORIZATION_RESULT_ACCEPTED = 0X00,
    EMV_AUTHORIZATION_RESULT_UNABLE_TO_GO_ONLINE = 0X01,
    EMV_AUTHORIZATION_RESULT_TECHNICAL_ISSUE = 0X02,
    EMV_AUTHORIZATION_RESULT_DECLINED = 0X03,
    EMV_AUTHORIZATION_RESULT_ISSUER_REFERAL = 0X04
} EMV_AUTHORIZATION_RESULT;
```

Chapter 10

EMV Tag Reference

Tag	Description
42	Issuer Identification Number (IIN)
4F	Application Identifier (ADF Name)
50	Application Label
52	Command to perform
56	Track 1 Data
57	Track 2 Equivalent Data
5A	Application Primary Account Number (PAN)
5D	Deleted (see 9D)
5F20	Cardholder Name
5F24	Application Expiration Date
5F28	Issuer Country Code
5F2A	Transaction Currency Code (Default: 08 40)
5F2D	Language Preference
5F30	Service Code
5F34	Application Primary Account Number (PAN) Sequence Number (PSN)
5F36	Transaction Currency Exponent
5F3C	Transaction Reference Currency Code
5F3D	Transaction Reference Currency Exponent
5F50	Issuer URL
5F53	International Bank Account Number (IBAN)
5F54	Bank Identifier Code (BIC)
5F55	Issuer Country Code (alpha2 format)
5F56	Issuer Country Code (alpha3 format)
5F57	Account Type Selection
6F	File Control Information (FCI) Template
61	Application Template
62	File Control Parameters (FCP) Template
70	READ RECORD Response Message Template
71	Issuer Script Template 1
72	Issuer Script Template 2
73	Directory Discretionary Template
77	Response Message Template Format 2
80	Response Message Template Format 1
81	Amount, Authorised (Binary)
82	Application Interchange Profile (AIP)

Tag	Description
83	Command Template
84	Dedicated File (DF) Name
86	Issuer Script Command
87	Application Priority Indicator
88	Short File Identifier (SFI)
89	Authorisation Code
8A	Authorization Response Code
8A	Authorisation Response Code (ARC)
8C	Card Risk Management Data Object List 1 (CDOL1)
8D	Card Risk Management Data Object List 2 (CDOL2)
8E	Cardholder Verification Method (CVM) List
8F	Certification Authority Public Key Index (PKI)
90	Issuer Public Key Certificate
91	Issuer Authentication Data
92	Issuer Public Key Remainder
93	Signed Application Data
94	Application File Locator (AFL)
95	Terminal Verification Results (TVR)
97	Transaction Certificate Data Object List (TDOL)
98	Transaction Certificate (TC) Hash Value
99	Transaction Personal Identification Number (PIN) Data
99	Transaction Personal Identification Number (PIN) Data
98	Transaction Certificate (TC) Hash Value
9A	Transaction Date (YYMMDD)
9A	Transaction Date
9B	Transaction Status Information
9B	Transaction Status Information
9C	Transaction Type
9C	Transaction Type
9D	Directory Definition File (DDF) Name
9F01	Acquirer Identifier
9F02	Amount, Authorized (Numeric)
9F03	Amount, Other (Numeric)
9F04	Amount, Other (Binary)
9F05	Application Discretionary Data
9F06	Application Identifier (AID) – terminal
9F07	Application Usage Control (AUC)
9F08	Application Version Number
9F09	Application Version Number (Default: 00 02)
9F0B	Cardholder Name Extended
9F0D	Issuer Action Code - Default
9F0E	Issuer Action Code - Denial
9F0F	Issuer Action Code - Online
9F10	Issuer Application Data (IAD)
9F11	Issuer Code Table Index
9F12	Application Preferred Name
9F13	Last Online Application Transaction Counter (ATC) Register
9F14	Lower Consecutive Offline Limit
9F15	Merchant Category Code

Tag	Description
9F16	Merchant Identifier
9F17	Personal Identification Number (PIN) Try Counter
9F18	Issuer Script Identifier
9F19	Deleted (see 9F49)
9F1A	Terminal Country Code
9F1B	Terminal Floor Limit
9F1C	Terminal Identification
9F1D	Terminal Risk Management Data
9F1E	Interface Device (IFD) Serial Number
9F1F	Track 1 Discretionary Data
9F20	Track 2 Discretionary Data
9F21	Transaction Time (HHMMSS)
9F22	Certification Authority Public Key Index
9F23	Upper Consecutive Offline Limit
9F26	Application Cryptogram (AC)
9F27	Cryptogram Information Data (CID)
9F29	Extended Selection
9F2A	Kernel Identifier
9F2D	Integrated Circuit Card (ICC) PIN Encipherment Public Key Certificate
9F2E	Integrated Circuit Card (ICC) PIN Encipherment Public Key Exponent
9F2F	Integrated Circuit Card (ICC) PIN Encipherment Public Key Remainder
9F32	Issuer Public Key Exponent
9F33	Terminal Capabilities (see below)
9F34	Cardholder Verification Method (CVM) Results
9F35	Terminal Type (see below)
9F36	Application Transaction Counter (ATC)
9F37	Unpredictable Number
9F38	Processing Options Data Object List (PDOL)
9F39	POS Entry Mode (Default: 07)
9F3A	Amount, Reference Currency
9F3B	Application Reference Currency
9F3C	Transaction Reference Currency Code
9F3D	Transaction Reference Currency Exponent
9F40	Additional Terminal Capabilities (see below)
9F41	Transaction Sequence Counter
9F42	Application Currency Code
9F43	Application Reference Currency Exponent
9F44	Application Currency Exponent
9F45	Data Authentication Code
9F46	Integrated Circuit Card (ICC) Public Key Certificate
9F47	Integrated Circuit Card (ICC) Public Key Exponent
9F48	Integrated Circuit Card (ICC) Public Key Remainder
9F49	Dynamic Data Authentication Data Object List (DDOL)
9F4A	Static Data Authentication Tag List (SDA)
9F4B	Signed Dynamic Application Data (SDAD)
9F4C	ICC Dynamic Number
9F4D	Log Entry
9F4E	Merchant Name and Location
9F4E	Merchant Name and Location

Tag	Description
9F4F	Log Format
9F50	Offline Accumulator Balance
9F51	Application Currency Code
9F52	Application Default Action (ADA)
9F53	Transaction Category Code
9F54	DS ODS Card
9F55	Geographic Indicator
9F56	Issuer Authentication Indicator
9F57	Issuer Country Code
9F58	Consecutive Transaction Counter Limit (CTCL)
9F59	Consecutive Transaction Counter Upper Limit (CTCUL)
9F5A	Application Program Identifier (Program ID)
9F5B	Issuer Script Results
9F5C	Magstripe Data Object List (MDOL)
9F5D	Available Offline Spending Amount (AOSA)
9F5D	Application Capabilities Information (ACI)
9F5E	Consecutive Transaction International Upper Limit (CTIUL)
9F5E	DS ID
9F5F	DS Slot Availability
9F60	CVC3 (Track1)
9F61	CVC3 (Track2)
9F62	PCVC3 (Track1)
9F64	NATC (Track1)
9F65	PCVC3 (Track2)
9F66	PUNATC (Track2)
9F67	NATC (Track2)
9F68	Card Additional Processes
9F69	UDOL
9F6A	Unpredictable Number (Numeric)
9F6B	Track 2 Data
9F6C	Card Transaction Qualifiers (CTQ)
9F6D	Mag-stripe Application Version Number (Reader)
9F6E	Third Party Data
9F6E	Terminal Transaction Capabilities
9F6F	DS Slot Management Control
9F70	Protected Data Envelope 1
9F71	Protected Data Envelope 2
9F72	Protected Data Envelope 3
9F73	Protected Data Envelope 4
9F74	Protected Data Envelope 5
9F75	Unprotected Data Envelope 1
9F76	Unprotected Data Envelope 2
9F77	Unprotected Data Envelope 3
9F78	Unprotected Data Envelope 4
9F79	Unprotected Data Envelope 5
9F7A	VLP Terminal Support Indicator
9F7B	VLP Terminal Transaction Limit
9F7C	Customer Exclusive Data (CED)
9F7D	DS Summary 1

Tag	Description
9F7F	DS Unpredictable Number
A5	File Control Information (FCI) Proprietary Template
BF0C	File Control Information (FCI) Issuer Discretionary Data
BF50	Visa Fleet - CDO
BF60	Integrated Data Storage Record Update Template
C3	Card issuer action code -decline
C4	Card issuer action code -default
C5	Card issuer action code online
C6	PIN Try Limit
C7	CDOL 1 Related Data Length
C8	Card risk management country code
C9	Card risk management currency code
CA	Lower cummulative offline transaction amount
CB	Upper cumulative offline transaction amount
CD	Card Issuer Action Code (PayPass) – Default
CE	Card Issuer Action Code (PayPass) – Online
CF	Card Issuer Action Code (PayPass) – Decline
D1	Currency conversion table
D2	Integrated Data Storage Directory (IDSD)
D3	Additional check table
D5	Application Control
D6	Default ARPC response code
D7	Application Control (PayPass)
D8	AIP (PayPass)
D9	AFL (PayPass)
DA	Static CVC3-TRACK1
DB	Static CVC3-TRACK2
DC	IVCVC3-TRACK1
DD	IVCVC3-TRACK2
DF01	ApplePay VAS Protocol
DF02	ApplePay VAS Failure Report
DF10	Terminal Languages Supported
DF10	Multi Language (Default: “enfr”)
DF11	Enable Transaction Logging
DF13	Terminal Action Code - Default
DF14	Terminal Action Code - Denial
DF15	Terminal Action Code - Online
DF17	Threshold Value for Biased Random Selection
DF18	Target Percentage to be Used for Random Selection
DF19	Maximum Target Percentage to be used for Biased Random Selection
DF1F	Last 4 digits of Primary Account Number (PAN)
DF21	Issuer Script Results
DF22	Force Online (1-Enable, 0-Disable)
DF25	Default DDOL (1-Enable, 0-Disable)
DF26	Revocation List Support (Default: Enable - 1)
DF27	Exception File Support (Default: Disable - 0)
DF28	Default TDOL
DF29	Terminal Capabilities - CVM Required
DF2A	Threshold Value for Biased Random Selection (Interac)

Tag	Description
DF2B	Maximum Target Percentage for Biased Random Selection (Interac)
DF2C	Target Percentage for Random Selection (Interac)
DF30	Track Data Source
DF31	DD Card Track 1
DF32	DD Card Track 2
DF33	Interac Receipt Required
DF34	TTK Customer - Firmware Version
DF40	Message to be displayed by EMV Kernel on “PIN Try Limit Exceeded” condition
DF41	Message to be displayed by EMV Kernel on “Last PIN Try” condition
DF42	Message to be displayed by EMV Kernel on “Please Try Again” condition
DF43	Message to be displayed by EMV Kernel on “Call Your Bank” condition
DF45	GMEDS Secret Keys
DF46	GMAD MIDs
DF47	ISIS Read Cmd Data
DF48	ISIS Write Data
DF49	ISIS Transaction Data
DF4A	TTK Customer - Current KSN of Data encryption Key
DF4B	TTK Customer - MSR all track data
DF4C	TTK Customer - Masked PAN
DF4D	TTK Customer - Additional POS Info
DF4E	Polling Options
DF4F	TTK Customer - Fallback Reason
DF50	Special Flow
DF51	Amex Terminal Capability
DF52	Transaction CVM
DF55	RID
DF56	Activate Trans for DESFireViVOComm Flows
DF57	Reader Primary Language
DF57	2nd usage: Remaining Candidates
DF58	Reader Secondary Language
DF5A	TLV Exclusion List
DF5B	Terminal Entry Capability
DF5C	RF Deactivate Period
DF5D	D-PAS Issuer Script Response status
DF5E	Transaction Timing Information
DF5F	Encrypted PAN for remote PIN Pad
DF60	Product ID
DF61	Processor ID
DF61	CVMRequiredLimit_JCBScheme
DF62	Main Firmware Build ID
DF63	CB Enhanced DDA Indicator (same block as DF03)
DF64	CB Wave 2 CVM Requirements (same block as DF04)
DF65	Build ID Num (Cxx)
DF65	CB Display Offline Funds Indicator (same block as DF05)
DF65	Serial heartbeat Required
DF66	SVN Number
DF66	CB Terminal Type (same block as 9F35)
DF66	Display Unsupported Card
DF68	Enable/Disable STOP command processing
DF69	ConfigureProprietaryTags

Tag	Description
DF6A	Enable/Disable Comm Error Recovery
DF6C	Cubic FTP Phase 2 Mode Options
DF6D	Cubic Mode 3 Match AID
DF6E	Cubic Fixed Fare Amounts
DF6F	Cubic Timestamp Data
DF70	Loyalty Program ID
DF70	Generic Name String
DF71	Value Added Tax 1
DF71	Generic Numeric
DF72	Value Added Tax 2
DF72	Generic Specification String
DF73	Merchant Category Code
DF73	Generic Implementation String
DF74	Discover Optional Features
DF75	Communications Error Message Delay
DF76	TVR from GenAC
DF77	ViVOpay MSR Custom Data Output Tag
DF78	MC Timing Performance Enable
DF79	Card Disable Mask
DF7A	Card Disable Interval
DF7B	Serial Port (UART) Inter-character Timeout Period
DF7C	Auto Switch Feature
DF7D	Track Formatting Feature
DF7F	Improved Collision Detection & Media Removal Feature
DF891B	Poll Mode
DF891C	Interac Retry Limit
DFDE04	MSR Encryption Option
DFEE0C	PPSE Terminate Flags
DFEE12	KID
DFEE15	Application Selection Indicator
DFEE16	DUKPT Key or MKSK Select for Online PIN Encrypted
DFEE17	ICC Terminal Entry Mode
DFEE18	MSR Terminal Entry Mode
DFEE19	Online DOL
DFEE1A	Output data element
DFEE1B	Authorization Request data elements
DFEE1E	Contact Terminal Configuration (see below)
DFEE1F	Issuer script device limit, Range: 0∼255 (Default: 128)
DFEE20	ICC Power on detect waiting time. (Unit: Sec) (Default: 60S)
DFEE21	ICC L1 waiting time. (Unit: Sec)(Default: 10 S)
DFEE22	Driver (Menu, Get PIN, Get MSR) Timeout. (Unit: Sec) (see below)
DFEE23	MSR Track Data
DFEE24	Force Acceptance (Default: 00)
DFEE25	ICC Response Code
DFEE26	Encryption Status Information
DFEE27	MSR Control
DFEF1A	TLV available
DFEF1A	Encrypted Sensitive Tags
DFEF1A	Auto Authenticate
DFEF20	MAC option in reponse data

Tag	Description
DFEF21	BIN
DFEF22	AID
DFEF23	HMAC
DFEF24	HMAC KSN
DFEF25	Output Data Format Select
DFEF26	MSR fallback
DFEF27	Online capability
DFEF28	Disable Encrypt ON
DFEF2C	Terminal AID List
DFEF2E	Terminal Transaction Log
DFEF2F	CUP configuration
DFEF30	White List
DFEF31	Black List
DFEF32	Auto-Switch
DFEF34	Antenna Detection Switch
DFEF35	Communications Watchdog Period
DFEF36	Media Control & Status Tracking
DFEF37	Interface Select
DFEF38	Timeout for Next Command
DFEF39	Network Indicate
DFEF3A	Reader Behavior Mode
DFEF3B	Autopoll Transaction Separation Interval
DFEF40	Ascii-code encryption Tag57 TLV
DFEF41	MAC Verification Data for SRED
DFEF42	MAC Verification KSN for SRED
DFEF43	Local TZ/DST information.
DFEF44	Combination Options
DFEF45	Removal Timeout
DFEF46	ACT Pass Response DOL
DFEF47	CDA Hash Input
DFEF48	Indicate - retrieve transaction result again due to Output RAM is Not enough.
DFEF49	Outcome Parameter Set
DFE↩ F4A	User Interface Request Data
DFEF4B	MSR Equivalent Data Option
DFEF4C	MSR Equivalent Data Track Lengths
DFEF4D	MSR Equivalent Data
DFEF4E	ACT MSD Response DOL
DFEF4F	ACT Decline Response DOL
DFEF50	Terminal Interchange Profile (JCB)
DFEF51	Bypass EMV Completion Output
DFEF52	Re-FallBack times
DFEF53	Dynamic Reader Limits
DFEF54	SmartTap AID Index
DFEF55	Kernel Specific Features
DFEF56	Retry Limit
DFEF57	PPSE Terminate Flags
DFEF59	Terminal Data Setting - Default Amount
DFEF5A	Terminal Data Setting - Tags to Return
DFE↩ F5B	Mask for Tag5A

Tag	Description
DFEF5C	Mask for Tag56
DFEF5D	Mask for Tag57
DFEF5E	Mask for Tag9F6B
DFEF5F	Mask for TagFFEE13
DFEF60	Mask for TagFFEE14
DFEF61	Error Code
DFEF62	Allow MSR Swipe data from ICC Card
DFEF63	Tags To Read Yet
DFEF64	Referral Timeout
DFEF6E	USB-KB Output Data Postfix
DFEF6F	Inter-character Delay for USB-KB Interface
DFEF70	PISCES dual interface interference prevention mechanism fine-tune parameters.
DFEF71	Waiting ICC insert time
DFEF72	Pre-poll card mechanism control in ACT cmd & config setting
DFEF73	Transaction Message Type
DFEF74	Reference amplitude value
DFEF75	Reference delta value
DFEF76	Transaction Interface Type to activate
DFEF77	Timeout for waiting next command
DFEF78	EMV contact L2 display messages option
DFEF79	PIN block format (when TDES)
DFEF7A	Enable Apple Pay Check
DFEF7B	Apple Pay Status
DFEF7C	Track Bit Encoding
DFEF7D	Re-power on times
DFEF7E	Fallback response code list
FF69	ViVOpay Proprietary Tag List
FF70	Serial Finite State Machine Version
FF71	Transaction Finite State Machine Version
FF72	System Information Suite
FF73	Serial Protocol Version
FF74	Serial Protocol Suite
FF75	L1 Paypass Version
FF76	L1 LCR Version
FF77	L2 Card App Version
FF78	L2 Card App Suite
FF79	GMEDs Data
FF79	User Experience Version
FF7A	User Experience Suite
FF7B	ViVOtech Proprietary Suite
FF7C	VIUDS Scheme IDs Supported
FF7D	VIUDS Scheme ID Selection Criteria
FFE0	Registered Application Provider Identifier (RID)
FFE1	Partial Selection Allowed
FFE2	Application Flow
FFE3	Selection Features - GR 1.2.10
FFE4	Group Number / Fallback Group
FFE5	Max AID Length
FFE6	AID Disabled

Tag	Description
FFE7	Interface Support
FFE8	Exclude from Processing
FFE9	Kernel ID Transaction Type Group List
FFEA	Default Kernel ID
FFEE01	ViVOpay TLV Group Tag
FFEE02	ViVOpay Pre-PPSE Special Flow Group Tag
FFEE03	ViVOpay Post-PPSE Special Flow Group Tag
FFEE04	M/Chip3 Intermediate Message Data
FFEE05	M/Chip3 Intermediate Message Marker
FFEE06	ApplePay VAS Container
FFEE07	Encrypted Sensitive Tags
FFEE08	Masked Tags
FFEE0A	BIN Range
FFEE0B	AID Range
FFEE0C	White List
FFEE10	ViVOpay MChip Group Tag
FFEE11	ViVOpay Discover Group Tag
FFEE12	KID
FFEE12	Cash Reader Risk Record
FFEE13	Track 1 Data
FFEE13	Cashback Reader Risk Record
FFEE14	Track 2 Data
FFEE14	DRL Record 1
FFEE15	DRL Record 2
FFEE16	DRL Record 3
FFEE17	DRL Record 4
FFEE18	Tags To Write Yet Before GenAC
FFEE19	Tags To Write Yet After GenAC
FFEE1A	Terminal App DET Data
FFEE1C	Unpredictable Number Range
FFEE1D	Sensitive Data Mask
FFE↩ E1E	Group 0 Initialize Flag
FFE↩ E1F	Error Code Table
FFEE20	Restart Deactivation Time
FFF0	Specific Features Switch
FFF1	Terminal Contactless Transaction Limit
FFF2	Terminal IFD
FFF3	Application Capability
FFF4	Visa Reader Risk Flags
FFF6	Torn Transaction Log Clean Interval (minutes)
FFF7	Burst Mode
FFF8	UI Scheme
FFF9	LCD Font Size
FFFA	LCD Delay Time
FFFB	Language Option for LCD
FFFC	Force MagStripe

9F33 Terminal Capabilities

