IDTECH®

Value through Innovation

Android SDK Guide

for Augusta

#80145504-001

Rev. A

Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;Augusta;SDK;Android	4/06/2016

Contents

Chapter 1

IDTech Android SDK Reference Guide for Augusta

Universal_SDK_X.XX.XXX.jar is an Android library that will be provided by IDTech as the main interface between Android applications, the Augusta 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.

- Augusta Error Code Reference
- LCD Foreign Language Mapping Table

Chapter 2

Connecting with Augusta

The Augusta connects through USB on Android.

2.1 Connect with USB

The Augusta 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 Augusta (mini-USB port on left side) to the Android host adapter cable.

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

Augusta Main Transaction Commands

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

4.1 EMV Methods

Start EMV Transaction

com.idtechproducts.device.IDT_Augusta.emv_startTransaction()

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

com.idtechproducts.device.IDT_Augusta.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_Augusta.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_Augusta.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_Augusta.emv_retrieveTerminalData()
com.idtechproducts.device.IDT_Augusta.emv_removeTerminalData()
com.idtechproducts.device.IDT_Augusta.emv_setTerminalData()

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

AID Management

com.idtechproducts.device.IDT_Augusta.emv_retrieveApplicationData:response()
com.idtechproducts.device.IDT_Augusta.emv_removeApplicationData()
com.idtechproducts.device.IDT_Augusta.emv_setApplicationData:configData()
com.idtechproducts.device.IDT_Augusta.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_Augusta.emv_retrieveCAPK)
com.idtechproducts.device.IDT_Augusta.emv_removeCAPK()
com.idtechproducts.device.IDT_Augusta.emv_setCAPK()
com.idtechproducts.device.IDT_Augusta.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_Augusta.emv_retrieveCRL()
com.idtechproducts.device.IDT_Augusta.emv_removeCRL()
com.idtechproducts.device.IDT_Augusta.emv_setCRL()

Methods for Certificate Revocation List management.

APDU Communication

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

com.idtechproducts.device.IDT_Augusta.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

Request Swipe

com.idtechproducts.device.IDT_Augusta.msr_startMSRSwipe()

Cancel Swipe

com.idtechproducts.device.IDT_Augusta.msr_cancelMSRSwipe()

Chapter 5

EMV Callback

During an EMV transaction, without a built-in LCD display on the Augusta, 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_Augusta::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

Core Implementation Augusta: Android

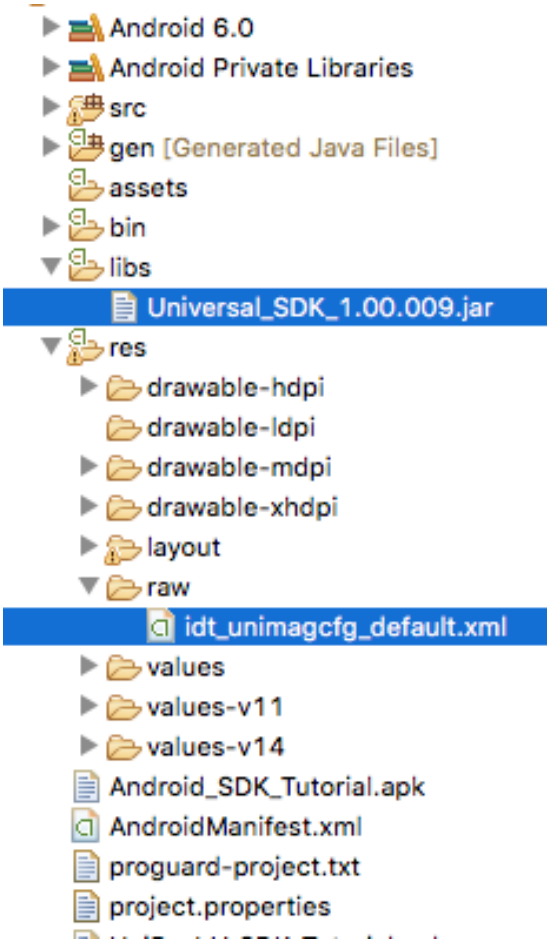
Universal_SDK_X.XX.XXX.jar includes class libraries to interface with the Augusta. This guide assume a fair understanding of Eclipse IDE and general Android programming knowledge.

6.1 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 Augusta objects
- Sample Project Tutorial

6.2 Import the necessary libraries

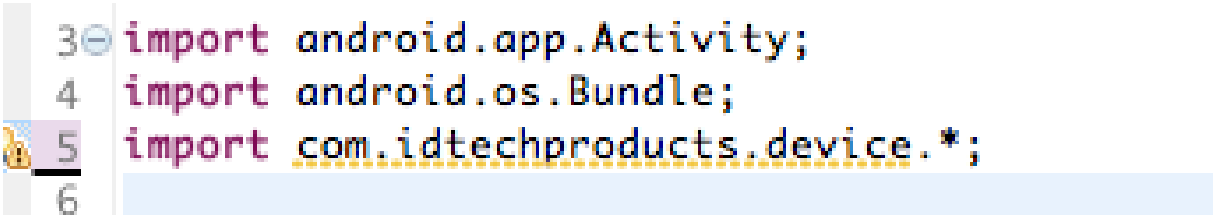
Communicating with Augusta requires the Universal_SDK_X.XX.XXX.jar file be added to the project. While build paths can be created pointing to the Universal_SDK_X.XX.XXX.jar file, the simplest solution is to add the jar file to the projects libs folder. In addition, the device .xml file needs to be accessable to the project. Example code will be given on how to retrieve this file from the Resource RAW folder:



6.3 Add Import statements to utilize libraries

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

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



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.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.*;
```

6.4 Implement OnReceiverListener for the activity:

In the class that will be a delegate of Augusta, 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

    }
}
```

6.5 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" />
```

6.6 Allocate/initialize Augusta objects:

Initialize IDT_Device object by passing context and OnReceiverListener delegate. Load the XML file for more device compatibility

```
// declaring the instance of the UniPayReader;
private IDT_Augusta myUniPayReader = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    if(myUniPayReader!=null){
        myUniPayReader.unregisterListen();
        myUniPayReader.release();
        myUniPayReader = null;
    }
    myUniPayReader = new IDT_Augusta(this,getActivity());
    myUniPayReader.registerListen();
    loadXMLfile()
}


