Value through Innovation

UniPay SDK Guide

for Android

#80131523-001
Rev. A

Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;UniPay;SDK;Android	7/23/2015

Contents

Chapter 1

IDTech UniPay for Android Reference Guide

The IDTechIDTech UniPay for Android SDK is a Java library (∗.jar) that will be provided by IDTech as the main interface between Android application, the UniPay 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 application wishing to deploy with the payment application.

- EMV Tag Reference
- Enumeration Reference
- UniPay Error Code Reference

Chapter 2

Connecting with UniPay

The UniPay connects through Headphone jack or USB on Android.

2.1 Connect with USB

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

2.2 Connect with Audio

By default, the SDK communicates with the device through the headphone jack. No further steps required. Please note: The USB port takes priority over the audio interface. A common mistake is using a computers USB port as a source of power while attempting to use the audio interface. Doing this will disable the audio interface. If power is desired while using the audio interface, only use a power adapter as the source of USB power, not a computers USB port.

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, IEEE 802.11i) to implement strong encryption for authentication and transmission. Controls must be in place to protect the implemented wireless network from unknown wireless access points and clients. This includes testing the end users wireless deployment on a quarterly basis to detect unauthorized access points within the system. Change wireless vendor defaults, including but not limited to default wireless encryption keys, passwords, and SNMP 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

UniPay Main Transaction Commands

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

4.1 EMV Methods

Start EMV Transaction

com.idtechproducts.emv.UniPayEMV.startEMVTransaction()

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

Complete Online EMV Transaction

com.idtechproducts.emv.UniPayEMV.completeOnlineEMVTransaction()

After receiving a host response, pass the result code as a string ("00"). The tags will be returned in the emv↩ TransactionData delegate protocol.

If there was a communication error with host, you must still finish the EMV transaction by passing "EMV_COMPL↩ ETION_RESULT_UNABLE_TO_GO_ONLINE".

```
typedef enum{
    EMV_COMPLETION_RESULT_ACCEPTED = 0X00,
    EMV_COMPLETION_RESULT_UNABLE_TO_GO_ONLINE = 0X01,
    EMV_COMPLETION_RESULT_TECHNICAL_ISSUE = 0X02,
    EMV_COMPLETION_RESULT_DECLINED = 0X03,
    EMV_COMPLETION_RESULT_ISSUER_REFERAL = 0X04
} EMV_COMPLETION_RESULT;
```

Terminal Configuration

com.idtechproducts.emv.UniPayEMV.retrieveTerminalData()
com.idtechproducts.emv.UniPayEMV.setTerminalData()

Methods for terminal configuration. When setting the terminal data, you populate and pass and UniPay_Terminal↩ Data structure.

AID Management

com.idtechproducts.emv.UniPayEMV.retrieveApplicationData()
com.idtechproducts.emv.UniPayEMV.retrieveAIDList()
com.idtechproducts.emv.UniPayEMV.setApplicationData()
com.idtechproducts.emv.UniPayEMV.removeApplicationData()

Methods for AID management. When setting the AID, you populate and pass UniPay_ApplicationID. When retrieving the AID list, the list of AID Names/length can be retrieved from the populated NSArray

Kernel Version

com.idtechproducts.emv.UniPayEMV.getEMVKernelVersion()

Method to retrieve kernel version.

APDU Communication

com.idtechproducts.device.IDT_Device.icc_exchangeAPDU()

Allows the direct sending of APDU packets to ICC

4.2 MSR

Request Swipe

com.idtechproducts.device.IDT_Device.msr_startMSRSwipe()

Enables MSR to receive Swipe. Results are returned as IDTMSRData in com.idtechproducts.device.OnReceiver↩ Listener.swipeMSRData

Cancel Swipe

com.idtechproducts.device.IDT_Device.msr_cancelMSRSwipe()

Disables the MSR from receiving swipes.