```
Byte 1
```

```
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Manual key entry
x   1   x   x   x   x   x   x   Magnetic stripe
x   x   1   x   x   x   x   x   IC with contacts
x   x   x   0   x   x   x   x   RFU
x   x   x   x   0   x   x   x   RFU
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   x   0   RFU

Byte 2
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Plaintext PIN for IC verification
x   1   x   x   x   x   X   x   Enciphered PIN for online verification
x   x   1   x   x   x   X   x   Signature(paper)
x   x   x   1   x   x   X   x   Enciphered PIN for offline verification
x   x   x   x   1   x   X   x   No CVM Required
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   X   0   RFU

Byte 3
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   SDA
X   1   x   x   x   x   x   x   DDA
X   x   1   x   x   x   x   x   Card capture
x   x   x   0   x   x   x   x   RFU
x   x   x   x   1   x   x   X   CDA
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   0   x   RFU
x   x   x   x   X   X   x   0   RFU
```

9F40 Additional Terminal Capabilities

```
Byte 1
b1  b2  b3  b4  b5  b6  b7  b8  Meaning
1   x   x   x   x   x   x   x   Cash
x   1   x   x   x   x   x   x   Goods
x   x   1   x   x   x   x   x   Services
x   x   x   1   x   x   x   x   Cashback
x   x   x   x   1   x   x   x   Inquiry
x   x   x   x   x   1   x   x   Transfer
x   x   x   x   x   x   1   x   Payment
x   x   x   x   x   x   x   1   Administrative

Byte 2
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Cash Deposit
x   0   x   x   x   x   x   x   RFU
x   x   0   x   x   x   x   x   RFU
x   x   x   0   x   x   x   x   RFU
x   x   x   x   0   x   x   x   RFU
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   x   0   RFU

Byte 3
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Numeric keys
x   1   x   x   x   x   x   x   Alphabetic and special characters keys
x   x   1   x   x   x   x   x   Command keys
x   x   x   1   x   x   x   x   Function Keys
x   x   x   x   0   x   x   x   RFU
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   x   0   RFU

Byte 4
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Print, attendant
x   1   x   x   x   x   x   x   Print, cardholder
x   x   1   x   x   x   x   x   Display, attendant
x   x   x   1   x   x   x   x   Display, cardholder
x   x   x   x   0   x   x   x   RFU
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   1   x   Code table 10
x   x   x   x   x   x   x   1   Code table 9

Byte 5
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Code table 8
x   1   x   x   x   x   x   x   Code table 7
x   x   1   x   x   x   x   x   Code table 6
x   x   x   1   x   x   x   x   Code table 5
x   x   x   x   1   x   x   x   Code table 4
x   x   x   x   x   1   x   x   Code table 3
```

```
x   x   x   x   x   x   1   x   Code table 2
x   x   x   x   x   x   x   1   Code table 1
```

9F35 Terminal Type

Environment	Financial Institution	Merchant	Cardholder
Attended			
Online only	11	21	
Offline with online capability	12	22	
Offline only	13	23	
Unattended			
Online only	14	24	34
Offline with online capability	15	25	35
Offline only	16	26	36

DFEE1E Contact Terminal Configuration (Default: F0 DC 3C F0 C2 9E 94 00)

```
Byte 1
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Key Pad support
x   1   x   x   x   x   x   x   LCD support
x   x   1   x   x   x   x   x   PIN Pad support
x   x   x   1   x   x   x   x   Print Support
x   x   x   x   0   x   x   x   RFU
x   x   x   x   x   0   x   x   RFU
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   X   0   RFU

Byte 2
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   PSE support
x   1   x   x   x   x   x   x   Cardholder confirmation
x   x   1   x   x   x   x   x   Preferred display order
x   x   x   1   x   x   x   x   Multi language
x   x   x   x   1   x   x   x   EMV language selection method
x   x   x   x   x   1   x   x   Default DDOL
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   x   0   RFU

Byte 3
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
0   x   x   x   x   x   x   x   RFU
(Revocation of Issuer Public Key Certificate (DF26))
x   1   x   x   x   x   x   x   Manual action when CA PK loading fails
x   x   1   x   x   x   x   x   CA PK verified with check sum
x   x   x   1   x   x   x   x   Bypass PIN Entry
x   x   x   x   1   x   x   x   Subsequent bypass PIN Entry
x   x   x   x   x   1   x   x   Get data for pin try counter
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   x   0   RFU

Byte 4
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Amount before CVM processing
x   1   x   x   x   x   x   x   Floor limit checking
x   x   1   x   x   x   x   x   Random transaction selection
x   x   x   1   x   x   x   x   Velocity checking
x   x   x   x   0   x   x   x   RFU
(Transaction Log (DF11))
x   x   x   x   x   0   x   x   RFU
(Exception File (DF27))
x   x   x   x   x   x   0   x   RFU
x   x   x   x   x   x   x   0   RFU

Byte 5
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   X   Terminal action code support
x   1   x   x   x   x   x   x   Terminal action code can be change
x   x   1   x   x   x   x   x   Terminal action code can be deleted or disable
x   x   x   1   x   x   x   x   Default Action code processing before 1st GAC
x   x   x   x   1   x   x   x   Default Action code processing after 1st GAC
x   x   x   x   x   1   x   x   TAC/IAC default process when unable to go online (Skipped)
x   x   x   x   x   x   1   x   TAC/IAC default process when unable to go online (Normal)
x   x   x   x   x   x   x   0   RFU
```

```
Byte 6
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   Forced Online support
x   1   x   x   x   x   x   x   Forced acceptance support
x   x   1   x   x   x   x   x   Advices support
x   x   x   1   x   x   x   x   Issuer referrals support
X   x   x   x   1   x   x   x   Batch data capture
x   x   x   x   x   1   x   x   Online data capture
X   x   x   x   x   x   1   x   Default TDOL
X   x   x   x   x   x   x   0   RFU


Byte 7
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
1   x   x   x   x   x   x   x   amount and pin entered on the same keypad
x   1   x   x   x   x   x   x   ICC/Magstripe reader combined
x   x   1   x   x   x   x   x   Magstripe read first
x   x   x   1   x   x   x   x   Support account type selection
x   x   x   x   1   x   x   x   On fly script processing
x   x   x   x   x   1   x   x   Internal date management
x   x   x   x   x   x   1   x   Reversal Mode
(1)Unable go online
(2) ARC Error
0: (3) Online Approved but reader not approved.
1: (3) Online Approved but card response AAC.
x   x   x   x   x   x   x   0   RFU

Byte 8
b8  b7  b6  b5  b4  b3  b2  b1  Meaning
x   x   x   x   x   x   x   x   RFU
```

DFEE22 Driver (Menu, Get PIN, Get MSR) Timeout. (Unit: Sec)

```
Byte1: Timeout for Menu. (Default: 30 S)
Byte2: Timeout for Get PIN. (Default: 60 S)
Byte3: Timeout for Get MSR. (Default: 60 S)
```

Chapter 11

LCD Foreign Language Mapping Table

ID	Message ID	English	French	Spanish	Chinese
0	MSG_NULL	-	-	-	-
1	MSG_AMOUNT	AMOUNT	MONTANT	CANTIDAD	金
2	MSG_AMOUNT_OK	AMOUNT OK?	MONTANT OK	MONTO CORRECTO?	确定金
3	MSG_APPROVED	APPROVED	APPROUVE	APROVADO	通
4	MSG_CALL_YOUR_BANK	CALL YOUR BANK	APPE VOTRE BANQE	LLAME A SU BANCO	系您的行
5	MSG_CANCEL_OR_ENTER	CANCEL OR ENTER	ANNULE OU ENTRER	CANCEL O ENTRAR	取消或确定
6	MSG_CARD_ERROR	CARD ERROR	ERREUR CARTE	ERROR DE TARJETA	卡
7	MSG_DECLINED	DECLINED	REFUSE	DECLINADO	卡被拒
8	MSG_ENTER_AMOUNT	ENTER AMOUNT	ENTRER MONTANT	INGRESE MONTO	入金
9	MSG_ENTER_PIN	ENTER PIN:	ENTRER PIN:	ENTRAR NPI:	入密
10	MSG_INCORRECT_PIN	INCORRECT PIN	NIP INCORRECT	NPI INCORRECTO	密
11	MSG_ICC_MSR1	SWIPE OR INSERT	PASSER OU INSERT	MOVER O INSERT	刷卡或插卡
12	MSG_ICC_MSR2	CARD	CARTE	TARJETA	卡
13	MSG_INSERT_CARD	INSERT CARD	INSERT CARTE	INSERTAR TARJETA	插卡
14	MSG_USE_CHIP_READER	USE CHIP READER UTI	LECTEUR CHIP	USO CHIP LECTOR	使用芯片卡
15	MSG_NOT_ACCEPTED	NOT ACCEPTED	PAS ACCEPTE	DENEGADO	法接受
16	MSG_PIN_OK	GET PIN OK			密正确
17	MSG_PLEASE_WAIT	PLEASE WAIT...	ATTENDRE...	POR FAVOR ESPERE	等候中
18	MSG_PROCESSING_ERROR	PROCESSING ERROR	ERREUR DE TRAITE	ERROR PROCESANDO	理
19	MSG_USE_MAGSTRIPE	USE MAGSTRIPE	USAGE MAGSTRIPE	USO DE MAGSTRIPE	使用磁卡
20	MSG_TRY_AGAIN	TRY AGAIN	REESSAYER	VUELV INTENTARLO	重

ID	Message ID	English	French	Spanish	Chinese
21	MSG_ONLINE	GO ONLINE	GO LIGNE	GO LINEA	在
22	MSG_TRANSAC↩ TION_ERROR	TRANSACTION ERR	ERREUR DE TR↩ ANS	ERROR DE TRA↩ NSAC	交易
23	MSG_TERMINA↩ TE	TERMINATE	RESILIER	TERMINAR	止
24	MSG_ADVICE	ADVICE	CONSEILS	CONSEJOS	建
25	MSG_TIMEOUT	TIME OUT	TIMEOUT	TIEMPO DE ESP↩ ERA	超
26	MSG_PROCES↩ SING	PROCESSING...	PROCESSUS...	PROCESANDO...	理中。。。
27	MSG_PIN_TRY_↩ EX	PIN TRY LIMIT EX	PIN TRY DEPAS↩ SE	TRY PIN SUPER↩ ADA	密次多
28	MSG_ISSUER_↩ AUTH_FAIL	ISSUER AUTH F↩ AIL	EMETTEUR FAIL	EMISOR FALLA	与卡机构
29	MSG_CONTINU↩ E_PROCESS	CONTINUE PRO↩ CESS	CONTINUER LA	CONTINUAR PR↩ OCES	理
30	MSG_GET_PIN_↩ ERROR	GET PIN ERROR	GET PIN ERROR	OBTENER PIN E↩ RR	密
31	MSG_GET_PIN_↩ FAIL	GET PIN FAIL	GET PIN FAIL	OBTENER PIN F↩ ALL	取密
32	MSG_NOKEY_G↩ ET_PIN	NO KEY GET PIN	NO KEY GET PIN	NO CLAVE GET PIN	法入密
33	MSG_CANCELL↩ ED	CANCELLED	ANNULE	CANCELADO	取消
34	MSG_LAST_PIN↩ _TRY	LAST PIN TRY	-	-	最后一次入密

Chapter 12

Class Index

12.1 Class List

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

Chapter 13

Class Documentation

13.1 com.idtechproducts.device.OnReceiverListener.EMV_RESULT_CODE_Types Enum Reference

Public Attributes

- **EMV_RESULT_CODE_OFFLINE_APPROVED** =(0)
- **EMV_RESULT_CODE_OFFLINE_DECLINED** =(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_USE_MAGSTRIPE** =(7)
- **EMV_RESULT_CODE_TIME_OUT** =(8)
- **EMV_RESULT_CODE_TRANSACTION_SUCCESS** =(9)
- **EMV_RESULT_CODE_TERMINATE** =(10)
- **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_REQUEST_ONLINE_PIN** =(0x81)
- **EMV_RESULT_CODE_REQUEST_SIGNATURE** =(0x82)
- **EMV_RESULT_CODE_FALLBACK_TO_CONTACT** =(0x83)
- **EMV_RESULT_CODE_FALLBACK_TO_OTHER** =(0x84)
- **EMV_RESULT_CODE_REVERSAL_REQUIRED** =(0x85)
- **EMV_RESULT_CODE_ADVISE_REQUIRED** =(0x86)
- **EMV_RESULT_CODE_ADVISE_REVERSAL_REQUIRED** =(0x87)
- **EMV_RESULT_CODE_NO_ADVISE_REVERSAL_REQUIRED** =(0x88)
- **EMV_RESULT_CODE_UNABLE_TO_REACH_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_SMARTCARD_FAIL** =(0x3002)

- **EMV_RESULT_CODE_SMARTCARD_INIT_FAILED** =(0x3003)
- **EMV_RESULT_CODE_FALLBACK_SITUATION** =(0x3004)
- **EMV_RESULT_CODE_SMARTCARD_ABSENT** =(0x3005)
- **EMV_RESULT_CODE_SMARTCARD_TIMEOUT** =(0x3006)
- **EMV_RESULT_CODE_MSR_CARD_ERROR** =(0x3007)
- **EMV_RESULT_CODE_MSR_CARD_ERROR_FALLBACK** =(0x3012)
- **EMV_RESULT_CODE_TIMEOUT_FOR_WAITING_ICC_INSERT_OR_MSR_SWIPE_FALLBACK** =(0x3013)
- **EMV_RESULT_CODE_CARD_INSERTION_FAILED** =(0x3014)
- **EMV_RESULT_CODE_PARSING_TAGS_FAILED** =(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_RECEIVED** =(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_DATE_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_INCORRECT** =(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_INCORRECT** =(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_SUPPORTED** =(0x6002)
- **EMV_RESULT_CODE_CVM_TAG_8E_MISSING** =(0x6003)
- **EMV_RESULT_CODE_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)
- final int **val**

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

- Source_Android/OnReceiverListener.java

13.2 com.idtechproducts.device.IDT_NEO2 Class Reference

Public Member Functions

- IDT_NEO2 (OnReceiverListener callback, OnReceiverListenerPIN pincallback, Context context)
- boolean device_disconnectBLE ()
- int device_enableBLESearch (DEVICE_TYPE type, String name, int timeout)
- String **getBTLEDeviceAddress** ()
- IDT_NEO2 (OnReceiverListener callback, OnReceiverListenerPIN pincallback, OnReceiverListenerPIN↩ Request callback2, Context context)
- IDT_NEO2 (OnReceiverListener callback, OnReceiverListenerPIN pincallback, OnReceiverListenerPIN↩ Request callback2, OnReceiverListenerLCD lcdcallback, Context context)
- void device_setNEOGen (int gen)
- boolean device_setDeviceType (ReaderInfo.DEVICE_TYPE deviceType)
- boolean device_setDeviceType (ReaderInfo.DEVICE_TYPE deviceType, int PID)
- void setIDT_Device (FirmwareUpdateTool fwTool)
- DEVICE_TYPE device_getDeviceType ()
- void registerListen ()
- void unregisterListen ()
- void release ()
- String config_getSDKVersion ()
- String config_getXMLVersionInfo ()
- String phone_getInfoManufacture ()
- String phone_getInfoModel ()
- void log_setVerboseLoggingEnable (boolean enable)
- void log_setSaveLogEnable (boolean enable)
- int log_deleteLogs ()
- void config_setXMLFileNameWithPath (String path)
- boolean config_loadingConfigurationXMLFile (boolean updateAutomatically)
- boolean device_connectWithProfile (StructConfigParameters profile)
- void device_ConnectWithoutValidation (boolean noValidate)
- int device_pollForToken (int timeout, ResDataStruct respData)
- boolean device_connect ()
- boolean device_isConnected ()
- int emv_callbackResponsePIN (int mode, byte[] KSN, byte[] PIN)
- void device_setSymmetric_RKI_URL (String srki_url)
- int device_startRKI ()
- int device_startRKI (boolean isDemo)
- int device_startRKI (String Key, boolean isDemo)
- int device_startRKI (String Key)
- int device_startTransaction (double amount, double amtOther, int type, final int timeout, byte[] tags)
- int device_startTransaction (double amount, double amtOther, int type, final int timeout, byte[] tags, boolean isFastEMV)
- int device_cancelTransaction ()
- int device_resetTransaction ()
- int device_getFirmwareVersion (StringBuilder version)
- int device_getDeviceTreeVersion (StringBuilder version)
- int device_pingDevice ()
- int device_setBluetoothParameters (String name, byte[] oldPassword, byte[] newPassword)
- int device_setSource (byte[] data)
- int device_getSource (byte[] data)
- int device_readFileFromSD (String directoryName, String fileName, int startingOffset, int numBytes, Res↩ DataStruct respData)
- int config_getSerialNumber (StringBuilder serialNumber)
- int config_getModelNumber (StringBuilder modNumber)

- int device_getKSN (byte keyNameIndex, byte[] keySlot, ResDataStruct resKSN)
- int device_setMerchantRecord (int index, boolean enabled, String merchantID, String merchantURL)
- int device_getMerchantRecord (int index, ResDataStruct respData)
- String device_getResponseCodeString (int errorCode)
- int device_sendDataCommand (String cmd, boolean calcLRC, String data, ResDataStruct respData)
- int device_sendDataCommand (String cmd, boolean calcLRC, String data, ResDataStruct respData, int timeout)
- int device_updateFirmware (byte[] data, FIRMWARE_UPDATE_TYPES ft)
- int device_updateFirmware (String[] commands)
- int device_getTransactionResults (IDTMSRData cardData)
- int device_setBurstMode (byte mode)
- int device_setPollMode (byte mode)
- int device_controlUserInterface (byte[] values)
- int device_getRTCDateTime (byte[] dateTime)
- int device_setRTCDateTime (byte[] dateTime)
- int device_loadCertCA (byte certType, byte[] CACertDate)
- int device_rrcUninstallApp (String appName)
- int device_rrcInstallApp (String appName)
- int device_rrcRunApp (String appName)
- int device_rrcDownloadApp (String zipFileName, String appName)
- int device_rrcConnect ()
- int device_rrcDisconnect ()
- int device_getBatteryPercentage (StringBuilder percentage)
- int icc_getICCReaderStatus (ICCReaderStatusStruct ICCStatus)
- int icc_powerOnICC (ResDataStruct atrPPS)
- int icc_passthroughOnICC ()
- int icc_passthroughOffICC ()
- int icc_powerOffICC (ResDataStruct respData)
- int icc_exchangeAPDU (byte[] dataAPDU, ResDataStruct response)
- int emv_getEMVKernelVersion (StringBuilder version)
- int emv_getEMVKernelCheckValue (ResDataStruct respData)
- int emv_getEMVConfigurationCheckValue (ResDataStruct respData)
- int emv_retrieveApplicationData (String aid, ResDataStruct respData)
- int emv_removeApplicationData (String aid, ResDataStruct respData)
- int emv_setApplicationData (String name, byte[] tlv, ResDataStruct respData)
- int emv_retrieveTerminalData (ResDataStruct respData)
- int emv_removeTerminalData (ResDataStruct respData)
- int emv_setTerminalData (byte[] TLV, ResDataStruct respData)
- int emv_retrieveAidList (ResDataStruct respData)
- int emv_retrieveCAPK (byte[] capk, ResDataStruct respData)
- int emv_removeCAPK (byte[] capk, ResDataStruct respData)
- int emv_setCAPK (byte[] key, ResDataStruct respData)
- int emv_retrieveCAPKList (ResDataStruct respData)
- int emv_retrieveCRL (ResDataStruct respData)
- int emv_removeCRL (byte[] crlList, ResDataStruct respData)
- int emv_setCRL (byte[] crlList, ResDataStruct respData)
- int emv_startTransaction (double amount, double amtOther, int type, final int timeout, byte[] tags, boolean forceOnline)
- int emv_cancelTransaction (ResDataStruct respData)
- void emv_setTransactionParameters (double amount, double amtOther, int type, final int timeout, byte[] tags)
- void emv_lcdControlResponse (byte mode, byte selection)
- int emv_authenticateTransaction (byte[] tags)
- int emv_completeTransaction (boolean commError, byte[] authCode, byte[] iad, byte[] tlvScripts, byte[] tags)
- int emv_retrieveTransactionResult (byte[] tags, Map< String, Map< String, byte[]>> retrievedTags)
- int emv_setTerminalMajorConfiguration (int configuration)

- int device_reviewAllSetting (ResDataStruct respData)
- int msr_defaultAllSetting ()
- int msr_cancelMSRSwipe ()
- int msr_startMSRSwipe ()
- int msr_startMSRSwipe (int timeout)
- int ctls_retrieveApplicationData (String aid, ResDataStruct respData)
- int ctls_removeApplicationData (String aid, ResDataStruct respData)
- int ctls_setApplicationData (byte[] tlv, ResDataStruct respData)
- int ctls_setConfigurationGroup (byte[] TLV, ResDataStruct respData)
- int ctls_getConfigurationGroup (int group, ResDataStruct respData)
- int ctls_getAllConfigurationGroups (ResDataStruct respData)
- int ctls_removeConfigurationGroup (int group)
- int ctls_retrieveTerminalData (ResDataStruct respData)
- int ctls_setTerminalData (byte[] TLV, ResDataStruct respData)
- int ctls_retrieveAidList (ResDataStruct respData)
- int ctls_retrieveCAPK (byte[] capk, ResDataStruct respData)
- int ctls_removeCAPK (byte[] capk, ResDataStruct respData)
- int ctls_setCAPK (byte[] key, ResDataStruct respData)
- int ctls_retrieveCAPKList (ResDataStruct respData)
- int ctls_removeAllApplicationData ()
- int ctls_removeAllCAPK ()
- int ctls_startTransaction (double amount, double amtOther, int type, final int timeout, byte[] tags)
- int ctls_cancelTransaction ()
- int pin_cancelPINEntry ()
- int pin_displayMessageGetEncryptedPIN (byte PANKeyType, byte[] PAN, byte PINMaxLen, byte PINMinLen, byte[] LCDMsg, ResDataStruct respData)
- int pin_getFunctionKey (ResDataStruct respData)
- int pin_displayMessageGetNumericKey (byte DisplayFlag, byte KeyMaxLen, byte KeyMinLen, byte[] Text↩ DisplayMsg, byte[] DisplayMsgSig, ResDataStruct respData)
- int pin_displayMessageGetAmount (byte DisplayFlag, byte KeyMaxLen, byte KeyMinLen, byte[] TextDisplay↩ Msg, byte[] DisplayMsgSig, ResDataStruct respData)
- int felica_authentication (byte[] key)
- int felica_readWithMac (int blockCnt, byte[] blockList, ResDataStruct respData)
- int felica_writeWithMac (byte blockNum, byte[] blockData)
- int felica_read (byte[] serviceCodeList, int blockCnt, byte[] blockList, ResDataStruct respData)
- int felica_write (byte[] serviceCodeList, int blockCnt, byte[] blockList, byte[] blockData, ResDataStruct resp↩ Data)
- int felica_poll (byte[] systemCode, ResDataStruct respData)
- int felica_requestService (byte[] nodeCode, ResDataStruct respData)
- int felica_SendCommand (byte command[], ResDataStruct respData)
- int lcd_clearDisplay ()
- int lcd_createScreen (String screenName, IDTLCDData returnItem)
- int lcd_getActiveScreen (IDTLCDData returnItem)
- int lcd_destroyScreen (String screenName)
- int lcd_showScreen (String screenName)
- int lcd_cloneScreen (String screenName, String cloneName, IDTLCDData returnItem)
- int lcd_updateLabel (String screenName, String objectName, String label)
- int lcd_updateColor (String screenName, String objectName, byte[] color)
- int lcd_updatePosition (String screenName, String objectName, byte alignment, int new_xCord, int new_y↩ Cord)
- int lcd_removeItem (String screenName, String objectName)
- int lcd_storeScreenInfo ()
- int lcd_loadScreenInfo ()
- int lcd_clearScreenInfo ()
- int lcd_getAllScreens (List< IDTLCDData > IDTScreenInfos)

- int lcd_getAllObjects (String screenName, List< IDTLCDData > IDTScreenInfos)
- int lcd_queryScreenbyName (String screenName, IDTLCDData returnItem)
- int lcd_queryObjectbyName (String objectName, List< IDTLCDData > IDTScreenInfos)
- int lcd_queryScreenbyID (int screenID, IDTLCDData returnItem)
- int lcd_queryObjectbyID (int objectID, List< IDTLCDData > IDTScreenInfos)
- int lcd_linkUIWithTransactionMessageId (byte MessageId, String screenName)
- int lcd_setBacklight (boolean isBacklightOn, byte backlightVal)
- int lcd_getButtonEvent (IDTLCDData returnItem)
- int lcd_addButton (String screenName, String buttonName, byte type, byte alignment, int xCord, int yCord, String label, IDTLCDData returnItem)
- int lcd_addEthernet (String screenName, String objectName, byte alignment, int xCord, int yCord, IDTLCD↩ Data returnItem)
- int lcd_addLED (String screenName, String objectName, byte alignment, int xCord, int yCord, IDTLCDData returnItem, byte[] LED)
- int lcd_addText (String screenName, String objectName, byte alignment, int xCord, int yCord, int width, int height, byte fontID, byte[] color, String label, IDTLCDData returnItem)
- int lcd_addImage (String screenName, String objectName, byte alignment, int xCord, int yCord, String filename, IDTLCDData returnItem)
- int lcd_addVideo (String screenName, String objectName, byte alignment, int xCord, int yCord, String filename, IDTLCDData returnItem)
- int lcd_addExtVideo (String screenName, String objectName, byte alignment, int xCord, int yCord, byte loop, byte numVideos, String filenames, IDTLCDData returnItem)
- String createFastEMVData (IDTEMVData emvData)
- int forwardTransaction (WorldpayListener callback, String forwardID, String password, boolean bypass↩ Processing)

Static Public Member Functions

- static IDT_Device getSDKInstance ()
- static void useUSBIntentFilter ()
- static IDT_Device **getIDT_Device** ()
- static boolean device_setPackageDownloadDelay (int delay)
- static int device_getPackageDownloadDelay ()
- static void emv_allowFallback (boolean allow)
- static void emv_setAutoAuthenticateTransaction (boolean auto)
- static boolean emv_getAutoAuthenticateTransaction ()
- static void emv_setAutoCompleteTransaction (boolean auto)
- static boolean emv_getAutoCompleteTransaction ()

13.2.1 Constructor & Destructor Documentation

13.2.1.1 IDT_NEO2() [1/3]