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 configuratin 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.
    myUniPayReader.config_setXMLFileNameWithPath(fileNameWithPath);
    Log.d("Demo Info >>>>>","loadingConfigurationXMLFile begin.");
    myUniPayReader.config_loadingConfigurationXMLFile(true);
}
```

6.7 Sample Project Tutorial

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

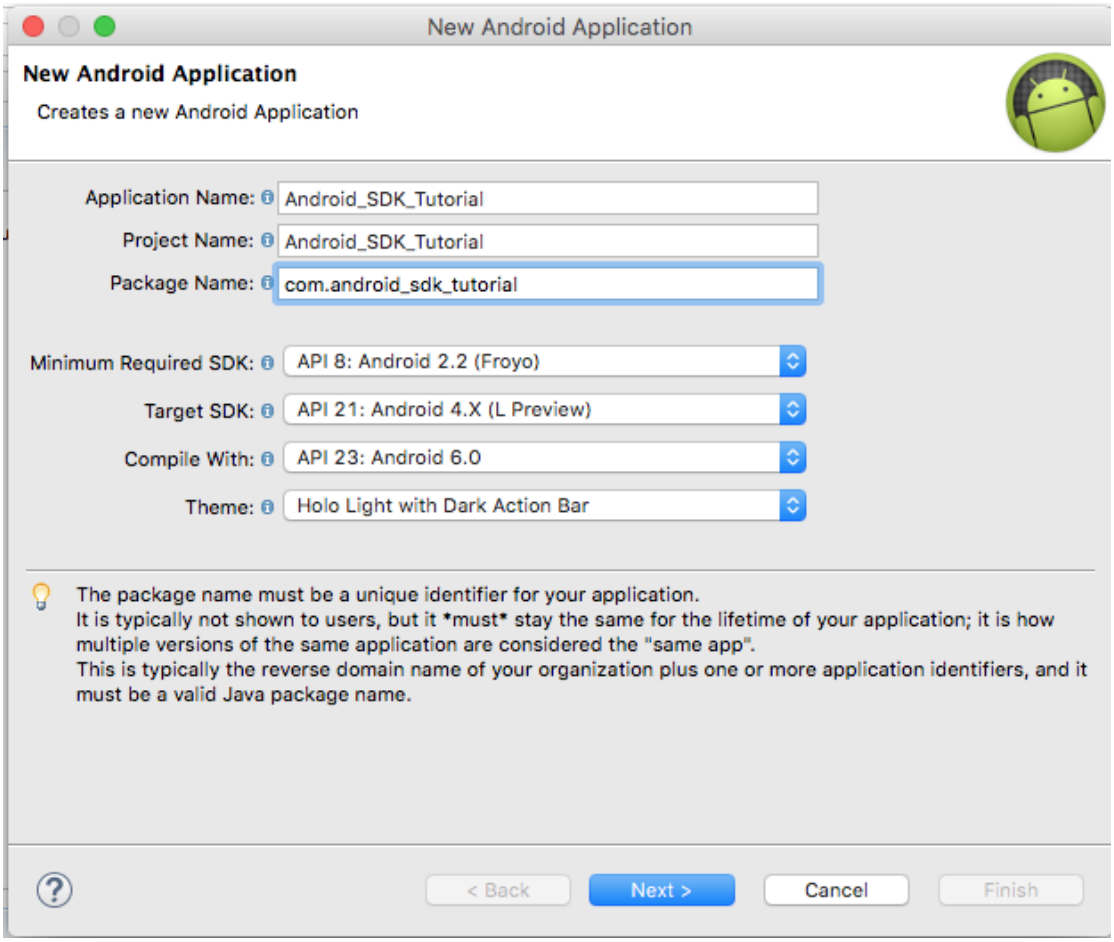
- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for MSR
- Start/Cancel EMV Transaction
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

Listeners:

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

6.7.1 Step 1: Create New Project

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



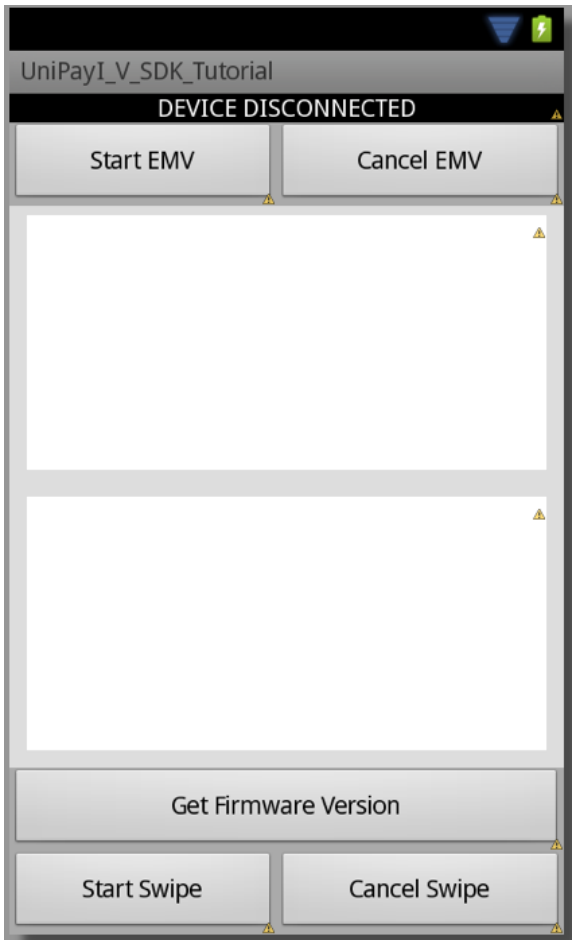
6.7.2 Step 2: Import IDTechSDK for Augusta

Import the necessary libraries

6.7.3 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 Augusta 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
  - Start ICC EMV
  - Complete ICC EMV
  - Cancel Transaction



6.7.4 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="Augusta 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" />
        <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" />
    </LinearLayout>
</LinearLayout>
```

6.7.5 Step 5: Configure Method File

In the activity file, perform the following:

- set delegate and initialize Augusta object in the onCreate method. Reference: Allocate/initialize Augusta objects

```
// declaring the instance of the UniPayReader;
private IDT_Augusta myUniPayReader = 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(myUniPayReader!=null){
        myUniPayReader.unregisterListen();
        myUniPayReader.release();
        myUniPayReader = null;
    }
    myUniPayReader = new IDT_Augusta(this,this);
    myUniPayReader.registerListen();
    loadXMLfile();
}
```

- 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("Augusta DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("Augusta 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 data didn't read correctly";
    else
        info = "Swipe Read Successfully";
    detail = Common.parse_MSRData(myUniPayReader.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};
         myUniPayReader.emv_completeTransaction(false, response, null, null,null);
     }
     else if (emvData.result == IDTEMVData.START_TRANS_SUCCESS){
           //Auto Authenticate
         myUniPayReader.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
                myUniPayReader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else if (mode == 0x08) //Language Menu Display
            {
                //automatically select first language
                myUniPayReader.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 = myUniPayReader.device_getFirmwareVersion(sb);
        if (ret == ErrorCode.SUCCESS) {
            info += "Firmware Version: " + sb.toString();
            detail = "";
            handler.post(doUpdateStatus);
        }
        else {
            info += "GetFirmwareVersion: Failed\n";
            info += "Status: "+ myUniPayReader.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_Augusta.emv_allowFallback(true);
        myUniPayReader.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();
        myUniPayReader.emv_cancelEMVTransaction(resData);
    }
});

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

```
        }
    });

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

6.7.6 Complete code listing

```
package com.example.unipayi_v_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.
      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 UniPayReader;
    private IDT_Augusta myUniPayReader = 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(myUniPayReader!=null){
            myUniPayReader.unregisterListen();
            myUniPayReader.release();
            myUniPayReader = null;
        }
        myUniPayReader = new IDT_Augusta(this,this);
        myUniPayReader.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 = myUniPayReader.device_getFirmwareVersion(sb);
                if (ret == ErrorCode.SUCCESS) {
                    info += "Firmware Version: " + sb.toString();
                    detail = "";
                    handler.post(doUpdateStatus);
                }
                else {
                    info += "GetFirmwareVersion: Failed\n";
                    info += "Status: "+ myUniPayReader.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_Augusta.emv_allowFallback(true);
                myUniPayReader.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();
                myUniPayReader.emv_cancelEMVTransaction(resData);
            }
        });

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

        btnCancelSwipe.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View v) {
                detail = "";
                info = "Cancelling Swipe Transaction\n";
                handler.post(doUpdateStatus);
                myUniPayReader.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("Augusta DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("Augusta 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};
         myUniPayReader.emv_completeTransaction(false, response, null, null,null);
     }
     else if (emvData.result == IDTEMVData.START_TRANS_SUCCESS){
            //Auto Authenticate
         myUniPayReader.emv_authenticateTransaction(null);
        }