Chapter 5

Core Implementation UniPay: Android

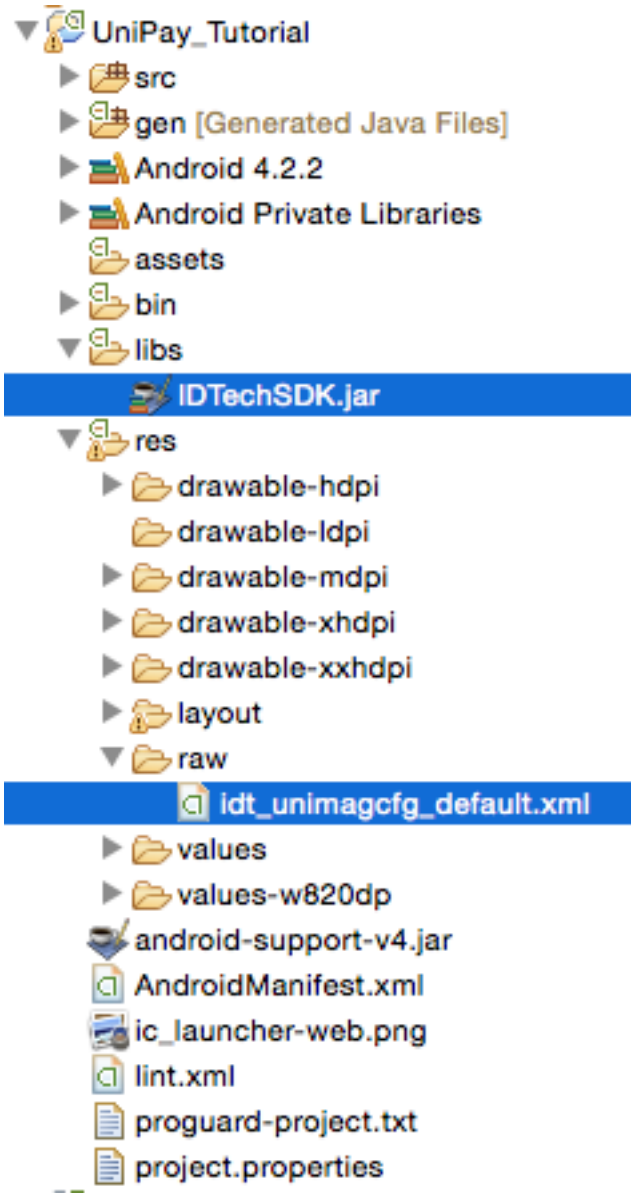
IDTechSDK.jar includes class libraries to interface with the UniPay. This guide assume a fair understanding of Eclipse IDE and general Android programming knowledge.

5.1 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 UniPay objects
- Sample Project Tutorial

5.2 Import the necessary libraries

Communicating with UniPay requires the IDTechSDK.jar file be added to the project. While build paths can be created pointing to the IDTechSDK.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:



5.3 Add Import statements to utilize libraries

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

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

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

5.4 Implement OnReceiverListener for the activity:

In the class that will be a delegate of UniPay, 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 AutoConfigCompleted(StructConfigParameters arg0) {
        // TODO Auto-generated method stub

    }

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

    }

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

    }

    @Override
    public void LoadXMLConfigFileInfo(int arg0, String arg1) {
        // 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 boolean getUserGrant(USER_GRANT_TYPE arg0, String arg1) {
        // TODO Auto-generated method stub
        return false;
    }

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

    }

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

    }

    @Override
    public void plugStatusChange(DEVICE_INTERFACE_Types arg0, boolean arg1) {
        // TODO Auto-generated method stub