```
com.idtechproducts.device.IDT_NEO2.IDT_NEO2 (
            OnReceiverListener callback,
            OnReceiverListenerPIN pincallback,
            Context context )
```

It is the constructor of the main class IDT_VP3600. When it is called, the SDK will create the Instance for IDT_V↩ P3600 device. The interface OnReceiverListner needs to be implemented in the application.

Parameters

callback	OnReceiverListener callback
pincallback	OnReceiverListenerPIN pincallback
context	Application context

13.2.1.2 IDT_NEO2() [2/3]

```
com.idtechproducts.device.IDT_NEO2.IDT_NEO2 (
            OnReceiverListener callback,
            OnReceiverListenerPIN pincallback,
            OnReceiverListenerPINRequest callback2,
            Context context )
```

It is the constructor of the main class IDT_VP3600. When it is called, the SDK will create the Instance for IDT_V↩ P3600 device. The interface OnReceiverListner needs to be implemented in the application.

Parameters

callback	OnReceiverListener callback
pincallback	OnReceiverListenerPIN pincallback
pincallback	OnReceiverListenerPINRequest pin request callback
context	Application context

13.2.1.3 IDT_NEO2() [3/3]

```
com.idtechproducts.device.IDT_NEO2.IDT_NEO2 (
            OnReceiverListener callback,
            OnReceiverListenerPIN pincallback,
            OnReceiverListenerPINRequest callback2,
            OnReceiverListenerLCD lcdcallback,
            Context context )
```

It is the constructor of the main class IDT_VP3600. When it is called, the SDK will create the Instance for IDT_V↩ P3600 device. The interface OnReceiverListner needs to be implemented in the application.

Parameters

callback	OnReceiverListener callback
pincallback	OnReceiverListenerPIN pincallback
pincallback	OnReceiverListenerPINRequest pin request callback
lcdcallback	OnReceiverListenerLCD lcdcallback
context	Application context

13.2.2 Member Function Documentation

13.2.2.1 config_getModelNumber()

```
int com.idtechproducts.device.IDT_NEO2.config_getModelNumber (
            StringBuilder modNumber )
```

Get the model number of device.

Parameters

modNumber	returns Model Number string.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.2 config_getSDKVersion()

```
String com.idtechproducts.device.IDT_NEO2.config_getSDKVersion ( )
```

READER CONFIG API LIST Get the version of SDK.

```
    for version string.
```

Returns

the string of the SDK version.

See also

ErrorCode

13.2.2.3 config_getSerialNumber()

```
int com.idtechproducts.device.IDT_NEO2.config_getSerialNumber (
            StringBuilder serialNumber )
```

Get the serial number of device.

Parameters

serialNumber	returns Serial Number string.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.4 config_getXMLVersionInfo()

```
String com.idtechproducts.device.IDT_NEO2.config_getXMLVersionInfo ( )
```

Get XML configuration version.

Returns

the version info.

13.2.2.5 config_loadingConfigurationXMLFile()

```
boolean com.idtechproducts.device.IDT_NEO2.config_loadingConfigurationXMLFile (
            boolean updateAutomatically )
```

Load XML Configuration File.

Parameters

updateAutomatically	

Returns

none

13.2.2.6 config_setXMLFileNameWithPath()

```
void com.idtechproducts.device.IDT_NEO2.config_setXMLFileNameWithPath (
            String path )
```

set XML Configuration File Name with the full path.

Parameters

path	XML Configuration File Name.

Returns

none

13.2.2.7 createFastEMVData()

```
String com.idtechproducts.device.IDT_NEO2.createFastEMVData (
            IDTEMVData emvData )
```

Create Fast EMV Data

At the completion of a Fast EMV Transaction, after the final card decision is returned and the IDTEMVData object is provided, sending that emvData object to this method will return string data that represents the Fast EMV data that would be returned from and IDTech FastEMV over KB protocol

Parameters

emvData	The IDTEMVData object populated with card data.

Returns

Fast EMV String data

13.2.2.8 ctls_cancelTransaction()

```
int com.idtechproducts.device.IDT_NEO2.ctls_cancelTransaction ( )
```

Cancel CTLS (or MSR) Transaction

Cancels the currently executing CTLS transaction (or MSR swipe request).

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.9 ctls_getAllConfigurationGroups()

```
int com.idtechproducts.device.IDT_NEO2.ctls_getAllConfigurationGroups (
            ResDataStruct respData )
```

Retrieve All Configuration Groups

Returns all the Configuration Groups installed on the terminal for CTLS.

Parameters

respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode. If no Application data exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.10 ctls_getConfigurationGroup()

```
int com.idtechproducts.device.IDT_NEO2.ctls_getConfigurationGroup (
            int group,
            ResDataStruct respData )
```

Get Configuration Group

Retrieves the Configuration for the specified Group.

Parameters

group	Configuration Group
respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode. If no Application data exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.11 ctls_removeAllApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.ctls_removeAllApplicationData ( )
```

Remove all Application Data

Removes all the Application Data

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.12 ctls_removeAllCAPK()

```
int com.idtechproducts.device.IDT_NEO2.ctls_removeAllCAPK ( )
```

Remove All Certificate Authority Public Key

Removes all CAPK

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.13 ctls_removeApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.ctls_removeApplicationData (
            String aid,
            ResDataStruct respData )
```

Remove Application Data

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

Parameters

aid	Aid file to remove.
respData	Status Code in ResDataStruct.statusCode. If no application data exists, status code will be 0x60. Format error status code 0x05

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.14 ctls_removeCAPK()

```
int com.idtechproducts.device.IDT_NEO2.ctls_removeCAPK (
            byte [] capk,
            ResDataStruct respData )
```

Remove Certificate Authority Public Key

Removes the CAPK as specified by the RID/Index

Parameters

capk	6 byte CAPK = 5 bytes RID + 1 byte INDEX
respData	Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.15 ctls_removeConfigurationGroup()

```
int com.idtechproducts.device.IDT_NEO2.ctls_removeConfigurationGroup (
            int group )
```

Remove Configuration Group

Removes the Configuration as specified by the Group. Must not by group 0

Parameters

group	Configuration Group

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.16 ctls_retrieveAidList()

```
int com.idtechproducts.device.IDT_NEO2.ctls_retrieveAidList (
            ResDataStruct respData )
```

Retrieve Aid List

Returns all the AID names installed on the terminal.

Parameters

respData	Array of AID string names passed back in ResDataStruct.stringArray. Status Code in ResDataStruct.statusCode. If no AIDs exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.17 ctls_retrieveApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.ctls_retrieveApplicationData (
            String aid,
            ResDataStruct respData )
```

Retrieve Application Data

Retrieves the TLV values of a provide AID.

Parameters

aid	Aid file to retrieve.
respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode. If no application data exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.18 ctls_retrieveCAPK()

```
int com.idtechproducts.device.IDT_NEO2.ctls_retrieveCAPK (
            byte [] capk,
            ResDataStruct respData )
```

Retrieve Certificate Authority Public Key

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

Parameters

capk	6 bytes CAPK = 5 bytes RID + 1 byte Index
respData	key Response returned in ResDataStruct.resData: [5 bytes RID][1 byte Index][1 byte Hash Algorithm][1 byte Encryption Algorithm][20 bytes HashValue][4 bytes Public Key Exponent][2 bytes Modulus Length][Variable bytes Modulus] Where: • Hash Algorithm: The only algorithm supported is SHA-1.The value is set to 0x01 • Encryption Algorithm: The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to 0x01. • HashValue: Which is calculated using SHA-1 over the following fields: RID & Index & Modulus & Exponent • Public Key Exponent: Actually, the real length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03), or 65537 (Format is 0x00 01 00 01) • Modulus Length: LenL LenH Indicated the length of the next field. • Modulus: This is the modulus field of the public key. Its length is specified in the field above.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.19 ctls_retrieveCAPKList()

```
int com.idtechproducts.device.IDT_NEO2.ctls_retrieveCAPKList (
            ResDataStruct respData )
```

Retrieve the Certificate Authority Public Key list

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

Parameters

respData	ResDataStruct.resData = [key1][key2]...[keyn], each key 6 bytes where key = 5 bytes RID + 1 byte index

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.20 ctls_retrieveTerminalData()

```
int com.idtechproducts.device.IDT_NEO2.ctls_retrieveTerminalData (
            ResDataStruct respData )
```

Retrieve Terminal Data

Retrieves the TLV values of a the terminal.