}

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

            if (mode == 0x01) //Menu Display
            {
                //automatically select 1st application
                myUniPayReader.emv_lcdControlResponse((byte)mode, (byte)0x01);
            }
            else if (mode == 0x08) //Language Menu Display
            {
```

```
                //automatically select first language
                myUniPayReader.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 data didn't read correctly";
        else
            info = "Swipe Read Successfully";
        detail = Common.parse_MSRData(myUniPayReader.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.
        myUniPayReader.config_setXMLFileNameWithPath(fileNameWithPath);
        Log.d("Demo Info >>>>>","loadingConfigurationXMLFile begin.");
        myUniPayReader.config_loadingConfigurationXMLFile(true);
    }




}
```

Chapter 7

Augusta 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 8

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_UniPayIII_IOS,
    IDT_DEVICE_UniPayIII_OSX_USB,
    IDT_DEVICE_UniPayI_V_IOS,
    IDT_DEVICE_UniPayI_V_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_UniPayIII_EMV_NO_ICC_MSR_DATA,
    EVENT_UniPayIII_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 9

EMV Tag Reference

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

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

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

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

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

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

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

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

Chapter 10

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 11

Class Index

11.1 Class List

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

Chapter 12

Class Documentation

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

Public Attributes

- **EMV_RESULT_CODE_OFFLINE_APPROVED**
- **EMV_RESULT_CODE_OFFLINE_DECLINED**
- **EMV_RESULT_CODE_APPROVED**
- **EMV_RESULT_CODE_DECLINED**
- **EMV_RESULT_CODE_GO_ONLINE**
- **EMV_RESULT_CODE_CALL_YOUR_BANK**
- **EMV_RESULT_CODE_NOT_ACCEPTED**
- **EMV_RESULT_CODE_USE_MAGSTRIPE**
- **EMV_RESULT_CODE_TIME_OUT**
- **EMV_RESULT_CODE_TRANSACTION_SUCCESS**
- **EMV_RESULT_CODE_TERMINATE**

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

- Source Android/OnReceiverListener.java

12.2 com.idtechproducts.device.IDT_Augusta Class Reference

Public Member Functions

- IDT_Augusta (OnReceiverListener callback, Context context, boolean isTTK, boolean isSRED, boolean is↩ Thales)
- boolean device_setDeviceType (ReaderInfo.DEVICE_TYPE deviceType, boolean isTTK, boolean isSRED, boolean isThales)
- boolean device_isTTK ()
- boolean device_isThales ()
- boolean device_isSRED ()
- IDT_Augusta (OnReceiverListener callback, Context context)
- boolean device_setDeviceType (ReaderInfo.DEVICE_TYPE deviceType)
- int device_getDRS (ResDataStruct respData)
- int device_verifyBackdoorKey ()
- int device_selfCheck ()

- int device_rebootDevice ()
- 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)
- boolean device_isConnected ()
- int device_startRKI ()
- 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 autoConfig_start (String strXMLFilename)
- void autoConfig_stop ()
- int device_getFirmwareVersion (StringBuilder version)
- int device_controlBeep (int index, int frequency, int duration)
- int device_controlLED (byte indexLED, byte control, int intervalOn, int intervalOff)
- int device_controlLED_ICC (int controlMode, int interval)
- int device_setDateTime (byte[] mac)
- int device_getKeyStatus (ResDataStruct respData)
- int config_setBeeperController (boolean firmwareControlBeeper)
- int config_setLEDController (boolean firmwareControlMSRLED, boolean firmwareControlICCLED)
- int config_setEncryptionControl (byte Encryption)
- int config_setEncryptionControl (boolean msr, boolean icc)
- int config_getEncryptionControl (ResDataStruct respData)
- int icc_setKeyTypeForICCDUKPT (byte encryption)
- int icc_getKeyTypeForICCDUKPT (ResDataStruct respData)
- int icc_setKeyFormatForICCDUKPT (byte encryption)
- int icc_getKeyFormatForICCDUKPT (ResDataStruct respData)
- int icc_enable (boolean withNotification)
- int icc_disable ()
- int icc_getFunctionStatus (ResDataStruct respData)
- int emv_getEMVKernelVersion (StringBuilder version)
- int emv_getEMVKernelCheckValue (ResDataStruct respData)
- int emv_getEMVConfigurationCheckValue (ResDataStruct respData)
- int emv_removeAllApplicationData ()
- int emv_removeAllCAPK ()
- int emv_removeAllCRL ()
- int emv_retrieveTransactionResult (byte[] tags, Map< String, Map< String, byte[]>> retrievedTags)
- int config_getSerialNumber (StringBuilder serialNumber)
- int config_getModelNumber (StringBuilder modNumber)
- int icc_getAPDU_KSN (byte KeyNameIndex, byte[] KeySlot, ResDataStruct resKSN)
- String device_getResponseCodeString (int errorCode)
- int device_sendDataCommand (String cmd, boolean calcLRC, String data, ResDataStruct respData, int timeout)
- int device_sendDataCommand (String cmd, boolean calcLRC, String data, ResDataStruct respData)

- int icc_getICCReaderStatus (ICCReaderStatusStruct ICCStatus)
- int icc_powerOnICC (ResDataStruct atrPPS)
- int icc_reviewAllSetting (ICCSettingStruct iccSetting)
- int icc_passthroughOnICC ()
- int icc_passthroughOffICC ()
- int icc_powerOffICC (ResDataStruct respData)
- int icc_exchangeAPDU (byte[] dataAPDU, APDUResponseStruct response)
- int emv_retrieveApplicationData (String aid, ResDataStruct respData)
- int emv_removeApplicationData (String aid, ResDataStruct respData)
- int emv_setApplicationData (String aid, 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[] data, 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_lcdControlResponse (byte mode, byte data)
- int emv_authenticateTransaction (byte[] tags)
- int emv_completeTransaction (boolean commError, byte[] authCode, byte[] iad, byte[] tlvScripts, byte[] tags)
- int msr_reviewAllSetting (MSRSettingStruct msrSetting)
- int msr_defaultAllSetting ()
- int msr_getSingleSetting (byte funcID, byte[] response)
- int msr_setSingleSetting (byte funcID, byte setData)
- int msr_cancelMSRSwipe ()
- int msr_startMSRSwipe ()
- int msr_startMSRSwipe (int timeout)
- int msr_setExpirationMask (boolean mask)
- int msr_getExpirationMask (ResDataStruct respData)
- int msr_setClearPANID (byte value)
- int msr_getClearPANID (ResDataStruct respData)
- int msr_getSwipeForcedEncryptionOption (ResDataStruct respData)
- int msr_setSwipeForcedEncryptionOption (boolean track1, boolean track2, boolean track3, boolean track3card0)
- int msr_getSwipeMaskOption (ResDataStruct respData)
- int msr_setSwipeMaskOption (boolean track1, boolean track2, boolean track3)
- int msr_getSetting (byte setting, ResDataStruct respData)
- int msr_setSetting (byte setting, byte val)
- int msr_setSwipeEncryption (byte encryption)
- int msr_getSwipeEncryption (ResDataStruct respData)
- int msr_enableBufferMode (boolean isBufferMode, boolean withNotification)
- int msr_disable ()
- int msr_setWhiteList (byte[] val)
- int msr_RetrieveWhiteList (ResDataStruct respData)
- int msr_getFunctionStatus (ResDataStruct respData)
- int ctls_startTransaction ()
- int ctls_cancelTransaction ()

Static Public Member Functions

- static IDT_Device getSDKInstance ()
- static void useUSBIntentFilter ()
- static IDT_Device **getIDT_Device** ()
- 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 ()

12.2.1 Constructor & Destructor Documentation

12.2.1.1 IDT_Augusta() [1/2]