    }

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

    }

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

    }
```

```
}
```

5.5 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"/>
```

5.6 Allocate/initialize UniPay 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_Device 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_Device(this,this);
    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);
}
```

5.7 Sample Project Tutorial

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

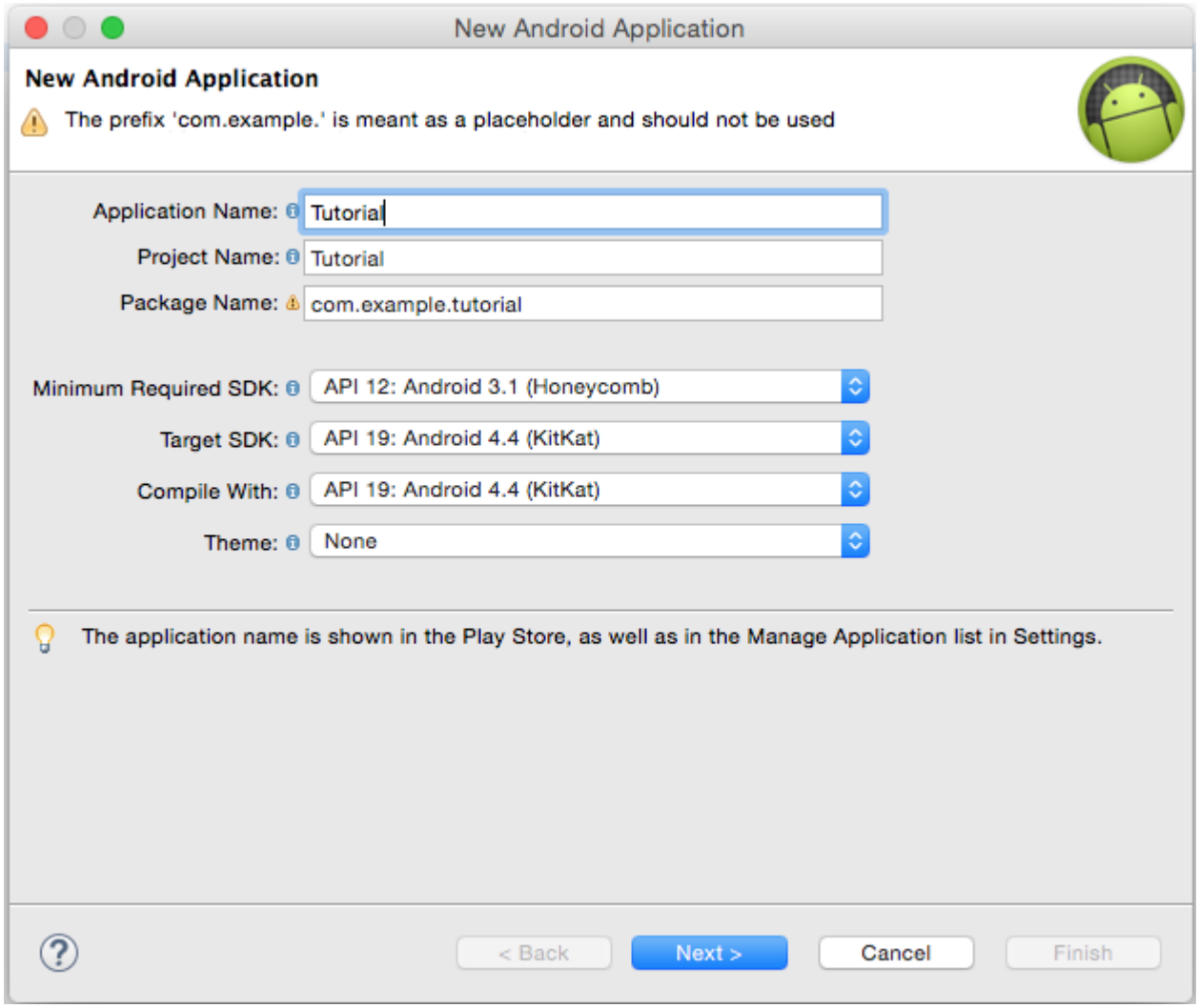
- Report Firmware
- Report connected/disconnected