Parameters

respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode. If no terminal data exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.21 ctls_setApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.ctls_setApplicationData (
            byte [] tlv,
            ResDataStruct respData )
```

Set Application Data by AID

Sets the Application Data for CTLS as specified by the TLV data

Parameters

tlv	Application data in TLV format The first tag of the TLV data must be the group number (FFE4). The second tag of the TLV data must be the AID (9F06)

Example valid TLV, for Group #2, AID a0000000035010: "ffe401029f0607a0000000051010ffe10101ffe50110ffe30114ffe20106"

Parameters

respData	error code will be stored in respData.statusCode

Returns

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

See also

ErrorCode

13.2.2.22 ctls_setCAPK()

```
int com.idtechproducts.device.IDT_NEO2.ctls_setCAPK (
            byte [] key,
            ResDataStruct respData )
```

Set Certificate Authority Public Key

Sets the CAPK as specified by the CAKey structure

Parameters

key	CAKey format: [5 bytes RID][1 byte Index][1 byte Hash Algorithm][1 byte Encryption Algorithm][20 bytes HashValue][4 bytes Public Key Exponent][2 bytes Modulus Length][Variable bytes Modulus] Where: • Hash Algorithm: The only algorithm supported is SHA-1.The value is set to 0x01 • Encryption Algorithm: The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to 0x01. • HashValue: Which is calculated using SHA-1 over the following fields: RID & Index & Modulus & Exponent • Public Key Exponent: Actually, the real length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03), or 65537 (Format is 0x00 01 00 01) • Modulus Length: LenL LenH Indicated the length of the next field. • Modulus: This is the modulus field of the public key. Its length is specified in the field above.
respData	Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.23 ctls_setConfigurationGroup()

```
int com.idtechproducts.device.IDT_NEO2.ctls_setConfigurationGroup (
            byte [] TLV,
            ResDataStruct respData )
```

Set Configuration Data for AID Group

Sets the Configuration Data for CTLS as specified by the TLV data

Parameters

TLV	Configuration data in TLV format The first tag of the TLV data must be the group number (FFE4). A second tag must exist
respData	error code will be stored in respData.statusCode

Returns

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

See also

ErrorCode

13.2.2.24 ctls_setTerminalData()

```
int com.idtechproducts.device.IDT_NEO2.ctls_setTerminalData (
            byte [] TLV,
            ResDataStruct respData )
```

Set Terminal Data

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

Parameters

TLV	TerminalData configuration file.
respData	Status Code in ResDataStruct.statusCode. If Flash error, status code will be 0x62. Format error status code 0x05

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.25 ctls_startTransaction()

```
int com.idtechproducts.device.IDT_NEO2.ctls_startTransaction (
            double amount,
```

```
            double amtOther,
            int type,
            final int timeout,
            byte [] tags )
```

Start CTLS (or MSR) Transaction Request

Authorizes the CTLS (or MSR) transaction for an ICC card

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)
amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request. Passed as a string. Example, tag 9F02 with amount 0x000000000100 would be "9F0206000000000100" If tags 9F02 (amount),9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method. Note: To request tags to be included in default response, use tag DFEE1A, and specify tag list. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37 For SmartTap, pass the tag FFEE08 with the value 0200 For Apple VAS, pass the tag FFEE06 with the value 9F220201009F2604000000009F2B050100000000DF010101

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.26 device_cancelTransaction()

```
int com.idtechproducts.device.IDT_NEO2.device_cancelTransaction ( )
```

Cancel Device Transaction

Cancels the currently executing Device transaction.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.27 device_connect()

```
boolean com.idtechproducts.device.IDT_NEO2.device_connect ( )
```

Connect the device.

This will attempt to connect to reader. For audio connection: it will first attempt to set the IDTech reader to the correct baud rate compatible with the Android Audio hardware Then it will attempt to recognize the reader that is attached by polling for model information

Returns

true: success, false: fail.

13.2.2.28 device_ConnectWithoutValidation()

```
void com.idtechproducts.device.IDT_NEO2.device_ConnectWithoutValidation (
            boolean noValidate )
```

Set connection mode to connect to the device without validation

This will tell the SDK to connect to a reader with supplied device type For audio connection: it will not attempt to set the IDTech reader to the correct baud rate compatible with the Android Audio hardware and will assume the hardware baud rate is correct. It will not attempt to validate an IDTech reader is connected. It will assume an IDTech reader is connected and charged

NOTE: It is left to the integrator to make the decision to validate the reader is connected. One method is execute device_getFirmware and evaluate the response. If there is a response, then a reader from IDTech. If it is an expected response (known firmware string), then the specific reader model can be verified.

Parameters

noValidate	TRUE = no validation, FALSE = validate (normal operation)

Returns

none.

13.2.2.29 device_connectWithProfile()

```
boolean com.idtechproducts.device.IDT_NEO2.device_connectWithProfile (
            StructConfigParameters profile )
```

connect the device with Profile.

Parameters

profile	the profile is the one which is the result from Auto config.

Returns

true: success, false: fail.

13.2.2.30 device_controlUserInterface()

```
int com.idtechproducts.device.IDT_NEO2.device_controlUserInterface (
```

```
byte [] values )
```

Control User Interface

Controls the User Interface: Display, Beep, LED

Parameters

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

Returns

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

13.2.2.31 device_disconnectBLE()

```
boolean com.idtechproducts.device.IDT_NEO2.device_disconnectBLE ( )
```

Disconnect from BLE -

Will disconnect from existing BLE connection.

Returns

status: True = valid current connection can be shut down False = connection doesn't exist, or error shutting down connection.

13.2.2.32 device_enableBLESearch()

```
int com.idtechproducts.device.IDT_NEO2.device_enableBLESearch (
            DEVICE_TYPE type,
            String name,
            int timeout )
```

Searches for Bluetooth Device

Parameters

type	Device Type to search for
name	Bluetooth Device Name or MAC Address
timeout	Scanning timeout, in milliseconds

Returns

error code 0 = Success 1 = Invalid DEVICE_TYPE 2 = Bluetooth LE is not supported on this device 3 = Bluetooth LE is not available 4 = Bluetooth LE is not enabled 5 = Device not paired. Please pair first

13.2.2.33 device_getBatteryPercentage()

```
int com.idtechproducts.device.IDT_NEO2.device_getBatteryPercentage (
            StringBuilder percentage )
```

DEVICE INFO API Get the Battery Percentage of device.

Parameters

percentage	for battery percentage string.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.34 device_getDeviceTreeVersion()

```
int com.idtechproducts.device.IDT_NEO2.device_getDeviceTreeVersion (
            StringBuilder version )
```

DEVICE INFO API Get the device tree version of device.

Parameters

version	for version string.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.35 device_getDeviceType()

```
DEVICE_TYPE com.idtechproducts.device.IDT_NEO2.device_getDeviceType ( )
```

Gets type of device

13.2.2.36 device_getFirmwareVersion()

```
int com.idtechproducts.device.IDT_NEO2.device_getFirmwareVersion (
            StringBuilder version )
```

DEVICE INFO API Get the firmware version of device.

Parameters

version	for version string.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.37 device_getKSN()

```
int com.idtechproducts.device.IDT_NEO2.device_getKSN (
            byte keyNameIndex,
            byte [] keySlot,
            ResDataStruct resKSN )
```

Get the Account DUKPT Key KSN of device.

Parameters

10-byte	KSN

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode Get KSN

Get the Account DUKPT Key KSN of device.

Parameters

keyNameIndex	1 byte Key Name Index 0x14 – LCL-KEK 0x01 – Pin encryption Key 0x02 - Data encryption Key 0x05 – MAC DUKPT Key
keySlot	2 bytes Key Slot Indicate different slots of a certain Key Name Example: slot =5 (0x00 0x05), slot=300 (0x01 0x2C)
resKSN	Response returned in resKSN.resData: 1 byte Key State: 00h : Unused (Slot is supported but no key injected) 01h : Valid (A valid key is available in this slot) 02h : End of life (The key on their slot has reached end of life) FFh : Not Available (This slot is not supported) 1 byte KSN Length (optional): Value is 10 to indicate TDES PIN DUKPT Key, Value is 12 to indicate AES PIN DUKPT Key 10 or 12 bytes KSN (optional): DUKPT Key KSN (Hex) Length is 10 bytes Hex to indicate TDES PIN DUKPT Key, Length is 12 bytes Hex to indicate AES PIN DUKPT Key

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.38 device_getMerchantRecord()

```
int com.idtechproducts.device.IDT_NEO2.device_getMerchantRecord (
            int index,
            ResDataStruct respData )
```

Get Merchant Record

Sets the burst mode for the device.

Parameters

respData	response data from reader. Merchant Record Index: 1 byte enabled: 1 byte Merchant ID: 32 bytes Length of Merchant URL: 1 byte Merchant URL: 64 bytes

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.39 device_getPackageDownloadDelay()

```
static int com.idtechproducts.device.IDT_NEO2.device_getPackageDownloadDelay ( )  [static]
```

Gets the package download delay for Bluetooth Tells the SDK what delay should be used during the package download for Bluetooth

Returns

the delay time interval in ms

13.2.2.40 device_getResponseCodeString()

```
String com.idtechproducts.device.IDT_NEO2.device_getResponseCodeString (
            int errorCode )
```

Get Response Code String

Interpret a response code and return string description.

Parameters

errorCode	Error code, range 0x0000 - 0xFFFF, example 0x0300

Returns

Verbose error description

13.2.2.41 device_getRTCDateTime()

```
int com.idtechproducts.device.IDT_NEO2.device_getRTCDateTime (
            byte [] dateTime )
```

get RTC date and time of the device

Parameters

dateTime	<dateTime data>=""> is: 6 byte data for yyMMddHHmmss in hex. For example 0x171003102543 stands for 2017 Oct 3rd 10:25:43

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.42 device_getSource()

```
int com.idtechproducts.device.IDT_NEO2.device_getSource (
            byte [] data )
```

Get Source for RTC/LCD/Buzzer/LED

Get the source for RTC/LCD/Buzzer/LED on the ViVOpay reader.

Parameters

data	the 2-byte source configuration for RTC/LCD/Buzzer/LED

Data Byte1 Definition:

Bit7 Bit6 | Bit5 Bit4 | Bit3 Bit2 | Bit1 Bit0

Reserved | RTC | LCD | Buzzer

Buzzer: 00 Don't use Buzzer 01 Use Buzzer from ViVOpay reader 10 Use Buzzer from external source 11 Use Buzzer from both reader and external source

LCD: 00 Don't use LCD 01 Use LCD from ViVOpay reader 10 Use LCD from external source 11 Use LCD from both reader and external source

RTC: 00 Don't use RTC 01 Use RTC from ViVOpay reader 10 Not allowed 11 Not allowed

Data Byte2 Definition:

Bit7 Bit6 | Bit5 Bit4 | Bit3 Bit2 | Bit1 Bit0

Reserved | Reserved | Power LED | Transaction LED

Transaction LED: 00 Don't use transaction LED 01 Use transaction LED from ViVOpay reader 10 Use transaction LED from external source 11 Use transaction LED from both reader and external source

Power LED: 00 Don't use power LED 01 Use power LED from ViVOpay reader 10 Use power LED from external source 11 Use power LED from both reader and external source

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.43 device_getTransactionResults()

```
int com.idtechproducts.device.IDT_NEO2.device_getTransactionResults (
            IDTMSRData cardData )
```

DEVICE INFO API Update the firmware of device through the zip file.

Parameters

zipFilename	The zip file name that includes the file Metadata.json and the firmware fm files.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode Get Transaction Results Gets the transaction results when the reader is functioning in "Auto Poll" mode

Parameters

cardData	The transaction results

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.44 device_isConnected()

```
boolean com.idtechproducts.device.IDT_NEO2.device_isConnected ( )
```

get the status if the device connected.

Returns

true: connected, false: disconnected

13.2.2.45 device_loadCertCA()

```
int com.idtechproducts.device.IDT_NEO2.device_loadCertCA (
            byte certType,
            byte [] CACertDate )
```

Use this function to load CA Cert (Intermediate Certificate)

Parameters

certType	Application CA: 0x00 TLS CA: 0x01
CACertDate	Multi bytes CA cert data

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.46 device_pingDevice()

```
int com.idtechproducts.device.IDT_NEO2.device_pingDevice ( )
```

Ping Device

Pings the reader. If connected, returns success. Otherwise, returns timeout.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.47 device_pollForToken()

```
int com.idtechproducts.device.IDT_NEO2.device_pollForToken (
            int timeout,
            ResDataStruct respData )
```

Poll for Token

Polls for a PICC

Parameters

timeout	timeout in milliseconds, must be multiple of 10 milliseconds. 30, 120, 630, or 1150 for example.
respData	Response data will be stored in respData. 1 byte of card type, and the Serial Number (or the UID) of the PICC if available.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.48 device_readFileFromSD()

```
int com.idtechproducts.device.IDT_NEO2.device_readFileFromSD (
            String directoryName,
            String fileName,
            int startingOffset,
            int numBytes,
            ResDataStruct respData )
```

Read Out File from SD Card This command retrieves a file from the SD card of the reader.

Parameters

directoryName	Directory Name. No longer than 32 bytes. If null, root directory is listed
fileName	File name string. No longer than 32 bytes. A ASCII string terminated by 0x00.
startingOffset	Starting offset in the file to retrieve
numBytes	Number of bytes of file data to retrieve
respData	Returns file data in ResDataStruct.resData. Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.49 device_resetTransaction()

```
int com.idtechproducts.device.IDT_NEO2.device_resetTransaction ( )
```

Reset Transaction

Abruptly terminates the currently executing device transaction. Does not produce any further transaction data or notifications after executing.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.50 device_reviewAllSetting()

```
int com.idtechproducts.device.IDT_NEO2.device_reviewAllSetting (
            ResDataStruct respData )
```

Review All Configuration Settings

it returns the current values for all the parameters that can be set using the Set Configuration command. Each parameter is returned as a TLV data object.

Parameters

respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.51 device_rrcConnect()

```
int com.idtechproducts.device.IDT_NEO2.device_rrcConnect ( )
```

Use this function to allow a host to establish an RRC connection to a reader

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.52 device_rrcDisconnect()

```
int com.idtechproducts.device.IDT_NEO2.device_rrcDisconnect ( )
```

Use this function to allow a host to terminate its existing RRC connection to a reader

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.53 device_rrcDownloadApp()

```
int com.idtechproducts.device.IDT_NEO2.device_rrcDownloadApp (
            String zipFileName,
            String appName )
```

Use this function to transfer a compressed application file from a host to the reader, extracts it, and performs signature verification on its contents.

Parameters

zipFileName	The zip file name
appName	Application Name in ASCII

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.54 device_rrcInstallApp()

```
int com.idtechproducts.device.IDT_NEO2.device_rrcInstallApp (
            String appName )
```

Use this function to install an application on a reader device

Parameters

appName	Application Name in ASCII

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.55 device_rrcRunApp()

```
int com.idtechproducts.device.IDT_NEO2.device_rrcRunApp (
            String appName )
```

Use this function to run an application on a reader device

Parameters

appName	Application Name in ASCII

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.56 device_rrcUninstallApp()

```
int com.idtechproducts.device.IDT_NEO2.device_rrcUninstallApp (
            String appName )
```

Use this function to uninstall an application on a reader device

Parameters

appName	Application Name in ASCII

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.57 device_sendDataCommand() [1/2]

```
int com.idtechproducts.device.IDT_NEO2.device_sendDataCommand (
            String cmd,
            boolean calcLRC,
            String data,
            ResDataStruct respData )
```

Sends a Direct Command Sends a NEO IDG ViVOtech 2.0 command

Parameters

cmd	Two bytes command (including subCommand) as per NEO IDG Reference Guide (UniPayIII)
calcLRC	Not used for IDG devices
data	Command data (if applicable) for IDG devices
respData	Returns response ResDataStruct.resData. Status Code in ResDataStruct.statusCode

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.58 device_sendDataCommand() [2/2]

```
int com.idtechproducts.device.IDT_NEO2.device_sendDataCommand (
            String cmd,
            boolean calcLRC,
            String data,
            ResDataStruct respData,
            int timeout )
```

Sends a Direct Command Sends a NEO IDG ViVOtech 2.0 command

Parameters

cmd	Two bytes command (including subCommand) as per NEO IDG Reference Guide (UniPayIII)
calcLRC	Not used for IDG devices
data	Command data (if applicable) for IDG devices
respData	Returns response in ResDataStruct.resData. Status Code in ResDataStruct.statusCode
timeout	Command timeout in seconds

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.59 device_setBluetoothParameters()

```
int com.idtechproducts.device.IDT_NEO2.device_setBluetoothParameters (
            String name,
            byte [] oldPassword,
            byte [] newPassword )
```

Retrieves Audio Jack setting.

Parameters

respData	the response will be stored in respData.resData respData.resData[0]: baud rate of the device connected. respData.resData[1]: level option of the device output signals. respData.resData[2]: the number of prefix "55", and end with "66".

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode Set Bluetooth Parameters

Sets the Bluetooth parameters for the device.

Parameters

name	The string of the Bluetooth ID of the device
oldPassword	6 bytes of the old password
newPassword	6 bytes of the new password

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.60 device_setBurstMode()

```
int com.idtechproducts.device.IDT_NEO2.device_setBurstMode (
            byte mode )
```

Set Burst Mode

Sets the burst mode for the device.

Parameters

mode	0 = OFF, 1 = Always On, 2 = Auto Exit

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.61 device_setDeviceType() [1/2]

```
boolean com.idtechproducts.device.IDT_NEO2.device_setDeviceType (
            ReaderInfo.DEVICE_TYPE deviceType )
```

Defines connection Bluetooth

Parameters

deviceType	DEVICE_TYPE.DEVICE_NEO2_BT

13.2.2.62 device_setDeviceType() [2/2]

```
boolean com.idtechproducts.device.IDT_NEO2.device_setDeviceType (
            ReaderInfo.DEVICE_TYPE deviceType,
            int PID )
```

Defines connection USB

Parameters

deviceType	DEVICE_TYPE.DEVICE_NEO2_USB

13.2.2.63 device_setMerchantRecord()

```
int com.idtechproducts.device.IDT_NEO2.device_setMerchantRecord (
            int index,
            boolean enabled,
            String merchantID,
            String merchantURL )
```

Set Merchant Record

Sets the burst mode for the device.

Parameters

index	Merchant Record Index. The valid value is 1–6.
enabled	1: The Merchant ID is valid, 0: The Merchant ID is not valid.
merchantID	The tag is 9F25.
merchantURL	The tag is 9F29.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.64 device_setNEOGen()

```
void com.idtechproducts.device.IDT_NEO2.device_setNEOGen (
            int gen )
```

Set the generation for the NEO 2 or 3 readers

Parameters

gen	The generation of the NEO 2/3 readers. 2: NEO 2, 3: NEO 3

13.2.2.65 device_setPackageDownloadDelay()

```
static boolean com.idtechproducts.device.IDT_NEO2.device_setPackageDownloadDelay (
            int delay )  [static]
```

Sets the package download delay for Bluetooth Tells the SDK what delay should be used during the package download for Bluetooth

Parameters

delay	time interval in ms from 20 to 50

Returns

true if success, false if failed

13.2.2.66 device_setPollMode()

```
int com.idtechproducts.device.IDT_NEO2.device_setPollMode (
            byte mode )
```

Set Poll Mode

Sets the poll mode for the device. Auto Poll keeps reader active, Poll On Demand only polls when requested by terminal

Parameters

mode	0 = Auto Poll, 1 = Poll On Demand

Returns

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

13.2.2.67 device_setRTCDateTime()

```
int com.idtechproducts.device.IDT_NEO2.device_setRTCDateTime (
            byte [] dateTime )
```

set RTC date and time of the device

Parameters

dateTime	<dateTime data>=""> is: 6 byte data for yyMMddHHmmss in hex. For example 0x171003102543 stands for 2017 Oct 3rd 10:25:43

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.68 device_setSource()

```
int com.idtechproducts.device.IDT_NEO2.device_setSource (
            byte [] data )
```

Set Source for RTC/LCD/Buzzer/LED

Set up the source for RTC/LCD/Buzzer/LED on the ViVOpay reader.

Parameters

data	the 2-byte source configuration for RTC/LCD/Buzzer/LED

Data Byte1 Definition:

Bit7 Bit6 | Bit5 Bit4 | Bit3 Bit2 | Bit1 Bit0

Reserved | RTC | LCD | Buzzer

Buzzer: 00 Don't use Buzzer 01 Use Buzzer from ViVOpay reader 10 Use Buzzer from external source 11 Use Buzzer from both reader and external source

LCD: 00 Don't use LCD 01 Use LCD from ViVOpay reader 10 Use LCD from external source 11 Use LCD from both reader and external source

RTC: 00 Don't use RTC 01 Use RTC from ViVOpay reader 10 Not allowed 11 Not allowed

Data Byte2 Definition:

Bit7 Bit6 | Bit5 Bit4 | Bit3 Bit2 | Bit1 Bit0

Reserved | Reserved | Power LED | Transaction LED

Transaction LED: 00 Don't use transaction LED 01 Use transaction LED from ViVOpay reader 10 Use transaction LED from external source 11 Use transaction LED from both reader and external source

Power LED: 00 Don't use power LED 01 Use power LED from ViVOpay reader 10 Use power LED from external source 11 Use power LED from both reader and external source

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.69 device_setSymmetric_RKI_URL()

```
void com.idtechproducts.device.IDT_NEO2.device_setSymmetric_RKI_URL (
            String srki_url )
```

Set symmetric RKI URL.

Parameters

srki_url	The URL for symmetric RKI

13.2.2.70 device_startRKI() [1/4]

```
int com.idtechproducts.device.IDT_NEO2.device_startRKI ( )
```

Start remote key injection.

Returns

success or error code.

See also

ErrorCode

13.2.2.71 device_startRKI() [2/4]