```
com.idtechproducts.device.IDT_Augusta.IDT_Augusta (
            OnReceiverListener callback,
            Context context,
            boolean isTTK,
            boolean isSRED,
            boolean isThales )
```

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

Parameters

callback	OnReceiverListener callback
context	Application context
isTTK	True if TTK device
isSRED	True if SRED device
isThales	True if Thales device

12.2.1.2 IDT_Augusta() [2/2]

```
com.idtechproducts.device.IDT_Augusta.IDT_Augusta (
            OnReceiverListener callback,
            Context context )
```

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

Parameters

callback	OnReceiverListener callback
context	Application context

12.2.2 Member Function Documentation

12.2.2.1 autoConfig_start()

```
int com.idtechproducts.device.IDT_Augusta.autoConfig_start (
            String strXMLFilename )
```

start Auto Config to search the profile.

Parameters

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

12.2.2.2 autoConfig_stop()

```
void com.idtechproducts.device.IDT_Augusta.autoConfig_stop ( )
```

stop Auto Config.

Returns

null.

12.2.2.3 config_getEncryptionControl()

```
int com.idtechproducts.device.IDT_Augusta.config_getEncryptionControl (
            ResDataStruct respData )
```

Get Encryption Control

Get Encryption Control to switch status between MSR and ICC/EMV function. Following Encryption status supported:

- MSR ON, ICC/EMV ON,
- MSR ON, ICC/EMV OFF,
- MSR OFF, ICC/EMV OFF,

Parameters

respData	Response Body is 78 01 07 02 <Option> <00> The Option is stored in the variable encryptionOption: -bit 0: • true: enabled MSR with Encryption, • false: disabled MSR with Encryption, -bit 1: • true: enabled ICC with Encryption, • false: disabled ICC with Encryption,

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.4 config_getModelNumber()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.5 config_getSDKVersion()

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

READER CONFIG API LIST Get the version of SDK.

Parameters

sdkVersion	for version string.

Returns

success or error code.

See also

ErrorCode

12.2.2.6 config_getSerialNumber()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.7 config_getXMLVersionInfo()

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

Get XML configuration version.

Returns

the version info.

12.2.2.8 config_loadingConfigurationXMLFile()

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

Load XML Configuration File.

Parameters

xmlFilename,XML	Configuration File Name.

Returns

none

12.2.2.9 config_setBeeperController()

```
int com.idtechproducts.device.IDT_Augusta.config_setBeeperController (
            boolean firmwareControlBeeper )
```

Set the Beeper Controller Set the Beeper controlled by software or firmware

Parameters

firmwareControlBeeper	true means firmware control the beeper, false means software control beeper.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.10 config_setEncryptionControl() [1/2]

```
int com.idtechproducts.device.IDT_Augusta.config_setEncryptionControl (
            byte Encryption )
```

Set Encryption Control

Set Encryption Control to switch status between MSR and ICC/EMV function. Following Encryption status supported:

- MSR ON, ICC/EMV ON,
- MSR ON, ICC/EMV OFF,
- MSR OFF, ICC/EMV OFF,

Parameters

Encryption	bit 0 msr: • true: enable MSR with Encryption, • false: disable MSR with Encryption, bit 1 icc: • true: enable ICC with Encryption, • false: disable ICC with Encryption,

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.11 config_setEncryptionControl() [2/2]

```
int com.idtechproducts.device.IDT_Augusta.config_setEncryptionControl (
            boolean msr,
            boolean icc )
```

Set Encryption Control

Set Encryption Control to switch status between MSR and ICC/EMV function. Following Encryption status supported:

- MSR ON, ICC/EMV ON,
- MSR ON, ICC/EMV OFF,
- MSR OFF, ICC/EMV OFF,

Parameters

msr	• true: enable MSR with Encryption, • false: disable MSR with Encryption,
icc	• true: enable ICC with Encryption, • false: disable ICC with Encryption,

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.12 config_setLEDController()

```
int com.idtechproducts.device.IDT_Augusta.config_setLEDController (
            boolean firmwareControlMSRLED,
            boolean firmwareControlICCLED )
```

Set the LED Controller Set the MSR / ICC LED controlled by software or firmware NOTE: The ICC LED always controlled by software.

Parameters

firmwareControlMSRLED	• true: firmware control the MSR LED • false: software control the MSR LED
firmwareControlICCLED	• true: firmware control the ICC LED • false: software control the ICC LED

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.13 config_setXMLFileNameWithPath()

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

set XML Configuration File Name with the full path.

Parameters

xmlFilename,XML	Configuration File Name.

Returns

none

12.2.2.14 ctls_cancelTransaction()

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

Cancel CTLS Transaction

Cancels the currently executing CTLS transaction.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.15 ctls_startTransaction()

```
int com.idtechproducts.device.IDT_Augusta.ctls_startTransaction ( )
```

Enable CTLS interface. Returns encrypted data by call back function.
The function swipeMSRData in interface OnReceiverListener will be called if contactless data received.

See also

OnReceiverListener

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.16 device_connectWithProfile()

```
boolean com.idtechproducts.device.IDT_Augusta.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.

12.2.2.17 device_controlBeep()

```
int com.idtechproducts.device.IDT_Augusta.device_controlBeep (
            int index,
            int frequency,
            int duration )
```

Retrieves Audio Jack setting.

Parameters

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

Returns

success or error code.

See also

ErrorCode Control beep of the device.

Parameters

index	the value can only be 1.
frequency	the frequency of the beep.
duration	the duration of the beep.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.18 device_controlLED()

```
int com.idtechproducts.device.IDT_Augusta.device_controlLED (
            byte indexLED,
            byte control,
            int intervalOn,
            int intervalOff )
```

Control LED of the device.

Parameters

indexLED	the value can only be 1.
control	the led options.
intervalOn	the led on interval between 200ms and 2000ms.
intervalOff	the led off interval between 200ms and 2000ms.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.19 device_controlLED_ICC()

```
int com.idtechproducts.device.IDT_Augusta.device_controlLED_ICC (
```

```
            int controlMode,
            int interval )
```

Control LED of the device.

Parameters

controlMode	0: Off, 1: Solid, 2: Blink.
interval	Only valid while controlMode is 2. The value can be 500, 1000, or 1500ms.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.20 device_getDeviceType()

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

Gets type of device

12.2.2.21 device_getDRS()

```
int com.idtechproducts.device.IDT_Augusta.device_getDRS (
            ResDataStruct respData )
```

Get DRS

Parameters

respData.resData	Response Body is <DRS sourceblk="" number>=""> <SourceBlk1> . . . [<SourceBlkN>] Where: DRS –Destructive Reset <DRS sourceblk="" number>=""> is 2 bytes, format is NumL NumH. It is Number of <SourceBlkX> <SourceBlkX> is n bytes, Format is <SourceID> <SourceLen> <SourceData> <SourceID> is 1 byte <SourceLen> is 1 byte, it is length of <SourceData> **Item | SourceID | SourceLen | SourceData**