```
int com.idtechproducts.device.IDT_NEO2.device_startRKI (
            boolean isDemo )
```

Start remote key injection.

Parameters

isDemo	TRUE: for demo unit, FALSE: for production unit.

Returns

success or error code.

See also

ErrorCode

13.2.2.72 device_startRKI() [3/4]

```
int com.idtechproducts.device.IDT_NEO2.device_startRKI (
            String Key,
            boolean isDemo )
```

Start remote key injection with Selectable Key.

Parameters

Key	Selectable Key.
isDemo	TRUE: for demo unit, FALSE: for production unit.

Returns

success or error code.

See also

ErrorCode

13.2.2.73 device_startRKI() [4/4]

```
int com.idtechproducts.device.IDT_NEO2.device_startRKI (
            String Key )
```

Start remote key injection with Selectable Key.

Returns

success or error code.

See also

ErrorCode

13.2.2.74 device_startTransaction() [1/2]

```
int com.idtechproducts.device.IDT_NEO2.device_startTransaction (
            double amount,
            double amtOther,
            int type,
            final int timeout,
            byte [] tags )
```

Use Gen2 or Gen4 server.

Parameters

isGen2	TRUR: use Gen2 URL, FALSE: use Gen4 URL start Auto Config to search the profile.
strXMLFilename	Input the customized XML file as the templates to search the profile.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode stop Auto Config.

Returns

null. Start Device Transaction Request

Authorizes the MSR (or CTLS) or EMV transaction for an ICC card

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)
amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request. Passed as a string. Example, tag 9F02 with amount 0x000000000100 would be "9F0206000000000100" If tags 9F02 (amount),9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method. Note: To request tags to be included in default response, use tag DFEE1A, and specify tag list. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.75 device_startTransaction() [2/2]

```
int com.idtechproducts.device.IDT_NEO2.device_startTransaction (
            double amount,
            double amtOther,
            int type,
            final int timeout,
            byte [] tags,
            boolean isFastEMV )
```

Start Device Transaction Request

Authorizes the MSR (or CTLS) or EMV transaction for an ICC card

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)
amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request. Passed as a string. Example, tag 9F02 with amount 0x000000000100 would be "9F0206000000000100" If tags 9F02 (amount),9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method. Note: To request tags to be included in default response, use tag DFEE1A, and specify tag list. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37
isFastEMV	If TRUE, it will populate the IDTTransactionData.fastEMV with ASCII data similar to IDTech FastEMV KB output, after performing an auto-authenticate and auto-complete with ResultCode = Could Not Contact Host

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.76 device_updateFirmware() [1/2]

```
int com.idtechproducts.device.IDT_NEO2.device_updateFirmware (
            byte [] data,
            FIRMWARE_UPDATE_TYPES ft )
```

DEVICE INFO API Update the firmware of device.

Parameters

data	for all the block data of the firmware fm file.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.77 device_updateFirmware() [2/2]

```
int com.idtechproducts.device.IDT_NEO2.device_updateFirmware (
            String [] commands )
```

DEVICE INFO API Update the firmware of device.

Parameters

commands	for all the lines/commands of the firmware text file.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.78 emv_allowFallback()

```
static void com.idtechproducts.device.IDT_NEO2.emv_allowFallback (
            boolean allow )  [static]
```

Allow fallback for EMV transactions. Default is TRUE

Parameters

allow	TRUE = allow fallback, FALSE = don't allow fallback

13.2.2.79 emv_authenticateTransaction()

```
int com.idtechproducts.device.IDT_NEO2.emv_authenticateTransaction (
            byte [] tags )
```

Authenticate EMV Transaction Request

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

The tags will be returned in the callback routine.

Parameters

tags	TLV stream that can be used to update the following values: • 9F02: Amount • 9F03: Other amount • 9C: Transaction type • 5F57: Account type In addition tag DFEE1A can be sent to specify tag list to include in results. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.80 emv_callbackResponsePIN()

```
int com.idtechproducts.device.IDT_NEO2.emv_callbackResponsePIN (
            int mode,
            byte [] KSN,
            byte [] PIN )
```

Callback Response PIN Request

Provides PIN data to the kernel after a callback was received pinRequest delegate.

Parameters

mode	PIN Mode: • EMV_PIN_MODE_CANCEL = 0X00, • EMV_PIN_MODE_ONLINE_PIN_DUKPT = 0X01, • EMV_PIN_MODE_ONLINE_PIN_MKSK = 0X02, • EMV_PIN_MODE_OFFLINE_PIN = 0X03
KSN	Key Serial Number. If no pairing and PIN is plaintext, value is nil
PIN	PIN data, encrypted. If no pairing, PIN will be sent plaintext

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.81 emv_cancelTransaction()

```
int com.idtechproducts.device.IDT_NEO2.emv_cancelTransaction (
            ResDataStruct respData )
```

Cancel EMV Transaction

Cancels the currently executing EMV transaction.

Parameters

respData	error code will be stored in respData.statusCode

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.82 emv_completeTransaction()

```
int com.idtechproducts.device.IDT_NEO2.emv_completeTransaction (
            boolean commError,
            byte [] authCode,
            byte [] iad,
            byte [] tlvScripts,
            byte [] tags )
```

Complete EMV Transaction Request

Completes the EMV transaction for an ICC card when online authorization request is received from emv_↩ authenticateTransaction

The tags will be returned in the callback routine.

Parameters

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

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.83 emv_getAutoAuthenticateTransaction()

```
static boolean com.idtechproducts.device.IDT_NEO2.emv_getAutoAuthenticateTransaction ( )  [static]
```

Gets Auto Authentication for EMV Transactions Check the boolean value of Auto Authentication.

13.2.2.84 emv_getAutoCompleteTransaction()

```
static boolean com.idtechproducts.device.IDT_NEO2.emv_getAutoCompleteTransaction ( )  [static]
```

Gets Auto Completion for EMV Transactions Check the boolean value of Auto Completion.

13.2.2.85 emv_getEMVConfigurationCheckValue()

```
int com.idtechproducts.device.IDT_NEO2.emv_getEMVConfigurationCheckValue (
            ResDataStruct respData )
```

Get EMV Kernel configuration check value info

Parameters

respData	Response returned of Kernel configuration check value info

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.86 emv_getEMVKernelCheckValue()

```
int com.idtechproducts.device.IDT_NEO2.emv_getEMVKernelCheckValue (
            ResDataStruct respData )
```

Get EMV Kernel check value info

Parameters

respData	Response returned of Kernel check value info will be stored in respData.resData

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.87 emv_getEMVKernelVersion()

```
int com.idtechproducts.device.IDT_NEO2.emv_getEMVKernelVersion (
            StringBuilder version )
```

Polls device for EMV Kernel Version

Parameters

version	Response returned of Kernel Version

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.88 emv_lcdControlResponse()

```
void com.idtechproducts.device.IDT_NEO2.emv_lcdControlResponse (
            byte mode,
            byte selection )
```

Callback Response LCD Display

Provides menu selection responses to the kernel after a callback was received lcdDisplay delegate.

Parameters

mode	The choices are as follows • 0x00 Cancel • 0x01 Menu Display • 0x02 Normal Display get Function Key supply either 0x43 ('C') for Cancel, or 0x45 ('E') for Enter/accept • 0x08 Language Menu Display
selection	Line number in hex (0x01, 0x02), or 'C'/'E' of function key

Returns

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

13.2.2.89 emv_removeApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.emv_removeApplicationData (
            String aid,
            ResDataStruct respData )
```

Remove Application Data

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

Parameters

aid	Aid file to remove.
respData	Status Code in ResDataStruct.statusCode. If no application data exists, status code will be 0x60. Format error status code 0x05

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.90 emv_removeCAPK()

```
int com.idtechproducts.device.IDT_NEO2.emv_removeCAPK (
            byte [] capk,
            ResDataStruct respData )
```

Remove Certificate Authority Public Key

Removes the CAPK as specified by the RID/Index

Parameters

capk	6 byte CAPK = 5 bytes RID + 1 byte INDEX
respData	Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.91 emv_removeCRL()

```
int com.idtechproducts.device.IDT_NEO2.emv_removeCRL (
            byte [] crlList,
            ResDataStruct respData )
```

Remove Certificate Revocation List Entries

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

Parameters

crlList	containing the list of CRL to remove: [CRL1][CRL2]...[CRLn] where each [CRL] is 9 bytes: [5 bytes RID][1 byte CAPK Index][3 bytes serial number]
respData	Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.92 emv_removeTerminalData()

```
int com.idtechproducts.device.IDT_NEO2.emv_removeTerminalData (
            ResDataStruct respData )
```

Remove Terminal Data

Removes the Terminal Data.

Parameters

respData	Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.93 emv_retrieveAidList()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveAidList (
            ResDataStruct respData )
```

Retrieve Aid List

Returns all the AID names installed on the terminal.

Parameters

respData	Array of AID string names passed back in ResDataStruct.stringArray. Status Code in ResDataStruct.statusCode. If no AIDs exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.94 emv_retrieveApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveApplicationData (
            String aid,
            ResDataStruct respData )
```

Retrieve Application Data

Retrieves the TLV values of a provide AID.

Parameters

aid	Aid file to retrieve.
respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode. If no application data exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.95 emv_retrieveCAPK()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveCAPK (
            byte [] capk,
            ResDataStruct respData )
```

Retrieve Certificate Authority Public Key

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

Parameters

capk	6 bytes CAPK = 5 bytes RID + 1 byte Index
respData	Response returned in ResDataStruct.resData: [5 bytes RID][1 byte Index][1 byte Hash Algorithm][1 byte Encryption Algorithm][20 bytes HashValue][4 bytes Public Key Exponent][2 bytes Modulus Length][Variable bytes Modulus] Where: • Hash Algorithm: The only algorithm supported is SHA-1.The value is set to 0x01 • Encryption Algorithm: The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to 0x01. • HashValue: Which is calculated using SHA-1 over the following fields: RID & Index & Modulus & Exponent • Public Key Exponent: Actually, the real length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03), or 65537 (Format is 0x00 01 00 01) • Modulus Length: LenL LenH Indicated the length of the next field. • Modulus: This is the modulus field of the public key. Its length is specified in the field above.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.96 emv_retrieveCAPKList()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveCAPKList (
            ResDataStruct respData )
```

Retrieve the Certificate Authority Public Key list

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

Parameters

respData	ResDataStruct.resData = [key1][key2]...[keyn], each key 6 bytes where key = 5 bytes RID + 1 byte index

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.97 emv_retrieveCRL()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveCRL (
            ResDataStruct respData )
```

Retrieve the Certificate Revocation List

Returns the CRL entries on the terminal.

Parameters

respData	key Response returned in ResDataStruct.resData: list [CRL1][CRL2]...[CRLn], each CRL 9 bytes where CRL = 5 bytes RID + 1 byte index + 3 bytes serial number

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.98 emv_retrieveTerminalData()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveTerminalData (
            ResDataStruct respData )
```

Retrieve Terminal Data

Retrieves the TLV values of a the terminal.

Parameters

respData	Returns TLV in ResDataStruct.resData. Status Code in ResDataStruct.statusCode. If no terminal data exists, status code will be 0x60

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.99 emv_retrieveTransactionResult()

```
int com.idtechproducts.device.IDT_NEO2.emv_retrieveTransactionResult (
            byte [] tags,
            Map< String, Map< String, byte[]>> retrievedTags )
```

Retrieve Transaction Results

Retrieves specified EMV tags from the currently executing transaction.

Parameters

tags	Tags to be retrieved. Example 0x9F028A will retrieve tags 9F02 and 8A
retrievedTags	All requested tags returned as unencrypted, encrypted and masked tags. The tlv Map will contain a Map with key "tags" that has the unencrypted tag data, a Map with the key "masked" that has the masked tag data, and a Map with the key "encrypted" that has the encrypted tag data

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.100 emv_setApplicationData()

```
int com.idtechproducts.device.IDT_NEO2.emv_setApplicationData (
            String name,
            byte [] tlv,
            ResDataStruct respData )
```

Set Application Data

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

Parameters

name	Application name, 10-32 ASCII hex characters representing 5-16 bytes Example "a0000000031010"
tlv	Application data in TLV format.
respData	Status Code in ResDataStruct.statusCode. If AID list is full, status code will be 0x61. Format error status code 0x05

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.101 emv_setAutoAuthenticateTransaction()

```
static void com.idtechproducts.device.IDT_NEO2.emv_setAutoAuthenticateTransaction (
            boolean auto )  [static]
```

Sets Auto Authentication for EMV Transactions Tells the SDK to automatically execute Authenticate Transaction after StartEMV Transaction. TRUE by default

Parameters

auto	true = auto authentication on false = auto authentication off

13.2.2.102 emv_setAutoCompleteTransaction()

```
static void com.idtechproducts.device.IDT_NEO2.emv_setAutoCompleteTransaction (
            boolean auto )  [static]
```

Sets Auto Completion for EMV Transactions Tells the SDK to automatically execute Complete Transaction after EMV Authentication. FALSE by default

Parameters

auto	true = auto complete on false = auto complete off

13.2.2.103 emv_setCAPK()

```
int com.idtechproducts.device.IDT_NEO2.emv_setCAPK (
            byte [] key,
            ResDataStruct respData )
```

Set Certificate Authority Public Key

Sets the CAPK as specified by the CAKey structure

Parameters

key	CAKey format: [5 bytes RID][1 byte Index][1 byte Hash Algorithm][1 byte Encryption Algorithm][20 bytes HashValue][4 bytes Public Key Exponent][2 bytes Modulus Length][Variable bytes Modulus][2 bytes MAC Length][Variable bytes MAC Data] Where: • Hash Algorithm: The only algorithm supported is SHA-1.The value is set to 0x01 • Encryption Algorithm: The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to 0x01. • HashValue: Which is calculated using SHA-1 over the following fields: RID & Index & Modulus & Exponent • Public Key Exponent: Actually, the real length of the exponent is either one byte or 3 bytes. It can have two values: 3 (Format is 0x00 00 00 03), or 65537 (Format is 0x00 01 00 01) • Modulus Length: LenL LenH Indicated the length of the next field. • Modulus: This is the modulus field of the public key. Its length is specified in the field above. • MAC Length: LenL LenH Indicated the length of the next field. For Non-PCI device, it is 0x00 0x00. For PCI device, it is 0x1E 0x00. • MAC Data: For Non-PCI device, i does not exist. For PCI device, it is 30 bytes data consists of [2 bytes MAC Value Length][16 bytes MAC Value][2 bytes MAC Key KSN Length][10 bytes MAC Key KSN] Where: – MAC Value Length: 0x10 0x00 – MAC value: MAC-HOST. – MAC Key KSN Length: 0x0A 0x00 – MAC Key KSN: MAC DUKPT Key KSN
respData	Status Code in ResDataStruct.statusCode.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.104 emv_setCRL()

```
int com.idtechproducts.device.IDT_NEO2.emv_setCRL (
            byte [] crlList,
            ResDataStruct respData )
```

Set Certificate Revocation List

Sets the CRL

Parameters

crlList	Entries containing the RID, Index, and serial numbers to set [CRL1][CRL2]...[CRLn] where each [CRL] is 9 bytes: [5 bytes RID][1 byte CAPK Index][3 bytes serial number]
respData	error code will be stored in respData.statusCode

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.105 emv_setTerminalData()

```
int com.idtechproducts.device.IDT_NEO2.emv_setTerminalData (
            byte [] TLV,
            ResDataStruct respData )
```

Set Terminal Data

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

Parameters

TLV	TerminalData configuration file.
respData	Status Code in ResDataStruct.statusCode. If Flash error, status code will be 0x62. Format error status code 0x05

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.106 emv_setTerminalMajorConfiguration()

```
int com.idtechproducts.device.IDT_NEO2.emv_setTerminalMajorConfiguration (
            int configuration )
```

Sets the terminal major configuration in ICS .

Parameters

configuration	A configuration value, range 1-23 • 1 = 1C • 2 = 2C • 3 = 3C • 4 = 4C • 5 = 5C ... • 23 = 23C

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.107 emv_setTransactionParameters()

```
void com.idtechproducts.device.IDT_NEO2.emv_setTransactionParameters (
            double amount,
            double amtOther,
            int type,
            final int timeout,
            byte [] tags )
```

Set EMV Transaction Parameters

Set the parameters to be used on EMV transactions for an ICC card when Auto Poll is on and Burst Mode is off

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)
amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request. Passed as a string. Example, tag 9F02 with amount 0x000000000100 would be "9F0206000000000100" If tags 9F02 (amount),9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method. Note: To request tags to be included in default response, use tag DFEE1A, and specify tag list. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37

13.2.2.108 emv_startTransaction()

```
int com.idtechproducts.device.IDT_NEO2.emv_startTransaction (
            double amount,
            double amtOther,
            int type,
            final int timeout,
            byte [] tags,
            boolean forceOnline )
```

Start EMV Transaction Request

Authorizes the EMV transaction for an ICC card

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)

Parameters

amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request. Passed as a byte array. Example, tag 9F02 with amount 0x000000000100 would be "9F0206000000000100" If tags 9F02 (amount),9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method.
forceOnline	TRUE = do not allow offline approval, FALSE = allow ICC to approve offline if terminal capable Note: To request tags to be included in default response, use tag DFEE1A, and specify tag list. Example four tags 9F02, 9F36, 95, 9F37 to be included in response = DFEE1A079F029F369f9F37

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.109 felica_authentication()

```
int com.idtechproducts.device.IDT_NEO2.felica_authentication (
            byte [] key )
```

FeliCa Authentication Provides a key to be used in a follow up FeliCa Read with MAC (3 blocks max) or Write with MAC (1 block max). This command must be executed before each Read w/MAC or Write w/MAC command

Parameters

key	16 byte key used for MAC generation of Read or Write with MAC

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.110 felica_poll()

```
int com.idtechproducts.device.IDT_NEO2.felica_poll (
            byte [] systemCode,
            ResDataStruct respData )
```

FeliCa Poll for Card

Polls for a Felica Card

Parameters

systemCode	System Code. Must be 2 bytes
respData	response data will be stored in respData.resData. Poll response as explained in FeliCA Lite-S User's Manual

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.111 felica_read()

```
int com.idtechproducts.device.IDT_NEO2.felica_read (
            byte [] serviceCodeList,
            int blockCnt,
            byte [] blockList,
            ResDataStruct respData )
```

FeliCa Read

Reads up to 4 blocks.