Master Chip Check Value Error | 00 | 1 | 01 – Application Error

Slave Chip Check Value Error | 01 | 1 | 01 – Application Error

Korea Self-Test Error | 02 | 1 | 0x01 – EMV L2 Configuration Check Value Error

| | | 0x02 – Future Key Check Value Error

Battery | 10 | 1 | 01 – Battery Error

Tamper Switch | 11 | 1 | Bit 0 – Tamper Switch 1 (0-No, 1-Error) | | | Bit 1 – Tamper Switch 2 (0-No, 1-Error) | | | Bit 2 – Tamper Switch 3 (0-No, 1-Error) | | | Bit 3 – Tamper Switch 4 (0-No, 1-Error) | | | Bit 4 – Tamper Switch 5 (0-No, 1-Error)

| | | Bit 5 – Tamper Switch 6 (0-No, 1-Error)

Temperature | 12 | 1 | 01 –TemperatureHigh or Low

Voltage | 13 | 1 | 01 –Voltage High or Low

Other | 1F | 4 | Reg31∼24bits, Reg23∼16bits, | | | Reg15∼8bits, Reg7∼0bits

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.22 device_getFirmwareVersion()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.23 device_getKeyStatus()

```
int com.idtechproducts.device.IDT_Augusta.device_getKeyStatus (
            ResDataStruct respData )
```

get key status of the device

Parameters

respData.resData	For Augusta: Response Body is PIN DUKPT Status + PIN Master Key Status + PIN Session Key Status + Data encryption KeyStatus + Data encryption KeyStatus + RKI-KEK **Where:**

Key | **Status** | **Note**

PIN DUKPT Key | 0: None. | | 1: Exist | Does not support this key. Always 0

| **0xFF: STOP** |

PIN Master Key | 0: None |

| **1: At least Exist a Master Key** | **Does not support this key. Always 0**

PIN Session Key | 0: None. |

| **1: Exist** | **Does not support this key. Always 0**

Data encryption Key/MSRDUKPT Key | 0: None. | | 1: Exist |

| **0xFF: STOP** |

Data encryption Key/ICC DUKPT Key | 0: None. | | 1: Exist |

| **0xFF: STOP** |

RKI-KEK | 0: None. | | 1: Exist |

| **0xFF: STOP** |

For Augusta S TTK <Block length=""> <KeyStatusBlock1> <[KeyStatusBlock2]> ... <[KeyStatusBlockN]> Where: <Block length=""> is 2 bytes, format is Len_L Len_H, is KeyStatusBlock Number <KeyStatusBlock↩ X> is 4 bytes, format is <Key index="" and="" key="" name=""> <key slot=""> <key status="">: <Key index="" and="" key="" name=""> is 1 byte. Please refer to following table and <80000426-001 KeyNameIndex Database – V51.xls> <key slot=""> is 2 bytes. Range is 0 – 9999 <key status=""> is 1 byte. 0 – Not Exist 1 – Exist 0xFF – (Stop. Only Valid for DUKPT Key)

Support <Key index="" and="" key="" name=""> Table

KeyNameIndex | **Key Name** | **Definition** | **Key Slot**

0x14 | LCL-KEK | Encrypt Other Keys | 0 0x02 | Data encryption Key | Encrypt ICC | 0 0x05 | MAC DUKPT Key | Host-Device – MAC Verification | 0 0x05 | MTK DUKPT Key | TTK Self-Test | 16 0x0C | RKI-KEK | Remote Key Injection | 0

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.24 device_getResponseCodeString()

```
String com.idtechproducts.device.IDT_Augusta.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

12.2.2.25 device_isConnected()

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

get the status if the device connected.

Returns

true: connected, false: disconnected

12.2.2.26 device_isSRED()

```
boolean com.idtechproducts.device.IDT_Augusta.device_isSRED ( )
```

Check if the device is SRED

Returns

true if it is SRED, otherwise false.

12.2.2.27 device_isThales()

```
boolean com.idtechproducts.device.IDT_Augusta.device_isThales ( )
```

Check if the device is Thales

Returns

true if it is Thales, otherwise false.

12.2.2.28 device_isTTK()

```
boolean com.idtechproducts.device.IDT_Augusta.device_isTTK ( )
```

Check if the device is TTK

Returns

true if it is TTK, otherwise false.

12.2.2.29 device_rebootDevice()

```
int com.idtechproducts.device.IDT_Augusta.device_rebootDevice ( )
```

Reboot device.The device will restart and need to reconnect device if success.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.30 device_selfCheck()

```
int com.idtechproducts.device.IDT_Augusta.device_selfCheck ( )
```

Self check for TTK If Self-Check function Failed, then work into De-activation State. If device work into De-activation State, All Sensitive Data will be erased and it need be fixed in Manufacture.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.31 device_sendDataCommand() [1/2]

```
int com.idtechproducts.device.IDT_Augusta.device_sendDataCommand (
            String cmd,
            boolean calcLRC,
            String data,
```

```
            ResDataStruct respData,
            int timeout )
```

Send a direct command to device

Sends a command represented by the provide string to the device.

Parameters

cmd	NSData representation of command to execute
calcLRC	If TRUE, this will wrap command with start/length/lrc/sum/end: '{STX}{Len_Low}{Len_High} data {CheckLRC} {CheckSUM} {ETX}'
data	Ignored. Not applicable for use with Augusta's NGA protocol
response	Returns response ResDataStruct.respData
timeout	Command timeout in seconds

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.32 device_sendDataCommand() [2/2]

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

Send a direct command to device

Sends a command represented by the provide string to the device.

Parameters

cmd	NSData representation of command to execute
calcLRC	If TRUE, this will wrap command with start/length/lrc/sum/end: '{STX}{Len_Low}{Len_High} data {CheckLRC} {CheckSUM} {ETX}'
data	Ignored. Not applicable for use with Augusta's NGA protocol
response	Returns response ResDataStruct.respData

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.33 device_setDateTime()

```
int com.idtechproducts.device.IDT_Augusta.device_setDateTime (
            byte [] mac )
```

set date and time of the device

Parameters

mac	<MAC data>=""> is: For Non-PCI device, Not Exist For PCI device, it is Fix30 bytes data: <MAC value="" length>=""> is 2 byte – Fix is 0x10 0x00 <MAC value>=""> is 16 bytes –Please refer to "Verification Algorithm" section.MAC value is MAC-HOST.The msgX is"78 53 01 50 <FunLen> <Date/Time Length> <Date/Time> <MAC length>=""> <MAC value="" length>="">" <MAC key="" ksn="" length>=""> is 2 byte – Fix is 0x0A 0x00 <MAC key="" ksn>=""> is 10 bytes – MAC DUKPT Key KSN

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.34 device_setDeviceType() [1/2]

```
boolean com.idtechproducts.device.IDT_Augusta.device_setDeviceType (
            ReaderInfo.DEVICE_TYPE deviceType,
            boolean isTTK,
            boolean isSRED,
            boolean isThales )
```

Defines connection USB

Parameters

deviceType	DEVICE_TYPE.DEVICE_AUGUSTA
isTTK	True if TTK device
isSRED	True if SRED device
isThales	True if Thales device

12.2.2.35 device_setDeviceType() [2/2]

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

Defines connection USB

Parameters

deviceType	DEVICE_TYPE.DEVICE_AUGUSTA

12.2.2.36 device_startRKI()

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

Start remote key injection.

Returns

success or error code.

See also

ErrorCode

12.2.2.37 device_startTransaction() [1/2]

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

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

12.2.2.38 device_startTransaction() [2/2]

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.39 device_verifyBackdoorKey()

```
int com.idtechproducts.device.IDT_Augusta.device_verifyBackdoorKey ( )
```

Verify Backdoor Key to Unlock Security

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.40 emv_allowFallback()

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

Allow fallback for EMV transactions. Default is TRUE

Parameters

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

12.2.2.41 emv_authenticateTransaction()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.42 emv_cancelTransaction()

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

Cancel EMV Transaction

Cancels the currently executing EMV transaction.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.43 emv_completeTransaction()