Parameters

serviceCodeList	Service Code List. Each service code in Service Code List = 2 bytes of data
blockCnt	Number of blocks in blockList. Maximum 4 block requests
blockList	Block to read. Each block in blockList = 2 or 3 bytes of data.
respData	Blocks read will be stored in respData.resData. Each block 16 bytes.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.112 felica_readWithMac()

```
int com.idtechproducts.device.IDT_NEO2.felica_readWithMac (
            int blockCnt,
            byte [] blockList,
            ResDataStruct respData )
```

FeliCa Read with MAC Generation

```
Reads up to 3 blocks with MAC Generation.  FeliCa Authentication must be performed first
```

Parameters

blockCnt	Number of blocks in blockList. Maximum 3 block requests
blockList	Block to read. Each block in blockList = 2 or 3 bytes of data.
respData	Blocks read will be stored in respData.resData. Each block 16 bytes.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.113 felica_requestService()

```
int com.idtechproducts.device.IDT_NEO2.felica_requestService (
            byte [] nodeCode,
            ResDataStruct respData )
```

FeliCa Request Service

Request Service for a Felica Card

Parameters

nodeCode	Node Code List. Each node 2 bytes
respData	response data will be stored in respData.resData. Response as explained in FeliCA Lite-S User's Manual

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.114 felica_SendCommand()

```
int com.idtechproducts.device.IDT_NEO2.felica_SendCommand (
            byte command[],
            ResDataStruct respData )
```

FeliCa Send Command

Send a Felica Command

Parameters

command	Command data from settlement center to be sent to felica card
respData	response data will be stored in respData.resData. Response as explained in FeliCA Lite-S User's Manual

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.115 felica_write()

```
int com.idtechproducts.device.IDT_NEO2.felica_write (
            byte [] serviceCodeList,
            int blockCnt,
            byte [] blockList,
            byte [] blockData,
            ResDataStruct respData )
```

FeliCa Write

Writes a block

Parameters

serviceCodeList	Service Code List. Each service code in Service Code List = 2 bytes of data
blockCnt	Number of blocks in blockList. Currently only support 1 block.
blockList	Block list. Each block in blockList = 2 or 3 bytes of data.
blockData	Block to write. Must be 16 bytes.
respData	If successful, the Status Flag (2 bytes) is stored in respData.resData. Status flag response as explained in FeliCA Lite-S User's Manual, Section 4.5

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.116 felica_writeWithMac()

```
int com.idtechproducts.device.IDT_NEO2.felica_writeWithMac (
            byte blockNum,
            byte [] blockData )
```

FeliCa Write with MAC Generation

Writes a block with MAC Generation. FeliCa Authentication must be performed first

Parameters

blockNum	Number of block
blockData	Block to write. Must be 16 bytes.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.117 forwardTransaction()

```
int com.idtechproducts.device.IDT_NEO2.forwardTransaction (
            WorldpayListener callback,
            String forwardID,
            String password,
            boolean bypassProcessing )
```

forwardTransaction Send the saved data to WorldPay and complete the transaction.

Parameters

callback	callback is the callback to send the results. Should the the same as executeTransaction callback.
forwardID	= ID, which could be either unique ID or Memo.
password	= password. If null/blank, no password.
bypassProcessing	= true means read the file from disk, but don't send to WorldPay, then delete the transaction as if you did send to WorldPay. The purpose of this is to allow them to delete transactions from the storage without sending to WorldPay.

Returns

RETURN_CODE: Values can be parsed with device_getIDGStatusCodeString()

13.2.2.118 getSDKInstance()

```
static IDT_Device com.idtechproducts.device.IDT_NEO2.getSDKInstance ( )  [static]
```

Returns an instance of the currently initialized IDT_Device class.

Returns

IDT_Device instance

13.2.2.119 icc_exchangeAPDU()

```
int com.idtechproducts.device.IDT_NEO2.icc_exchangeAPDU (
            byte [] dataAPDU,
            ResDataStruct response )
```

Exchange APDU

Sends an APDU packet to the ICC. If successful, response is returned in APDUResult class instance in response parameter.

Parameters

dataAPDU	APDU data packet
response	Unencrypted/encrypted parsed APDU response will be stored in response.resData

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.120 icc_getICCReaderStatus()

```
int com.idtechproducts.device.IDT_NEO2.icc_getICCReaderStatus (
            ICCReaderStatusStruct ICCStatus )
```

Get Reader Status

Returns the reader status

Parameters

ICCStatus	Pointer that will return with the ICCReaderStatus results. bit 0: 0 = ICC Power Not Ready, 1 = ICC Powered bit 1: 0 = Card not seated, 1 = card seated

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.121 icc_passthroughOffICC()

```
int com.idtechproducts.device.IDT_NEO2.icc_passthroughOffICC ( )
```

Disables pass through mode for ICC. Required when executing transactions (start EMV, start MSR, authenticate transaction)

Returns

success or error code.

See also

ErrorCode

13.2.2.122 icc_passthroughOnICC()

```
int com.idtechproducts.device.IDT_NEO2.icc_passthroughOnICC ( )
```

Enables pass through mode for ICC. Required when direct ICC commands are required (power on/off ICC, exchange APDU)

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.123 icc_powerOffICC()

```
int com.idtechproducts.device.IDT_NEO2.icc_powerOffICC (
            ResDataStruct respData )
```

Power Off ICC

Powers down the ICC

Parameters

respData	the response code will be stored in respData.statusCode

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

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

13.2.2.124 icc_powerOnICC()

```
int com.idtechproducts.device.IDT_NEO2.icc_powerOnICC (
            ResDataStruct atrPPS )
```

Power up the currently selected microprocessor card in the ICC reader. It follows the ISO7816-3 power up sequence and returns the ATR as its response.

Parameters

Parameters

atrPPS	the class for ATR string. the ATR string is following: 1. 2D01: Card Not Supported; 2. 2D03: Card Not Supported, wants CRC; 3. 690D: Command not supported on reader without ICC support; 4. 8100: ICC error time out on power-up; 5. 8200: invalid TS character received; 6. 8500: PPS confirmation error; 7. 8600: Unsupported F, D, or combination of F and D; 8. 8700: protocol not supported EMV TD1 out of range; 9. 8800: power not at proper level; 10. 8900: ATR length too long; 11. 8B01: EMV invalid TA1 byte value∗; 12. 8B02: EMV TB1 required∗; 13. 8B03: EMV Unsupported TB1 only 00 allowed∗; 14. 8B04: EMV Card Error, invalid BWI or CWI∗; 15. 8B06: EMV TB2 not allowed in ATR∗; 16. 8B07: EMV TC2 out of range∗; 17. 8B08: EMV TC2 out of range∗; 18. 8B09: per EMV96 TA3 must be $>$ 0xF∗; 19. 8B10: ICC error on power-up; 20. 8B11: EMV T=1 then TB3 required∗;
	21. 8B12: Card Error, invalid BWI or CWI; 22. 8B13: Card Error, invalid BWI or CWI;

Parameters

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.125 lcd_addButton()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addButton (
            String screenName,
            String buttonName,
            byte type,
            byte alignment,
            int xCord,
            int yCord,
            String label,
            IDTLCDData returnItem )
```

Add Button

Adds a button to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add button
buttonName	Button name that will be the target of add button
type	Button Type • Large = 0x01 • Medium = 0x02 • Invisible = 0x03 (70px by 60 px)
alignment	Position for Button • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x andy • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for Button, range 0-271
yCord	y-coordinate for Button, range 0-479
label	Label to show on the button. Required for Large/Medium buttons. Not used for Invisible buttons.
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created button

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.126 lcd_addEthernet()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addEthernet (
            String screenName,
            String objectName,
            byte alignment,
            int xCord,
            int yCord,
            IDTLCDData returnItem )
```

Add Ethernet Settings

Adds an Ethernet settings to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add widget
objectName	Object name that will be the target of add widget
alignment	Position for widget • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x and y • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for widget, range 0-271
yCord	y-coordinate for widget, range 0-479
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created widget

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

Item	Maximum can be created for each screen
Text Area	20
Large Button	8
Medium Button	16
Invisible Button	16
Numeric Entry	1
Ethernet Setting	1
Led widget	1

Item	Maximum can be created for each screen
image	20

13.2.2.127 lcd_addExtVideo()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addExtVideo (
            String screenName,
            String objectName,
            byte alignment,
            int xCord,
            int yCord,
            byte loop,
            byte numVideos,
            String filenames,
            IDTLCDData returnItem )
```

Add Extended Video

Adds a extended video to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add video
objectName	Object name that will be the target of add video
alignment	Position for Video • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x and y • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for Video, range 0-271
yCord	y-coordinate for Video, range 0-479
loop	loop the videos when it's done. 0: Do not Loop 1: Loop
numVideos	number of video files, range 1-4
filenames	Filenames of the videos separated by '\0'. Must be available in the sd card.
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created video

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

Item	Maximum can be created for each screen
Text Area	20
Large Button	8
Medium Button	16
Invisible Button	16
Numeric Entry	1
Ethernet Setting	1

Item	Maximum can be created for each screen
Led widget	1
image	20
video	1

13.2.2.128 lcd_addImage()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addImage (
            String screenName,
            String objectName,
            byte alignment,
            int xCord,
            int yCord,
            String filename,
            IDTLCDData returnItem )
```

Add Image

Adds a image to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add image
objectName	Object name that will be the target of add image
alignment	Position for Image • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x and y • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for Image, range 0-271
yCord	y-coordinate for Image, range 0-479
filename	Filename of the image. Must be available in device memory.
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created image

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

Item	Maximum can be created for each screen
Text Area	20
Large Button	8
Medium Button	16
Invisible Button	16
Numeric Entry	1
Ethernet Setting	1
Led widget	1
image	20

13.2.2.129 lcd_addLED()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addLED (
            String screenName,
            String objectName,
            byte alignment,
            int xCord,
            int yCord,
            IDTLCDData returnItem,
            byte [] LED )
```

Add LED

Adds a LED widget to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add LED
objectName	Object name that will be the target of add LED
alignment	Position for LED • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x andy • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for LED, range 0-271
yCord	y-coordinate for LED, range 0-479
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created widget
LED	Must be 4 bytes, LED 0 = byte 0, LED 1 = byte 1, LED 2 = byte 2, LED 3 = byte 3 • Value 0 = LED OFF • Value 1 = LED Green • Value 2 = LED Yellow • Value 3 = LED Red

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

Item	Maximum can be created for each screen
Text Area	20
Large Button	8
Medium Button	16
Invisible Button	16
Numeric Entry	1
Ethernet Setting	1
Led widget	1

Item	Maximum can be created for each screen
image	20

13.2.2.130 lcd_addText()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addText (
            String screenName,
            String objectName,
            byte alignment,
            int xCord,
            int yCord,
            int width,
            int height,
            byte fontID,
            byte [] color,
            String label,
            IDTLCDData returnItem )
```

Add text

Adds a text component to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add text
objectName	Object name that will be the target of add text
alignment	Position for Text • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x and y • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for Text, range 0-271
yCord	y-coordinate for Text, range 0-479
width	Width of text area
height	Height of text area
fontID	Font ID
color	Four bytes for color, example, Blue = 0xFF000000, Black = 0x00000000 • Byte 0 = B • Byte 1 = G • Byte 2 = R • Byte 3 = Reserved
label	Label to show on the text
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created text area

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

Font ID	Typography Name	Font	Size
0	RoundBold_12	RoundBold.ttf	12
1	RoundBold_18	RoundBold.ttf	18
2	RoundBold_24	RoundBold.ttf	24
3	RoundBold_36	RoundBold.ttf	36
4	RoundBold_48	RoundBold.ttf	48
5	RoundBold_60	RoundBold.ttf	60
6	RoundBold_72	RoundBold.ttf	72
7	RoundCondensedBold_12	RoundCondensedBold.ttf	12
8	RoundCondensedBold_18	RoundCondensedBold.ttf	18
9	RoundCondensedBold_24	RoundCondensedBold.ttf	24
10	RoundCondensedBold_36	RoundCondensedBold.ttf	36
11	RoundCondensedBold_48	RoundCondensedBold.ttf	48
12	RoundCondensedBold_60	RoundCondensedBold.ttf	60
13	RoundCondensedBold_72	RoundCondensedBold.ttf	72
14	RoundCondensedMedium_12	RoundCondensedMedium_↩ 0.ttf	12
15	RoundCondensedMedium_18	RoundCondensedMedium_↩ 0.ttf	18
16	RoundCondensedMedium_24	RoundCondensedMedium_↩ 0.ttf	24
17	RoundCondensedMedium_36	RoundCondensedMedium_↩ 0.ttf	36
18	RoundCondensedMedium_48	RoundCondensedMedium_↩ 0.ttf	48
19	RoundCondensedMedium_60	RoundCondensedMedium_↩ 0.ttf	60
20	RoundCondensedMedium_72	RoundCondensedMedium_↩ 0.ttf	72
21	RoundCondensedSemibold_12	RoundCondensedSemibold.ttf	12
22	RoundCondensedSemibold_18	RoundCondensedSemibold.ttf	18
23	RoundCondensedSemibold_24	RoundCondensedSemibold.ttf	24
24	RoundCondensedSemibold_36	RoundCondensedSemibold.ttf	36
25	RoundCondensedSemibold_48	RoundCondensedSemibold.ttf	48
26	RoundCondensedSemibold_60	RoundCondensedSemibold.ttf	60
27	RoundCondensedSemibold_72	RoundCondensedSemibold.ttf	72
28	RoundMedium_12	RoundMedium.ttf	12
29	RoundMedium_18	RoundMedium.ttf	18
30	RoundMedium_24	RoundMedium.ttf	24
31	RoundMedium_36	RoundMedium.ttf	36
32	RoundMedium_48	RoundMedium.ttf	48
33	RoundMedium_60	RoundMedium.ttf	60
34	RoundMedium_72	RoundMedium.ttf	72
35	RoundSemibold_12	RoundSemibold.ttf	12
36	RoundSemibold_18	RoundSemibold.ttf	18
37	RoundSemibold_24	RoundSemibold.ttf	24
38	RoundSemibold_36	RoundSemibold.ttf	36
39	RoundSemibold_48	RoundSemibold.ttf	48
40	RoundSemibold_60	RoundSemibold.ttf	60
41	RoundSemibold_72	RoundSemibold.ttf	72

Item	Maximum can be created for each screen
Text Area	20
Large Button	8
Medium Button	16
Invisible Button	16
Numeric Entry	1
Ethernet Setting	1
Led widget	1
image	20

13.2.2.131 lcd_addVideo()

```
int com.idtechproducts.device.IDT_NEO2.lcd_addVideo (
            String screenName,
            String objectName,
            byte alignment,
            int xCord,
            int yCord,
            String filename,
            IDTLCDData returnItem )
```

Add Video

Adds a video to a selected screen. Must execute lcd_createScreen first to establish a screen to draw on.

Parameters

screenName	Screen name that will be the target of add video
objectName	Object name that will be the target of add video
alignment	Position for Video • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x and y • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
xCord	x-coordinate for Video, range 0-271
yCord	y-coordinate for Video, range 0-479
filename	Filename of the video. Must be available in the sd card.
returnItem	The item includes screen ID, object ID, top-left x-coordinate, top-left y-coordinate, bottom-right x-coordinate, bottom-left y-coordinate, which are all assigned to the created video

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

Item	Maximum can be created for each screen
Text Area	20
Large Button	8
Medium Button	16

Item	Maximum can be created for each screen
Invisible Button	16
Numeric Entry	1
Ethernet Setting	1
Led widget	1
image	20
video	1

13.2.2.132 lcd_clearDisplay()

```
int com.idtechproducts.device.IDT_NEO2.lcd_clearDisplay ( )
```

Clear Display

Command to clear the display screen on the reader.It returns the display to the currently defined background color and terminates all events

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.133 lcd_clearScreenInfo()

```
int com.idtechproducts.device.IDT_NEO2.lcd_clearScreenInfo ( )
```

Clear Screen Info

Clear all current screen information in RAM and flash. And then show'power-on screen'

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.134 lcd_cloneScreen()

```
int com.idtechproducts.device.IDT_NEO2.lcd_cloneScreen (
            String screenName,
            String cloneName,
            IDTLCDData returnItem )
```

Clone Screen

Clones an existing screen.

Parameters

screenName	Screen name to clone.
cloneName	Name of the cloned screen.
returnItem	Screen ID of the cloned screen will be returned in returnItem

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.135 lcd_createScreen()

```
int com.idtechproducts.device.IDT_NEO2.lcd_createScreen (
            String screenName,
            IDTLCDData returnItem )
```

Create Screen

Creates a new screen.

Parameters

screenName	Screen name to use.
returnItem	includes the Screen ID that was created.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.136 lcd_destroyScreen()

```
int com.idtechproducts.device.IDT_NEO2.lcd_destroyScreen (
            String screenName )
```

Destroy Screen

Destroys a previously created inactive screen. The screen cannot be active

Parameters

screenName	Screen name to destroy. The screen number is assigned with lcd_createScreen.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.137 lcd_getActiveScreen()

```
int com.idtechproducts.device.IDT_NEO2.lcd_getActiveScreen (
            IDTLCDData returnItem )
```

Get Active Screen

Returns the active screen ID.

Parameters

returnItem	includes the active Screen name.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.138 lcd_getAllObjects()

```
int com.idtechproducts.device.IDT_NEO2.lcd_getAllObjects (
            String screenName,
            List< IDTLCDData > IDTScreenInfos )
```

Get All Objects on Screen

Get all created objects' name on certain screen

Parameters

screenName	Screen name to get all objects
IDTScreenInfos	The list of IDTLCDData that contains all created objects each element includes -objectID of a created object -objectName of a created object

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.139 lcd_getAllScreens()

```
int com.idtechproducts.device.IDT_NEO2.lcd_getAllScreens (
            List< IDTLCDData > IDTScreenInfos )
```

Get All Screens

Get all created screens' name

Parameters

IDTScreenInfos	Array of all created screens each element includes -screenID of a created screen -screenName of a created screen

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.140 lcd_getButtonEvent()

```
int com.idtechproducts.device.IDT_NEO2.lcd_getButtonEvent (
            IDTLCDData returnItem )
```

Get Button Event

Reports back the ID of the button that encountered a click event after the last Get Button Event.

Parameters

returnItem	conbtains screenID, objectID, screenName, objectName, and longPressed of the button event screenID Screen ID of the last clicked button objectID Button ID of the last clicked button screenName Screen Name of the last clicked button screenNameLen Length of screenName objectName Button Name of the last clicked button objectNameLen Length of objectName longPress 1 = Long Press, 0 = Short Press

Returns

RETURN_CODE: Values can be parsed with device_getIDGStatusCodeString()

13.2.2.141 lcd_linkUIWithTransactionMessageId()

```
int com.idtechproducts.device.IDT_NEO2.lcd_linkUIWithTransactionMessageId (
            byte MessageId,
            String screenName )
```

Link UI with Transaction Message ID

Link an existing Customer UI ID with a specified transaction message ID. During transaction, the linked System UI will be replaced by the linked Customer UI

Parameters

MessageId	Transaction Message ID: Refer to Doc “EMV Display Message ID Assignment” Selection by Customer
screenName	Name of the screen (No longer than 32 bytes including the NULL character)

Returns

RETURN_CODE: Values can be parsed with device_getIDGStatusCodeString()

13.2.2.142 lcd_loadScreenInfo()

```
int com.idtechproducts.device.IDT_NEO2.lcd_loadScreenInfo ( )
```

Load Screen Info

Load all current screen information from RAM to flash

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.143 lcd_queryObjectbyID()

```
int com.idtechproducts.device.IDT_NEO2.lcd_queryObjectbyID (
            int objectID,
            List< IDTLCDData > IDTScreenInfos )
```

Queery Object by ID

Check if the given object exists or not. If exists, return all screen names which contains the object of the given object ID

Parameters

objectID	ID of the checked object
IDTScreenInfos	a list of IDTLCDData which contains screen names containing the item

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.144 lcd_queryObjectbyName()

```
int com.idtechproducts.device.IDT_NEO2.lcd_queryObjectbyName (
            String objectName,
            List< IDTLCDData > IDTScreenInfos )
```

Queery Object by Name

Check if the given object exists or not. If exists, return all screen names which contains the object of the given object name

Parameters

objectName	Name of the checked object
IDTScreenInfos	Array of all the screen names that contain the object

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.145 lcd_queryScreenbyID()

```
int com.idtechproducts.device.IDT_NEO2.lcd_queryScreenbyID (
            int screenID,
            IDTLCDData returnItem )
```

Query Screen by ID

Check if the given screen exists or not

Parameters

screenID	ID of the checked screen
returnItem	contains the screenName of the checked screen

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.146 lcd_queryScreenbyName()

```
int com.idtechproducts.device.IDT_NEO2.lcd_queryScreenbyName (
            String screenName,
            IDTLCDData returnItem )
```

Queery Screen by Name

Check if the given screen exists or not

Parameters

screenName	Name of the checked screen
returnItem	return the checking result in returnItem

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.147 lcd_removeItem()

```
int com.idtechproducts.device.IDT_NEO2.lcd_removeItem (
            String screenName,
            String objectName )
```

Removed Item

Removes a component.