```
int com.idtechproducts.device.IDT_Augusta.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 TVL data to return with transaction results (if any)

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.44 emv_getAutoAuthenticateTransaction()

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

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

12.2.2.45 emv_getAutoCompleteTransaction()

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

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

12.2.2.46 emv_getEMVConfigurationCheckValue()

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

Get EMV Kernel configuration check value info

Parameters

response	Response returned of Kernel configuration check value info

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.47 emv_getEMVKernelCheckValue()

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

Get EMV Kernel check value info

Parameters

response	Response returned of Kernel check value info

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.48 emv_getEMVKernelVersion()

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

Polls device for EMV Kernel Version

Parameters

response	Response returned of Kernel Version

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.49 emv_lcdControlResponse()

```
void com.idtechproducts.device.IDT_Augusta.emv_lcdControlResponse (
            byte mode,
            byte data )
```

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()

12.2.2.50 emv_removeAllApplicationData()

```
int com.idtechproducts.device.IDT_Augusta.emv_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

12.2.2.51 emv_removeAllCAPK()

```
int com.idtechproducts.device.IDT_Augusta.emv_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

12.2.2.52 emv_removeAllCRL()

```
int com.idtechproducts.device.IDT_Augusta.emv_removeAllCRL ( )
```

Remove All Certificate Revocation List Entries

Removes all CRLEntry entries

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.53 emv_removeApplicationData()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.54 emv_removeCAPK()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.55 emv_removeCRL()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.56 emv_removeTerminalData()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.57 emv_retrieveAidList()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.58 emv_retrieveApplicationData()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.59 emv_retrieveCAPK()

```
int com.idtechproducts.device.IDT_Augusta.emv_retrieveCAPK (
            byte [] data,
            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
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

12.2.2.60 emv_retrieveCAPKList()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.61 emv_retrieveCRL()

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

Retrieve the Certificate Revocation List

Returns the CRL entries on the terminal.

Parameters

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

12.2.2.62 emv_retrieveTerminalData()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.63 emv_retrieveTransactionResult()

```
int com.idtechproducts.device.IDT_Augusta.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
tlv	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

12.2.2.64 emv_setApplicationData()

```
int com.idtechproducts.device.IDT_Augusta.emv_setApplicationData (
            String aid,
            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

12.2.2.65 emv_setAutoAuthenticateTransaction()

```
static void com.idtechproducts.device.IDT_Augusta.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

12.2.2.66 emv_setAutoCompleteTransaction()

```
static void com.idtechproducts.device.IDT_Augusta.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

12.2.2.67 emv_setCAPK()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.68 emv_setCRL()

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

Set Certificate Revocation List

Sets the CRL

Parameters

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

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.69 emv_setTerminalData()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.70 emv_startTransaction()

```
int com.idtechproducts.device.IDT_Augusta.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)
amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.

Parameters

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.
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

12.2.2.71 getSDKInstance()

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

Returns an instance of the currently initialized IDT_Device class.

Returns

IDT_Device instance

12.2.2.72 icc_disable()

```
int com.idtechproducts.device.IDT_Augusta.icc_disable ( )
```

ICC Function disable Disable ICC function

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.73 icc_enable()

```
int com.idtechproducts.device.IDT_Augusta.icc_enable (
            boolean withNotification )
```

ICC Function enable Enable ICC function with or without seated notification

Parameters

withNotification	• true: with notification when ICC seated status changed, • false: without notification.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.74 icc_exchangeAPDU()

```
int com.idtechproducts.device.IDT_Augusta.icc_exchangeAPDU (
            byte [] dataAPDU,
            APDUResponseStruct response )
```

Exchange APDU with plain text For Non-SRED Augusta Only

Sends an APDU packet to the ICC. If successful, response is the APDU data in response parameter.

Parameters

dataAPDU	APDU data packet
response	Unencrypted APDU response

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.75 icc_getAPDU_KSN()

```
int com.idtechproducts.device.IDT_Augusta.icc_getAPDU_KSN (
            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 the current KSN for Smart card. ∗ The KSN in the Response should be the KSN in the loop exchanging until ICC is powered off.

Parameters

KeyNameIndex	1 byte value. See the table below.
KeySlot	1 or 2 bytes value. See the table below.

KeyNameIndex | Key Slot (1 byte or 2 bytes) | Key Name (Usage)

0x14 | 0 | LCL-KEK 0x02 | 0 | Data encryption Key 0x0C | 0 | RKI-KEK 0x05 | 0 | MAC DUKPT Key (Verify commands)

0x05 | 16 | MTK DUKPT Key (TTK Self-Test)

Parameters

resKSN	the class for current KSN.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.76 icc_getFunctionStatus()

```
int com.idtechproducts.device.IDT_Augusta.icc_getFunctionStatus (
            ResDataStruct respData )
```

Get ICC Function status Get ICC Function status about enable/disable and with or without seated notification

Parameters

respData	Response Body is 72 01 11 01 <ICC reading="" characteristics>=""> <ICC reading="" characteristics>=""> is also stored in the variable functionStatus functionStatus: 0x30: ICC Function Off 0x31: ICC Function Enable & Notification Off 0x32: ICC Function Enable & Notification On

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.77 icc_getICCReaderStatus()

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

Get Reader Status

Returns the reader status

Parameters

status	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

12.2.2.78 icc_getKeyFormatForICCDUKPT()

```
int com.idtechproducts.device.IDT_Augusta.icc_getKeyFormatForICCDUKPT (
            ResDataStruct respData )
```

Get key format for ICC DUKPT.

Specifies how data will be encrypted with Data Key or PIN key (if DUKPT key loaded)

Parameters

respData	Response Body is 78 01 03 01 <Option> The Option is stored in the variable encryptionOption encryptionOption: • 'TDES': Encrypted card data with TDES if DUKPT Key had been loaded.(default) • 'AES': Encrypted card data with AES if DUKPT Key had been loaded. • 'NONE': No Encryption.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.79 icc_getKeyTypeForICCDUKPT()

```
int com.idtechproducts.device.IDT_Augusta.icc_getKeyTypeForICCDUKPT (
            ResDataStruct respData )
```

Get key type for ICC DUKPT.

Specifies the key type used for ICC DUKPT encryption

Parameters

respData	Response Body is 78 01 02 01 <Option> <Option> is also stored in keyType. keyType: • 'DATA': Encrypted card data with Data Key DUKPT Key had been loaded.(default) • 'PIN': Encrypted card data with PIN Key if DUKPT Key had been loaded.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.80 icc_passthroughOffICC()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.81 icc_passthroughOnICC()

```
int com.idtechproducts.device.IDT_Augusta.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.