Parameters

screenName	Screen name where to remove the target from.
objectName	Identifier of the component

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.148 lcd_setBacklight()

```
int com.idtechproducts.device.IDT_NEO2.lcd_setBacklight (
            boolean isBacklightOn,
            byte backlightVal )
```

Set Backlight

Set backlight percentage. If the percent $>$ 100, it will be rejected. If 0 $<$ percent $<$ 10, backlight percent will be set to 10. If percent == 0, backlight will be turned off

Parameters

backlightVal	Backlight percent value to be sat

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.149 lcd_showScreen()

```
int com.idtechproducts.device.IDT_NEO2.lcd_showScreen (
            String screenName )
```

Show Screen

Displays and makes active a previously created screen.

Parameters

screenName	Screen to display. The screen name is defined with lcd_createScreen.

Returns

RETURN_CODE: Values can be parsed with device_getIDGStatusCodeString()

13.2.2.150 lcd_storeScreenInfo()

```
int com.idtechproducts.device.IDT_NEO2.lcd_storeScreenInfo ( )
```

Store Screen Info

Store all current screen information from RAM to flash

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.151 lcd_updateColor()

```
int com.idtechproducts.device.IDT_NEO2.lcd_updateColor (
            String screenName,
```

```
            String objectName,
            byte [] color )
```

Update Color

Updates the component color, or updates the LED colors if specifying LCD component

Parameters

screenName	Screen name that will be the target of update color
objectName	Identifier of the component
color	Non LCD Widget: Four bytes for color, example, Blue = 0xFF000000, Black = 0x00000000 • Byte 0 = B • Byte 1 = G • Byte 2 = R • Byte 3 = Reserved LCD Widget: Four bytes for LED color, byte 0 = LED 0, byte 1 = LED 1, byte 2 = LED2, byte 3 = LED 3 • Value 0 = LED OFF • Value 1 = LED Green • Value 2 = LED Yellow • Value 3 = LED Red

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.152 lcd_updateLabel()

```
int com.idtechproducts.device.IDT_NEO2.lcd_updateLabel (
            String screenName,
            String objectName,
            String label )
```

Update Label

Updates the component label.

Parameters

screenName	Screen name that will be the target of update label
objectName	Identifier of the component
label	Label to show on the component

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.153 lcd_updatePosition()

```
int com.idtechproducts.device.IDT_NEO2.lcd_updatePosition (
            String screenName,
            String objectName,
            byte alignment,
            int new_xCord,
            int new_yCord )
```

Update Position

Updates the component position.

Parameters

screenName	Screen Name that will be the target of update position
objectName	Identifier of the component
alignment	Alignment for the target • 0 = Display object at the horizon center of specified y, while x ignored • 1 = Display object at specified x andy • 2 = Display object at center of screen, x, y are both ignored • 3 = Display object at left of the screen of specified y, while x ignored • 4 = Display object at right of the screen of specified y, while x ignored
new_xCord	x-coordinate for Text, range 0-271
new_yCord	y-coordinate for Text, range 0-479

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString

13.2.2.154 log_deleteLogs()

```
int com.idtechproducts.device.IDT_NEO2.log_deleteLogs ( )
```

delete the log in the root path of SD card.

Returns

number of log files deleted

See also

log_setSaveLogEnable

13.2.2.155 log_setSaveLogEnable()

```
void com.idtechproducts.device.IDT_NEO2.log_setSaveLogEnable (
            boolean enable )
```

Enable/Disable save the log into the root path of SD card.

Parameters

enable	true: enable save the log, the log includes the .txt text log and .wav signals file. false: disable save the log.

Returns

none

See also

deleteLogs

13.2.2.156 log_setVerboseLoggingEnable()

```
void com.idtechproducts.device.IDT_NEO2.log_setVerboseLoggingEnable (
            boolean enable )
```

Enable/Disable Verbose Logging show in the logcat view window.

Parameters

enable	true: enable to show the log in the logcat view window. false: disable to show the log in the logcat view window.

Returns

none

13.2.2.157 msr_cancelMSRSwipe()

```
int com.idtechproducts.device.IDT_NEO2.msr_cancelMSRSwipe ( )
```

Disable MSR swipe card.

Cancels MSR swipe request.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.158 msr_defaultAllSetting()

```
int com.idtechproducts.device.IDT_NEO2.msr_defaultAllSetting ( )
```

Default all setting of Mask and Encryption.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.159 msr_startMSRSwipe() [1/2]

```
int com.idtechproducts.device.IDT_NEO2.msr_startMSRSwipe ( )
```

Enable MSR swipe card. Returns encrypted MSR data or function key value by call back function. The function swipeMSRData in interface OnReceiverListener will be called if swiping card data received.

See also

OnReceiverListener

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.160 msr_startMSRSwipe() [2/2]

```
int com.idtechproducts.device.IDT_NEO2.msr_startMSRSwipe (
            int timeout )
```

Enable MSR swipe card. Returns encrypted MSR data or function key value by call back function.
The function swipeMSRData in interface OnReceiverListener will be called if swiping card data received.

See also

OnReceiverListener

Parameters

timeout	Swipe Timeout Value timeout value in seconds; maximum value is 255 seconds. If it is 0, it will be set to 5 seconds.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.161 phone_getInfoManufacture()

```
String com.idtechproducts.device.IDT_NEO2.phone_getInfoManufacture ( )
```

Get manufacture version.

Returns

the manufacture info

13.2.2.162 phone_getInfoModel()

```
String com.idtechproducts.device.IDT_NEO2.phone_getInfoModel ( )
```

Get phones's model number information.

Returns

the model number information.

13.2.2.163 pin_cancelPINEntry()

```
int com.idtechproducts.device.IDT_NEO2.pin_cancelPINEntry ( )
```

Cancel PIN Entry

Cancels PIN entry request

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.164 pin_displayMessageGetAmount()

```
int com.idtechproducts.device.IDT_NEO2.pin_displayMessageGetAmount (
            byte DisplayFlag,
            byte KeyMaxLen,
            byte KeyMinLen,
            byte [] TextDisplayMsg,
            byte [] DisplayMsgSig,
            ResDataStruct respData )
```

Display Message and Get Numeric Key

make device display requested message and then get Numeric Key sequence from device.

Parameters

DisplayFlag	Display Flag reserved

Parameters

KeyMaxLen	The max length for numeric key. Value is 1-15
KeyMinLen	The min length for numeric key. Value is 1-15
TextDisplayMsg	Plaintext Display Message. ASCII code.
DisplayMsgSig	256 bytes Display message signed by Numeric Private Key using RSAPSS algorithm. Algorithm: 1. Calculate 32 bytes Hash for "<Display Flag><Key Max Length>< Key Min Length><Plaintext Display Message>" 2. Using RSAPSS algorithm calculate the Hash to be 256 bytes Raw Data 3. Using Numeric Private Key to sign the Raw Data to be 256 bytes signature
respData	Status Code in ResDataStruct.statusCode.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString()

13.2.2.165 pin_displayMessageGetEncryptedPIN()

```
int com.idtechproducts.device.IDT_NEO2.pin_displayMessageGetEncryptedPIN (
            byte PANKeyType,
            byte [] PAN,
            byte PINMaxLen,
            byte PINMinLen,
            byte [] LCDMsg,
            ResDataStruct respData )
```

Display Message and Get Encrypted PIN

make device display requested message and then get encrypted PIN from device.

Parameters

PANKeyType	PAN and Key Type. 00h – MKSK to encrypt PIN, Internal PAN (from MSR) 01h – DUKPT to encrypt PIN, Internal PAN (from MSR) 10h –MKSK to encrypt PIN, External Plaintext PAN 11h – DUKPT to encrypt PIN, External Plaintext PAN 20h –MKSK to encrypt PIN, External Ciphertext PAN 21h – DUKPT to encrypt PIN, External Ciphertext PAN
PAN	If PAN is 00h, this field is not present. ASCII code for digital (30h-39h) If TDES PIN DUKPT Key, length is 16 bytes digital If AES PIN DUKPT Key, length is 12∼19 bytes digital Device will check PAN length, if length is error, device will response fail state code
PINMaxLen	Max Len of PIN, Value is 4-12
PINMinLen	Min Len of PIN, Value is 4-12
LCDMsg	If LCD Length is 00h, this field is not present ASCII code Display Message
respData	Status Code in ResDataStruct.statusCode.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString()

13.2.2.166 pin_displayMessageGetNumericKey()

```
int com.idtechproducts.device.IDT_NEO2.pin_displayMessageGetNumericKey (
            byte DisplayFlag,
            byte KeyMaxLen,
            byte KeyMinLen,
            byte [] TextDisplayMsg,
            byte [] DisplayMsgSig,
            ResDataStruct respData )
```

Display Message and Get Numeric Key

make device display requested message and then get Numeric Key sequence from device.

Parameters

DisplayFlag	Display Flag 0 - Display numeric for numeric key on LCD 1 - Display "∗" for numeric key on LCD
KeyMaxLen	The max length for numeric key. Value is 1-16
KeyMinLen	The min length for numeric key. Value is 1-16
TextDisplayMsg	Plaintext Display Message. ASCII code.
DisplayMsgSig	256 bytes Display message signed by Numeric Private Key using RSAPSS algorithm. Algorithm: 1. Calculate 32 bytes Hash for "<Display Flag><Key Max Length>< Key Min Length><Plaintext Display Message>" 2. Using RSAPSS algorithm calculate the Hash to be 256 bytes Raw Data 3. Using Numeric Private Key to sign the Raw Data to be 256 bytes signature
respData	Status Code in ResDataStruct.statusCode.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString()

13.2.2.167 pin_getFunctionKey()

```
int com.idtechproducts.device.IDT_NEO2.pin_getFunctionKey (
            ResDataStruct respData )
```

Get Function Key

get a single Keypress value from device.

Parameters

respData	the response data will be stored in respData.resData if available

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

13.2.2.168 registerListen()

```
void com.idtechproducts.device.IDT_NEO2.registerListen ( )
```

General API:registerListen.

registerListen to enable SDK detect the phone jack plug in/off notification

13.2.2.169 release()

```
void com.idtechproducts.device.IDT_NEO2.release ( )
```

release, make the SDK in the idle status.

13.2.2.170 setIDT_Device()

```
void com.idtechproducts.device.IDT_NEO2.setIDT_Device (
            FirmwareUpdateTool fwTool )
```

Sets the Device Type for NEO2 USB devices with the generation number (2 for NEO 2 and 3 for NEO 3) Tells the SDK what device type and PID to configure for For System Use Only

Parameters

fwTool	Parameter for firmware update

13.2.2.171 unregisterListen()

```
void com.idtechproducts.device.IDT_NEO2.unregisterListen ( )
```

unregisterListen to disable the detect

13.2.2.172 useUSBIntentFilter()

```
static void com.idtechproducts.device.IDT_NEO2.useUSBIntentFilter ( )  [static]
```

Use USB Intent Filter For USB Devices, you may opt to incorporate an Intent Filter that will automatically start your application when a specific USB device is attached. The SDK must be informed to bypass it's normal Enumeration of USB Devices when an Intent Filter is being use. This function MUST be called BEFORE device_setDeviceType() is executed if a USB Intent Filter is being utilized. https://developer.android.com/guide/topics/connectivity/usb/host.html

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

- Source_Android/IDT_NEO2.java

13.3 com.idtechproducts.device.IDTEMVData Class Reference

Public Attributes

- int result

  Result Code =.
- int encryptionMode

  0 = TDES, 1 = AES
- int cardType

  0 = Contact, 1 = Contactless
- boolean **emv_hasReversal** = false
- boolean **emv_hasAdvise** = false
- Map< String, byte[]> unencryptedTags

  Unencrypted EMV Tags. Key = tag name (String), Object = tag value (byte[])
- Map< String, byte[]> encryptedTags

  Encrypted EMV Tags. Key = tag name (String), Object = tag value (byte[])
- Map< String, byte[]> maskedTags

  Masked EMV Tags. Key = tag name (String), Object = tag value (byte[])
- IDTMSRData msr_cardData

  Tag DFEE23 parsed into a msr_cardData object.

Static Public Attributes

- static final int **APPROVED_OFFLINE** = 0x0000
- static final int **DECLINED_OFFLINE** = 0x0001
- static final int **APPROVED** = 0x0002
- static final int **DECLINED** = 0x0003
- static final int **GO_ONLINE** = 0x0004
- static final int **CALL_YOUR_BANK** = 0x0005
- static final int **NOT_ACCEPTED** = 0x0006
- static final int **USE_MAGSTRIPE** = 0x0007
- static final int **TIME_OUT** = 0x0008
- static final int **GO_ONLINE_CTLS** = 0x0009
- static final int **START_TRANS_SUCCESS** = 0x0010
- static final int **MSR_SUCCESS** = 0x0011
- static final int **TRANSACTION_CANCELED** = 0x0012
- static final int **CTLS_TWO_CARDS** = 0x007A
- static final int **CTLS_TERMINATE** = 0x007E
- static final int **CTLS_TERMINATE_TRY_ANOTHER** = 0x007D
- static final int **MSR_SWIPE_CAPTURED** = 0x0080
- static final int **REQUEST_ONLINE_PIN** = 0x0081
- static final int **REQUEST_SIGNATURE** = 0x0082
- static final int **FALLBACK_TO_CONTACT** = 0x0083
- static final int **FALLBACK_TO_OTHER** = 0x0084
- static final int **REVERSAL_REQUIRED** = 0x0085
- static final int **ADVISE_REQUIRED** = 0x0086
- static final int **ADVISE_REVERSAL_REQUIRED** = 0x0087
- static final int **NO_ADVISE_REVERSAL_REQUIRED** = 0x0088
- static final int **UNABLE_TO_REACH_HOST** = 0x00FF
- static final int **FILE_ARG_INVALID** = 0x1001
- static final int **FILE_OPEN_FAILED** = 0x1002
- static final int **FILE_OPERATION_FAILED** = 0X1003
- static final int **MEMORY_NOT_ENOUGH** = 0x2001
- static final int **SMARTCARD_OK** = 0x3001
- static final int **SMARTCARD_FAIL** = 0x3002

- static final int **SMARTCARD_INIT_FAILED** = 0x3003
- static final int **FALLBACK_SITUATION** = 0x3004
- static final int **SMARTCARD_ABSENT** = 0x3005
- static final int **SMARTCARD_TIMEOUT** = 0x3006
- static final int **MSR_CARD_ERROR** = 0x3007
- static final int **MSR_CARD_ERROR_FALLBACK** = 0x3012
- static final int **TIMEOUT_FOR_WAITING_ICC_INSERT_OR_MSR_SWIPE_FALLBACK** = 0x3013
- static final int **PARSING_TAGS_FAILED** = 0X5001
- static final int **CARD_DATA_ELEMENT_DUPLICATE** = 0X5002
- static final int **DATA_FORMAT_INCORRECT** = 0X5003
- static final int **APP_NO_TERM** = 0X5004
- static final int **APP_NO_MATCHING** = 0X5005
- static final int **AMANDATORY_OBJECT_MISSING** = 0X5006
- static final int **APP_SELECTION_RETRY** = 0X5007
- static final int **AMOUNT_ERROR_GET** = 0X5008
- static final int **CARD_REJECTED** = 0X5009
- static final int **AIP_NOT_RECEIVED** = 0X5010
- static final int **AFL_NOT_RECEIVEDE** = 0X5011
- static final int **AFL_LEN_OUT_OF_RANGE** = 0X5012
- static final int **SFI_OUT_OF_RANGE** = 0X5013
- static final int **AFL_INCORRECT** = 0X5014
- static final int **EXP_DATE_INCORRECT** = 0X5015
- static final int **EFF_DATE_INCORRECT** = 0X5016
- static final int **ISS_COD_TBL_OUT_OF_RANGE** = 0X5017
- static final int **CRYPTOGRAM_TYPE_INCORRECT** = 0X5018
- static final int **PSE_BY_CARD_NOT_SUPPORTED** = 0X5019
- static final int **USER_LANGUAGE_SELECTED** = 0X5020
- static final int **SERVICE_NOT_ALLOWED** = 0X5021
- static final int **NO_TAG_FOUND** = 0X5022
- static final int **CARD_BLOCKED** = 0X5023
- static final int **LEN_INCORRECT** = 0X5024
- static final int **CARD_COM_ERROR** = 0X5025
- static final int **TSC_NOT_INCREASED** = 0X5026
- static final int **HASH_INCORRECT** = 0X5027
- static final int **ARC_NOT_PRESENCED** = 0X5028
- static final int **ARC_INVALID** = 0X5029
- static final int **COMM_NO_ONLINE** = 0X5030
- static final int **TRAN_TYPE_INCORRECT** = 0X5031
- static final int **APP_NO_SUPPORT** = 0X5032
- static final int **APP_NOT_SELECT** = 0X5033
- static final int **LANG_NOT_SELECT** = 0X5034
- static final int **TERM_DATA_NOT_PRESENCED** = 0X5035
- static final int **PIN_ENTRY_TIMEOUT** = 0X5039
- static final int **CVM_TYPE_UNKNOWN** = 0X6001
- static final int **CVM_AIP_NOT_SUPPORTED** = 0X6002
- static final int **CVM_TAG_8E_MISSING** = 0X6003
- static final int **CVM_TAG_8E_FORMAT_ERROR** = 0X6004
- static final int **CVM_CODE_IS_NOT_SUPPORTED** = 0X6005
- static final int **CVM_COND_CODE_IS_NOT_SUPPORTED** = 0X6006
- static final int **CVM_NO_MORE** = 0X6007
- static final int **PIN_BYPASSED_BEFORE** = 0X6008
- static final int **UNKONWN** = 0xffff

13.3.1 Detailed Description

This class provides all information for emv transaction data.
Application can get the card data by calling the Properties of class IDTEMVData when received by emv callbak.

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

- Source_Android/IDTEMVData.java

13.4 com.idtechproducts.device.IDTLCDData Class Reference

Public Attributes

- String screenName
- int screenID
- String objectName
- int objectID
- int topLeftX
- int topLeftY
- int bottomRightX
- int bottomRightY
- int longPressed
- byte [] asyncLCDMessage
- int result

13.4.1 Member Data Documentation

13.4.1.1 asyncLCDMessage

```
byte [] com.idtechproducts.device.IDTLCDData.asyncLCDMessage
```

The async LCD message for contactless transactions.