See also

ErrorCode

12.2.2.82 icc_powerOffICC()

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

Power Off ICC

Powers down the ICC

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

12.2.2.83 icc_powerOnICC()

```
int com.idtechproducts.device.IDT_Augusta.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

options	the options is optional. please see PowerOnStructure class for more information.

See also

PowerOnStructure (not used for the UniPay II)

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

12.2.2.84 icc_reviewAllSetting()

```
int com.idtechproducts.device.IDT_Augusta.icc_reviewAllSetting (
            ICCSettingStruct iccSetting )
```

Review all setting of ICC status.

Parameters

iccSetting	for ICC setting. please see class ICCSettingStruct for more information. mainCardTypeOption: 0x00 ISO, 0xFF EMV timeout: default value 8 readingCharacteristics: 0x30 ICC Function Off, 0x31 ICC Function Enable & Notification Off, 0x32 ICC Function Enable & Notification On prePANIDLen: Default is 4 postPANIDLen: Default is 4 maskCharWithASCII: Default is 0x2A maskCharWithBCD: Default is 0x0C CTL2Interval: Default is 0x0C

See also

ICCSettingStruct

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.85 icc_setKeyFormatForICCDUKPT()

```
int com.idtechproducts.device.IDT_Augusta.icc_setKeyFormatForICCDUKPT (
            byte encryption )
```

Set Key Format for ICC DUKPT

Sets how data will be encrypted, with either TDES or AES (if DUKPT key loaded)

Parameters

encryption	Encryption Type • 00: Encrypt with TDES • 01: Encrypt with AES

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.86 icc_setKeyTypeForICCDUKPT()

```
int com.idtechproducts.device.IDT_Augusta.icc_setKeyTypeForICCDUKPT (
            byte encryption )
```

Set Key Type for ICC DUKPT Key

Sets which key the data will be encrypted with, with either Data Key or PIN key (if DUKPT key loaded)

Parameters

encryption	Encryption Type • 00: Encrypt with Data Key • 01: Encrypt with PIN Key

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.87 log_deleteLogs()

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

delete the log in the root path of SD card.

Returns

number of log files deleted

See also

log_setSaveLogEnable

12.2.2.88 log_setSaveLogEnable()

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

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

Parameters

enableShowLog,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

12.2.2.89 log_setVerboseLoggingEnable()

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

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

Parameters

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

Returns

none

12.2.2.90 msr_cancelMSRSwipe()

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.91 msr_defaultAllSetting()

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

Default all setting of Mask and Encryption.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.92 msr_disable()

```
int com.idtechproducts.device.IDT_Augusta.msr_disable ( )
```

Disable MSR.

Returns

success or error code.

See also

ErrorCode

12.2.2.93 msr_enableBufferMode()

```
int com.idtechproducts.device.IDT_Augusta.msr_enableBufferMode (
            boolean isBufferMode,
            boolean withNotification )
```

Get MSR enable buffer mode.

Parameters

isBufferMode	set to true if in buffer mode, false if in auto mode
withNotification	set to true with notification, false otherwise

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.94 msr_getClearPANID()

```
int com.idtechproducts.device.IDT_Augusta.msr_getClearPANID (
            ResDataStruct respData )
```

Get Clear PAN ID.

Returns the number of digits that begin the PAN that will be in the clear

Parameters

respData	4901 <Setting value>="">. The Setting Value is stored in the variable settingValue settingValue: Number of digits in clear. Values are char '0' - '6'

Returns

success or error code. Returns the number of digits that begin the PAN that will be in the clear

See also

ErrorCode

12.2.2.95 msr_getExpirationMask()

```
int com.idtechproducts.device.IDT_Augusta.msr_getExpirationMask (
            ResDataStruct respData )
```

Get MSR expiration date mask.

Parameters

respData	5001 <Setting value>="">. The Setting Value is stored in the variable settingValue settingValue: '0' = masked, '1' = not-masked

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.96 msr_getFunctionStatus()

```
int com.idtechproducts.device.IDT_Augusta.msr_getFunctionStatus (
            ResDataStruct respData )
```

Get MSR function status.

Get MSR Function status about enable/disable and with or without seated notification

Parameters

respData	1A01 <Setting value>="">. The Setting Value is stored in the variable functionStatus functionStatus: 0x31: MSR Function enabled = true, in the buffer mode = false, without notification when swiped MSR Card 0x32: MSR Function enabled = true, in the buffer mode = true, without notification when swiped MSR Card 0x33: MSR Function enabled = true, in the buffer mode = true, with notification when swiped MSR Card

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.97 msr_getSetting()

```
int com.idtechproducts.device.IDT_Augusta.msr_getSetting (
            byte setting,
            ResDataStruct respData )
```

Get Single MSR Setting value

Returns the encryption used for swipe data

Parameters

setting	the msr setting to retrieve.
respData	setting 01 <Setting value>="">. The Setting Value is stored in the variable settingValue settingValue: MSR Setting value

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.98 msr_getSingleSetting()

```
int com.idtechproducts.device.IDT_Augusta.msr_getSingleSetting (
            byte funcID,
            byte [] response )
```

Get single setting of Mask and Encryption by Function ID.

Parameters

funcID	function ID. 0x49:Leading PAN digits to display(0x00∼0x06). 0x4A:Last PAN digits to display(0x00∼0x04). 0x4B:Mask ASCII code track data(0x20∼0x7E). 0x4C:Encryption type ('1'-'2'). '1' 3DES, '2' AES. 0x50:Mask or display expiration date(0x30 or 0x31);0x31:don't mask expiration date. 0x7E:Security Level ID. 0x84:Encryption Option (Forced encryption or not) Bit 0 : T1 force encrypt Bit 1 : T2 force encrypt Bit 2 : T3 force encrypt Bit 3 : T3 force encrypt when card type is 0

0x86:Masked / clear data sending option Bit 0 : T1 mask allowed
Bit 1 : T2 mask allowed
Bit 2 : T3 mask allowed

NOTE:
UniPay support 0x49,0x50,0x4C,0x7E,0x84 and 0x86.
UniPay II support 0x49,0x50,0x4A, 0x4B, 0x4C,0x7E,0x84 and 0x86.

Parameters

response	response[0] for setting data.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.99 msr_getSwipeEncryption()

```
int com.idtechproducts.device.IDT_Augusta.msr_getSwipeEncryption (
            ResDataStruct respData )
```

Get Swipe Data Encryption For Non-SRED Augusta Only Returns the encryption used for swipe data

Parameters

respData	4C01 <Setting value>="">. The Setting Value is stored in the variable settingValue settingValue 1 = TDES, 2 = AES, 0 = NONE

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.100 msr_getSwipeForcedEncryptionOption()

```
int com.idtechproducts.device.IDT_Augusta.msr_getSwipeForcedEncryptionOption (
            ResDataStruct respData )
```

Get MSR Swipe Forced Encryption Option.

Parameters

respData	8401 <Setting value>="">. The Setting Value is stored in the variable encryptionOption option Byte using lower four bits as flags. 0 = Force Encryption Off, 1 = Force Encryption On bit0 = Track 1 bit1 = Track 2 bit2 = Track 3 bit4 = Track 3 Card Option 0

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.101 msr_getSwipeMaskOption()

```
int com.idtechproducts.device.IDT_Augusta.msr_getSwipeMaskOption (
            ResDataStruct respData )
```

Get MSR Swipe Mask Option.

Gets the swipe mask/clear data sending option

Parameters

respData	8601 <Setting value>="">. The Setting Value is stored in the variable settingValue settingValue Byte using lower three bits as flags. 0 = Mask Option Off, 1 = Mask Option On bit0 = Track 1 bit1 = Track 2 bit2 = Track 3 Example: Response 0x03 = Track1/Track2 Masked Option ON, Track3 Masked Option Off

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.102 msr_RetrieveWhiteList()

```
int com.idtechproducts.device.IDT_Augusta.msr_RetrieveWhiteList (
            ResDataStruct respData )
```

Get MSR white list.

For Non-SRED Augusta Only

Parameters

respData	response data from reader. The return data is stored in respData.resData. respData.resData: the white list data which is ASN.1 Block format

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.103 msr_reviewAllSetting()

```
int com.idtechproducts.device.IDT_Augusta.msr_reviewAllSetting (
            MSRSettingStruct msrSetting )
```

Review all setting of Mask and Encryption.

Parameters

msrSetting	for MSR setting. please see class MSRSettingStruct for more information

See also

MSRSettingStruct

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.104 msr_setClearPANID()

```
int com.idtechproducts.device.IDT_Augusta.msr_setClearPANID (
            byte value )
```

Set Clear PAN ID.

Parameters

value	Set Clear PAN ID to value: Number of digits to show in clear. Range 0-6.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.105 msr_setExpirationMask()

```
int com.idtechproducts.device.IDT_Augusta.msr_setExpirationMask (
            boolean mask )
```

Set Expiration Masking

Sets the flag to mask the expiration date

Parameters

mask	TRUE = mask expiration

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.106 msr_setSetting()

```
int com.idtechproducts.device.IDT_Augusta.msr_setSetting (
            byte setting,
            byte val )
```

Set MSR settings.

Parameters

setting	the msr setting to set.
val	the value to set to the msr setting.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.107 msr_setSingleSetting()

```
int com.idtechproducts.device.IDT_Augusta.msr_setSingleSetting (
            byte funcID,
            byte setData )
```

Set single setting of Mask and Encryption by Function ID.

Parameters

funcID	function ID. 0x49:Leading PAN digits to display(0x00∼0x06). 0x4A:Last PAN digits to display(0x00∼0x04). 0x4B:Mask ASCII code track data(0x20∼0x7E). 0x4C:Encryption type ('1'-'2'). '1' 3DES, '2' AES. 0x50:Mask or display expiration date(0x30 or 0x31);0x31:don't mask expiration date. 0x7E:Security Level ID. 0x84:Encryption Option (Forced encryption or not) Bit 0 : T1 force encrypt Bit 1 : T2 force encrypt Bit 2 : T3 force encrypt Bit 3 : T3 force encrypt when card type is 0

0x86:Masked / clear data sending option Bit 0 : T1 mask allowed
Bit 1 : T2 mask allowed
Bit 2 : T3 mask allowed

NOTE:
UniPay support 0x49,0x50,0x4C,0x7E,0x84 and 0x86.
UniPay II support 0x49,0x50,0x4A, 0x4B, 0x4C,0x7E,0x84 and 0x86.

Parameters

setData	for setting data.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.108 msr_setSwipeEncryption()

```
int com.idtechproducts.device.IDT_Augusta.msr_setSwipeEncryption (
            byte encryption )
```

Set MSR Swipe Forced Encryption Option.

For Non-SRED Augusta Only

Sets the swipe encryption method

Parameters

encryption	1 = TDES, 2 = AES Set swipe encryption to encryption value.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.109 msr_setSwipeForcedEncryptionOption()

```
int com.idtechproducts.device.IDT_Augusta.msr_setSwipeForcedEncryptionOption (
            boolean track1,
            boolean track2,
            boolean track3,
            boolean track3card0 )
```

Set MSR Swipe Forced Encryption Option.

Parameters

tarck1	Set track1 encryption to true or false.
tarck2	Set track2 encryption to true or false.
tarck3	Set track3 encryption to true or false.
tarck3card0	Set track3 card0 encryption to true or false.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.110 msr_setSwipeMaskOption()

```
int com.idtechproducts.device.IDT_Augusta.msr_setSwipeMaskOption (
            boolean track1,
```

```
            boolean track2,
            boolean track3 )
```

Set MSR Swipe Mask Option.

Sets the swipe mask/clear data sending option

Parameters

tarck1	Set track1 mask to true or false.
tarck2	Set track2 mask to true or false.
tarck3	Set track3 mask to true or false.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.111 msr_setWhiteList()

```
int com.idtechproducts.device.IDT_Augusta.msr_setWhiteList (
            byte [] val )
```

Set MSR white list.

For Non-SRED Augusta Only

Parameters

val	the white list data which is ASN.1 Block format.

Returns

success or error code. Values can be parsed with device_getResponseCodeString

See also

ErrorCode

12.2.2.112 msr_startMSRSwipe() [1/2]

```
int com.idtechproducts.device.IDT_Augusta.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

12.2.2.113 msr_startMSRSwipe() [2/2]

```
int com.idtechproducts.device.IDT_Augusta.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 30 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

12.2.2.114 phone_getInfoManufacture()

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

Get manufacture version.

Returns

the manufacture info

12.2.2.115 phone_getInfoModel()

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

Get phones's model number information.

Returns

the model number information.

12.2.2.116 registerListen()

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

General API:registerListen.

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

12.2.2.117 release()

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

release, make the SDK in the idle status.

12.2.2.118 setIDT_Device()

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

For System Use Only

Parameters

fwTool	Parameter for firmware update

12.2.2.119 unregisterListen()

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

unregisterListen to disable the detect

12.2.2.120 useUSBIntentFilter()

```
static void com.idtechproducts.device.IDT_Augusta.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_Augusta.java

12.3 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 **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

12.3.1 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

12.4 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

12.4.1 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.

12.4.2 Member Data Documentation

12.4.2.1 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;

12.4.2.2 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;

12.4.2.3 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.

12.4.2.4 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

12.4.2.5 ctlsApplication

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

CTLS Application

12.4.2.6 DE055Data

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

Get the swiped card of DE055 data.

12.4.2.7 DE055Len

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

Get the swiped card length of DE055 data.

12.4.2.8 encryptedTags

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

Encrypted card data provided via TLV.

12.4.2.9 encTrack1

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

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

12.4.2.10 encTrack2

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

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

12.4.2.11 encTrack3

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

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

12.4.2.12 event

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

MSR type,please see EVENT_MSR_Types for more information.

12.4.2.13 fastEMV

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

Fast EMV String.

12.4.2.14 hasDE055

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

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

12.4.2.15 iccPresent

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

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

12.4.2.16 isCTLS

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

Track data was captured via CTLS interface

12.4.2.17 KSN

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

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

12.4.2.18 maskedTags

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

Masked card data provided via TLV.

12.4.2.19 optionalBytes

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

Get optional bytes of the swiped card data.

12.4.2.20 rawTrackData

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

Get the DFEE23 MSR raw data.

12.4.2.21 result

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

Return error code.

12.4.2.22 serialNumber

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

Get the Reader Serial Number.

12.4.2.23 TLVData

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

Get the swiped card TLV data.

12.4.2.24 TLVLen

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

Get the swiped card length of TLV data.

12.4.2.25 track1

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

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

12.4.2.26 track1Length

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

Get the swiped card length of Track1 data.

12.4.2.27 track2

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

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

12.4.2.28 track2Length

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

Get the swiped card length of Track2 data.

12.4.2.29 track3

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

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

12.4.2.30 track3Length

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

Get the swiped card length of Track3 data.

12.4.2.31 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

12.5 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)

12.5.1 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.

12.5.2 Member Function Documentation

12.5.2.1 autoConfigCompleted()

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

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

12.5.2.2 autoConfigProgress()

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

The auto config process percent value.

12.5.2.3 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
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

12.5.2.4 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.

12.5.2.5 deviceConnected()

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

Fires when device connects.

12.5.2.6 deviceDisconnected()

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

Fires when device disconnects.

12.5.2.7 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

12.5.2.8 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.

12.5.2.9 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

12.5.2.10 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.

12.5.2.11 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.

12.5.2.12 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.

12.5.2.13 msgBatteryLow()

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

Battery low status notification,

12.5.2.14 msgRKICompleted()

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

RKI succeeded; MAC result as return value.

12.5.2.15 msgToConnectDevice()

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

The message notify the application to connect the device.

12.5.2.16 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.

12.5.2.17 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

Index