13.4.1.2 bottomRightX

```
int com.idtechproducts.device.IDTLCDData.bottomRightX
```

Bottom-Right x

13.4.1.3 bottomRightY

```
int com.idtechproducts.device.IDTLCDData.bottomRightY
```

Bottom-Right y

13.4.1.4 longPressed

```
int com.idtechproducts.device.IDTLCDData.longPressed
```

The flag indicates if the button is long pressed.

13.4.1.5 objectID

```
int com.idtechproducts.device.IDTLCDData.objectID
```

Object ID.

13.4.1.6 objectName

```
String com.idtechproducts.device.IDTLCDData.objectName
```

Object name.

13.4.1.7 result

```
int com.idtechproducts.device.IDTLCDData.result
```

The checking result.

13.4.1.8 screenID

```
int com.idtechproducts.device.IDTLCDData.screenID
```

Screen ID.

13.4.1.9 screenName

```
String com.idtechproducts.device.IDTLCDData.screenName
```

Screen name.

13.4.1.10 topLeftX

```
int com.idtechproducts.device.IDTLCDData.topLeftX
```

Top-Left x

13.4.1.11 topLeftY

```
int com.idtechproducts.device.IDTLCDData.topLeftY
```

Top-Left y

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

- Source_Android/IDTLCDData.java

13.5 com.idtechproducts.device.IDTMSRData Class Reference

Public Attributes

- EVENT_MSR_Types event
- byte **cardDataFlag**
- boolean isCTLS
- byte [] cardData
- byte **t1DecodeStatus**

- byte **t2DecodeStatus**
- byte **t3DecodeStatus**
- byte [] encTrack1
- byte [] encTrack2
- byte [] encTrack3
- String track1
- String track2
- String track3
- byte [] serialNumber
- byte [] KSN
- int track1Length
- int track2Length
- int track3Length
- boolean iccPresent
- CAPTURE_ENCODE_TYPE cardType
- CTLS_APPLICATION ctlsApplication
- byte [] optionalBytes
- byte captureEncodeStatus
- CAPTURE_ENCRYPT_TYPE captureEncryptType
- byte hasDE055
- int DE055Len
- byte [] DE055Data
- int TLVLen
- byte [] TLVData
- byte [] rawTrackData
- Map< String, byte[]> unencryptedTags
- Map< String, byte[]> encryptedTags
- Map< String, byte[]> maskedTags
- int result = ErrorCode.SUCCESS
- String fastEMV = null

13.5.1 Detailed Description

This class provides all information of card data.
Application can get the card data by calling the Properties of class IDTMSRData when finish swiping.

13.5.2 Member Data Documentation

13.5.2.1 captureEncodeStatus

```
byte com.idtechproducts.device.IDTMSRData.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;

13.5.2.2 captureEncryptType

```
CAPTURE_ENCRYPT_TYPE com.idtechproducts.device.IDTMSRData.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;

13.5.2.3 cardData

```
byte [] com.idtechproducts.device.IDTMSRData.cardData
```

Get the swiped card data.
Containing complete unparsed swipe data as received from MSR.
NOTE:
Just refer to this item cardData if the card data is the clear data.

13.5.2.4 cardType

```
CAPTURE_ENCODE_TYPE com.idtechproducts.device.IDTMSRData.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

13.5.2.5 ctlsApplication

```
CTLS_APPLICATION com.idtechproducts.device.IDTMSRData.ctlsApplication
```

CTLS Application

13.5.2.6 DE055Data

```
byte [] com.idtechproducts.device.IDTMSRData.DE055Data
```

Get the swiped card of DE055 data.

13.5.2.7 DE055Len

```
int com.idtechproducts.device.IDTMSRData.DE055Len
```

Get the swiped card length of DE055 data.

13.5.2.8 encryptedTags

```
Map<String, byte[]> com.idtechproducts.device.IDTMSRData.encryptedTags
```

Encrypted card data provided via TLV.

13.5.2.9 encTrack1

```
byte [] com.idtechproducts.device.IDTMSRData.encTrack1
```

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

13.5.2.10 encTrack2

```
byte [] com.idtechproducts.device.IDTMSRData.encTrack2
```

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

13.5.2.11 encTrack3

```
byte [] com.idtechproducts.device.IDTMSRData.encTrack3
```

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

13.5.2.12 event

```
EVENT_MSR_Types com.idtechproducts.device.IDTMSRData.event
```

MSR type,please see EVENT_MSR_Types for more information.

13.5.2.13 fastEMV

```
String com.idtechproducts.device.IDTMSRData.fastEMV = null
```

Fast EMV String.

13.5.2.14 hasDE055

```
byte com.idtechproducts.device.IDTMSRData.hasDE055
```

The flag to indicate the availability of the swiped card DE055 data.

13.5.2.15 iccPresent

```
boolean com.idtechproducts.device.IDTMSRData.iccPresent
```

Determines if ICC is present in card (service code starts with "2" or "6".

13.5.2.16 isCTLS

```
boolean com.idtechproducts.device.IDTMSRData.isCTLS
```

Track data was captured via CTLS interface

13.5.2.17 KSN

```
byte [] com.idtechproducts.device.IDTMSRData.KSN
```

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

13.5.2.18 maskedTags

```
Map<String, byte[]> com.idtechproducts.device.IDTMSRData.maskedTags
```

Masked card data provided via TLV.

13.5.2.19 optionalBytes

```
byte [] com.idtechproducts.device.IDTMSRData.optionalBytes
```

Get optional bytes of the swiped card data.

13.5.2.20 rawTrackData

```
byte [] com.idtechproducts.device.IDTMSRData.rawTrackData
```

Get the DFEE23 MSR raw data.

13.5.2.21 result

```
int com.idtechproducts.device.IDTMSRData.result = ErrorCode.SUCCESS
```

Return error code.

13.5.2.22 serialNumber

```
byte [] com.idtechproducts.device.IDTMSRData.serialNumber
```

Get the Reader Serial Number.

13.5.2.23 TLVData

```
byte [] com.idtechproducts.device.IDTMSRData.TLVData
```

Get the swiped card TLV data.

13.5.2.24 TLVLen

```
int com.idtechproducts.device.IDTMSRData.TLVLen
```

Get the swiped card length of TLV data.

13.5.2.25 track1

```
String com.idtechproducts.device.IDTMSRData.track1
```

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

13.5.2.26 track1Length

```
int com.idtechproducts.device.IDTMSRData.track1Length
```

Get the swiped card length of Track1 data.

13.5.2.27 track2

```
String com.idtechproducts.device.IDTMSRData.track2
```

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

13.5.2.28 track2Length

```
int com.idtechproducts.device.IDTMSRData.track2Length
```

Get the swiped card length of Track2 data.

13.5.2.29 track3

```
String com.idtechproducts.device.IDTMSRData.track3
```

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

13.5.2.30 track3Length

```
int com.idtechproducts.device.IDTMSRData.track3Length
```

Get the swiped card length of Track3 data.

13.5.2.31 unencryptedTags

```
Map<String, byte[]> com.idtechproducts.device.IDTMSRData.unencryptedTags
```

Unencrypted card data provided via TLV.

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

- Source_Android/IDTMSRData.java

13.6 com.idtechproducts.device.OnReceiverListener Interface Reference

Classes

- enum EMV_RESULT_CODE_Types

Public Member Functions

- void swipeMSRData (IDTMSRData card)
- void lcdDisplay (int mode, String[] lines, int timeout)
- void lcdDisplay (int mode, String[] lines, int timeout, byte[] languageCode, byte messageId)
- void ctlsEvent (byte event, byte scheme, byte data)
- void emvTransactionData (IDTEMVData emvData)
- void deviceConnected ()
- void deviceDisconnected ()
- void timeout (int errorCode)
- void autoConfigCompleted (StructConfigParameters profile)
- void autoConfigProgress (int progressValue)
- void msgRKICompleted (String MACResult)

- void ICCNotifyInfo (byte[] dataNotify, String strMessage)
- void msgBatteryLow ()
- void LoadXMLConfigFailureInfo (int index, String strMessage)
- void msgToConnectDevice ()
- void msgAudioVolumeAjustFailed ()
- void dataInOutMonitor (byte[] data, boolean isIncoming)

13.6.1 Detailed Description

The interface includes the callback functions for card data, PIN data and EMV data. The android activity should implement this interface then implement callback functions.

13.6.2 Member Function Documentation

13.6.2.1 autoConfigCompleted()

```
void com.idtechproducts.device.OnReceiverListener.autoConfigCompleted (
            StructConfigParameters profile )
```

The auto config process finished, and succeeded to get one profile to connect the device.

13.6.2.2 autoConfigProgress()

```
void com.idtechproducts.device.OnReceiverListener.autoConfigProgress (
            int progressValue )
```

The auto config process percent value.

13.6.2.3 ctlsEvent()

```
void com.idtechproducts.device.OnReceiverListener.ctlsEvent (
            byte event,
            byte scheme,
            byte data )
```

Contactless Event Asynchronous UI Message Event

Parameters

event	Asynchronous UI Message Event: • 0x01: LED event • 0x02: Buzzer event • 0x03: LCD event
scheme	• 0x00: ViVOtech UI Scheme • 0x02: VisaWave UI Scheme • 0x03: EMEA UI Scheme

Parameters

data	Event Data: For LED event: Higher nibble: LED # 00: LED0 01: LED1 02: LED2 03: LED3 FF: all Lower nibble: 00: Off 01: On 11: No change For Buzzer event: Higher nibble: 1: short beeps 2: long beeps Lower nibble, short beep: 0: No change 1: Single beep 2: Double beep 3: Triple beep Lower nibble, long beep: 0: 200ms 1: 400ms 2: 600ms For LCD event: LCD message index

13.6.2.4 dataInOutMonitor()

```
void com.idtechproducts.device.OnReceiverListener.dataInOutMonitor (
            byte [] data,
            boolean isIncoming )
```

The input/output data notification,

Parameters

data	the input/output data.
isIncoming	true if is incoming data, false if it is out going data.

13.6.2.5 deviceConnected()

```
void com.idtechproducts.device.OnReceiverListener.deviceConnected ( )
```

Fires when device connects.

13.6.2.6 deviceDisconnected()

```
void com.idtechproducts.device.OnReceiverListener.deviceDisconnected ( )
```

Fires when device disconnects.

13.6.2.7 emvTransactionData()

```
void com.idtechproducts.device.OnReceiverListener.emvTransactionData (
            IDTEMVData emvData )
```

EMV Transaction Data

This protocol will receive results from IDT_Device::startEMVTransaction:otherAmount:timeout:cashback↩ :additionalTags:()

Parameters

emvData	EMV Results Data. Result code, card type, encryption type, masked tags, encrypted tags, unencrypted tags and KSN

13.6.2.8 ICCNotifyInfo()

```
void com.idtechproducts.device.OnReceiverListener.ICCNotifyInfo (
            byte [] dataNotify,
            String strMessage )
```

The ICC Card seated status notification,

Parameters

dataNotify	the response data.
strMessage,the	ICC notification message information.

13.6.2.9 lcdDisplay() [1/2]

```
void com.idtechproducts.device.OnReceiverListener.lcdDisplay (
            int mode,
            String [] lines,
            int timeout )
```

LCD Display Request During an EMV transaction, this delegate will receive data to clear virtual LCD display, display messages, display menu, or display language. Applies to UniPay III

Parameters

mode	LCD Display Mode: • 0x01: Menu Display. A selection must be made to resume the transaction • 0x02: Normal Display get function key. A function must be selected to resume the transaction • 0x03: Display without input. Message is displayed without pausing the transaction • 0x04: List of languages are presented for selection. A selection must be made to resume the transaction • 0x10: Clear Screen. Command to clear the LCD screen
lines	Line(s) of data to display
timeout	Timeout value when displaying dialog box

13.6.2.10 lcdDisplay() [2/2]

```
void com.idtechproducts.device.OnReceiverListener.lcdDisplay (
            int mode,
            String [] lines,
            int timeout,
            byte [] languageCode,
            byte messageId )
```

LCD Display Request During an EMV transaction, this delegate will receive data to clear virtual LCD display, display messages, display menu, or display language. Applies to UniPay III

Parameters

mode	LCD Display Mode: • 0x01: Menu Display. A selection must be made to resume the transaction • 0x02: Normal Display get function key. A function must be selected to resume the transaction • 0x03: Display without input. Message is displayed without pausing the transaction • 0x04: List of languages are presented for selection. A selection must be made to resume the transaction • 0x10: Clear Screen. Command to clear the LCD screen
lines	Line(s) of data to display
timeout	Timeout value when displaying dialog box
languageCode	2 bytes language code ("EN", "ES", "FR", or "ZH") of the LCD message.
messageId	1 byte id (from 1 to 34) for a LCD message string.

13.6.2.11 LoadXMLConfigFailureInfo()

```
void com.idtechproducts.device.OnReceiverListener.LoadXMLConfigFailureInfo (
            int index,
            String strMessage )
```

Get the user grant to continue process ,

Parameters

index	1: "This phone model is not supported by the current SDK. Please contact supporter for assistance."; 2: "Wrong XML file name, please set the filename or enable the auto update."; 3: "The XML file does not exist and the auto update disabled."; 4: "Can't download the XML file. Please make sure the network is accessible.";
strMessage,the	message information when loading the XML file.

13.6.2.12 msgAudioVolumeAjustFailed()

```
void com.idtechproducts.device.OnReceiverListener.msgAudioVolumeAjustFailed ( )
```

The message notify the application failed to adjust the audio volume.

Parameters

strMessage,the	message of description about the failure info when to adjust the audio volume.

13.6.2.13 msgBatteryLow()

```
void com.idtechproducts.device.OnReceiverListener.msgBatteryLow ( )
```

Battery low status notification,

13.6.2.14 msgRKICompleted()

```
void com.idtechproducts.device.OnReceiverListener.msgRKICompleted (
            String MACResult )
```

RKI succeeded; MAC result as return value.

13.6.2.15 msgToConnectDevice()

```
void com.idtechproducts.device.OnReceiverListener.msgToConnectDevice ( )
```

The message notify the application to connect the device.

13.6.2.16 swipeMSRData()

```
void com.idtechproducts.device.OnReceiverListener.swipeMSRData (
            IDTMSRData card )
```

Call back function,this function will be called automatically if Card decode has been completed after swiping card.

Parameters

card	the MSR data. Card data.It is encrypted data and format is following: 1. Data Length low byte - 1 byte; 2. Data length high byte - 1 byte;

1. Card Encode Type - 1 byte.0x00/0x80-ISO/ABA format,0x01/0x81-AAMVA format,0x03/0x83-Other and 0x04/0x84-undecoded format.

2. Track1∼3 Status - 1 byte.Bit0,1,2:Track1∼3 decode and Bit3,4,5:Track1∼3 sampling.

3. Track1 data length - 1 byte.This length is the plain card data's length.

4. Track2 data length - 1 byte.

5. Track3 data length - 1 byte.

6. Clear/mask data sent status - 1 byte.
   Bit0:1–Track1 clear/mask status present,0–not present.
   Bit1:1–Track2 clear/mask status present,0–not present.
   Bit2:1–Track1 clear/mask status present,0–not present.
   Bit3∼Bit7:Reserved.Set to 0.

9.Encrypted/Hash data sent status - 1 byte.
Bit0:1–Track1 encrypted data present.
Bit1:1–Track2 encrypted data present.
Bit2:1–Track3 encrypted data present.
Bit3:1–Track1 hash data present.
Bit4:1–Track2 hash data present.
Bit5:1–Track3 hash data present.
Bit0:0.
Bit7:1–KSN present.

7. Track1 clear/mask data – Var bytes.
8. Track2 clear/mask data – Var bytes.
9. Track3 clear/mask data – Var bytes.
10. Track1 encrypted data – Var bytes.
11. Track2 encrypted data – Var bytes.
12. Track3 encrypted data – Var bytes.
13. Track1 hash data – 20 bytes if exist.
14. Track2 hash data – 20 bytes if exist.
15. Track3 hash data – 20 bytes if exist.
16. KSN – 10 bytes.

13.6.2.17 timeout()

```
void com.idtechproducts.device.OnReceiverListener.timeout (
            int errorCode )
```

Notify the plug status of phone jack. Timeout when wait for the response.
This happens in the process of get PINpad, swipe MSR, EMV Level 2 transaction

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

- Source_Android/OnReceiverListener.java

13.7 com.idtechproducts.device.OnReceiverListenerLCD Interface Reference

Public Member Functions

- void lcdData (IDTLCDData lcdData)

13.7.1 Detailed Description

The interface includes the callback functions when the kernel is returning an LCD event

13.7.2 Member Function Documentation

13.7.2.1 lcdData()

```
void com.idtechproducts.device.OnReceiverListenerLCD.lcdData (
            IDTLCDData lcdData )
```

LCD event During an EMV transaction, this delegate will receive data that is a request to collect a PIN

Parameters

mode	PIN Mode: • EMV_PIN_MODE_CANCEL = 0X00, • EMV_PIN_MODE_ONLINE_PIN_DUKPT = 0X01, • EMV_PIN_MODE_ONLINE_PIN_MKSK = 0X02, • EMV_PIN_MODE_OFFLINE_PIN = 0X03
key	Either DUKPT or SESSION, depending on mode. If offline plaintext, value is nil
PAN	PAN for calculating PINBlock
startTO	Timeout value to start PIN entry
intervalTO	Timeout value between key presses
language	"EN"=English, "ES"=Spanish, "ZH"=Chinese, "FR"=French

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

- Source_Android/OnReceiverListenerLCD.java

13.8 com.idtechproducts.device.OnReceiverListenerPIN Interface Reference

Public Member Functions

- void pinpadData (byte[] data)

13.8.1 Detailed Description

The interface includes the callback functions for PIN Pad Data that needs to be sent to the attached device.

13.8.2 Member Function Documentation

13.8.2.1 pinpadData()

```
void com.idtechproducts.device.OnReceiverListenerPIN.pinpadData (
            byte [] data )
```

When PIN functions with second response are called, the data of the second response will be passed to this call back function

Parameters

data	the data that was returned by the second response of the PIN function.

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

- Source_Android/OnReceiverListenerPIN.java

13.9 com.idtechproducts.device.OnReceiverListenerPINRequest Interface Reference

Public Member Functions

- void pinRequest (int mode, byte[] key, byte[] PAN, int startTO, int intervalTO, String language)

13.9.1 Detailed Description

The interface includes the callback functions when the kernel is requesting PIN entry

13.9.2 Member Function Documentation

13.9.2.1 pinRequest()

```
void com.idtechproducts.device.OnReceiverListenerPINRequest.pinRequest (
            int mode,
            byte [] key,
            byte [] PAN,
            int startTO,
            int intervalTO,
            String language )
```

PIN Request During an EMV transaction, this delegate will receive data that is a request to collect a PIN

Parameters

mode	PIN Mode: • EMV_PIN_MODE_CANCEL = 0X00, • EMV_PIN_MODE_ONLINE_PIN_DUKPT = 0X01, • EMV_PIN_MODE_ONLINE_PIN_MKSK = 0X02, • EMV_PIN_MODE_OFFLINE_PIN = 0X03
key	Either DUKPT or SESSION, depending on mode. If offline plaintext, value is nil

Parameters

PAN	PAN for calculating PINBlock
startTO	Timeout value to start PIN entry
intervalTO	Timeout value between key presses
language	"EN"=English, "ES"=Spanish, "ZH"=Chinese, "FR"=French

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

- Source_Android/OnReceiverListenerPINRequest.java

Index