- Turn on/off MSR reader and perform a card capture

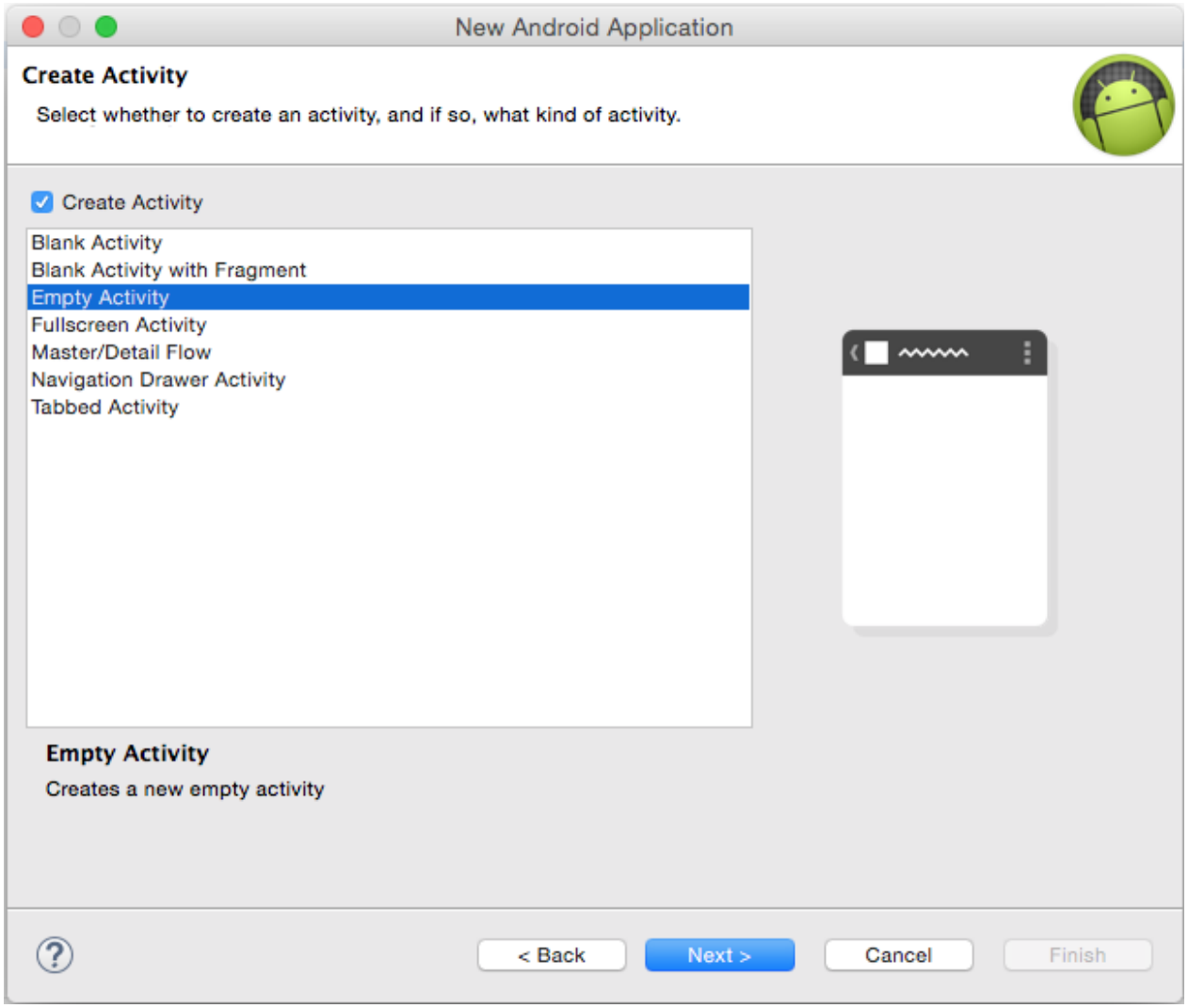
OnReceiverListener methods:

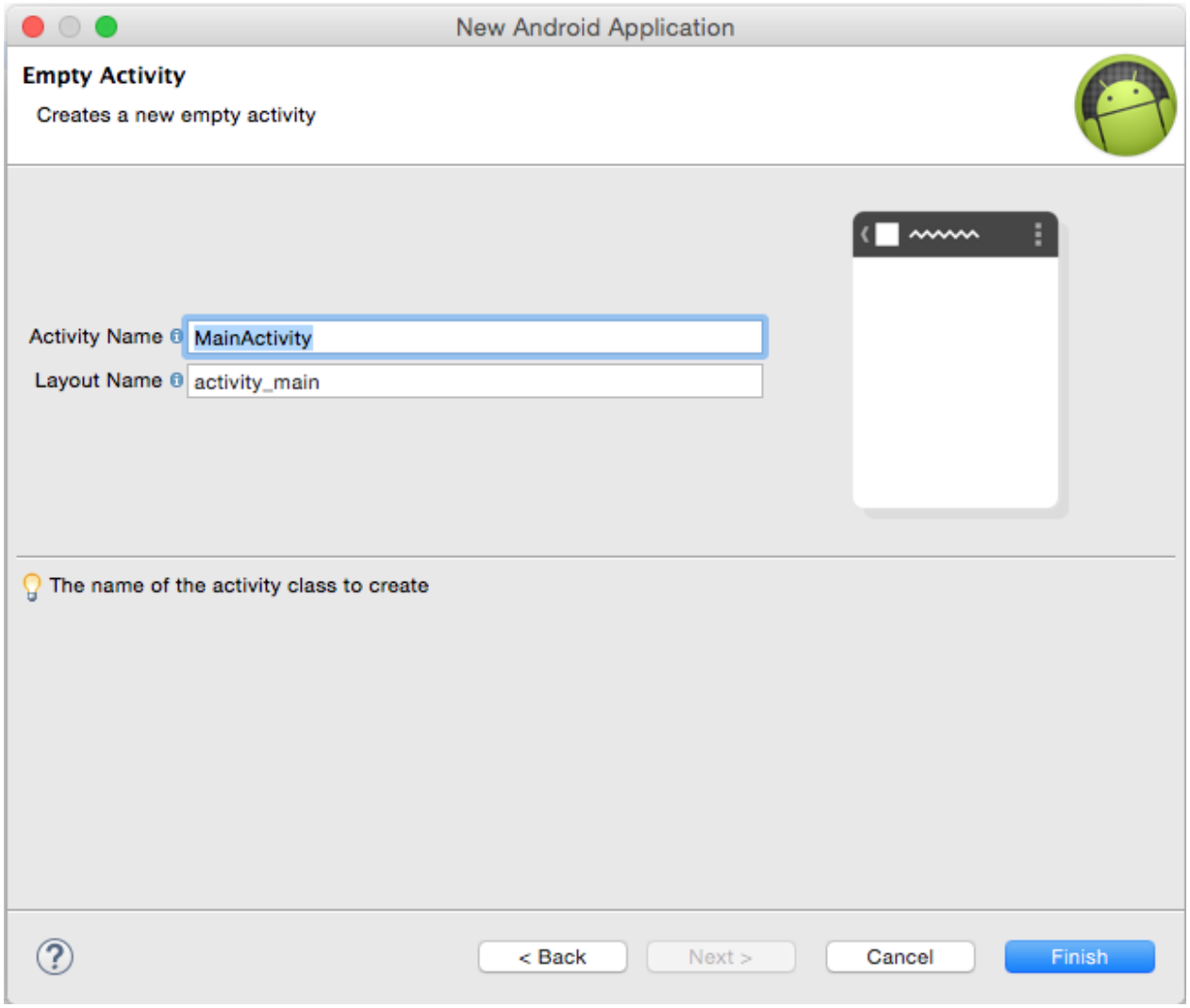
- Method to report unsolicited card swipes
- Method to report device connected
- Method to report device disconnected

5.7.1 Step 1: Create New Project

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


New Android Application
New Android Application
The prefix 'com.example.' is meant as a placeholder and should not be used
Application Name: Tutorial
Project Name: Tutorial
Package Name: com.example.tutorial
Minimum Required SDK: API 12: Android 3.1 (Honeycomb)
Target SDK: API 19: Android 4.4 (KitKat)
Compile With: API 19: Android 4.4 (KitKat)
Theme: None
The application name is shown in the Play Store, as well as in the Manage Application list in Settings.
< Back
Next >
Cancel
Finish


New Android Application
Create Activity
Select whether to create an activity, and if so, what kind of activity.
Create Activity
Blank Activity
Blank Activity with Fragment
Empty Activity
Fullscreen Activity
Master/Detail Flow
Navigation Drawer Activity
Tabbed Activity
Empty Activity
Creates a new empty activity
< Back
Next >
Cancel
Finish



5.7.2 Step 2: Import IDTechSDK for UniPay

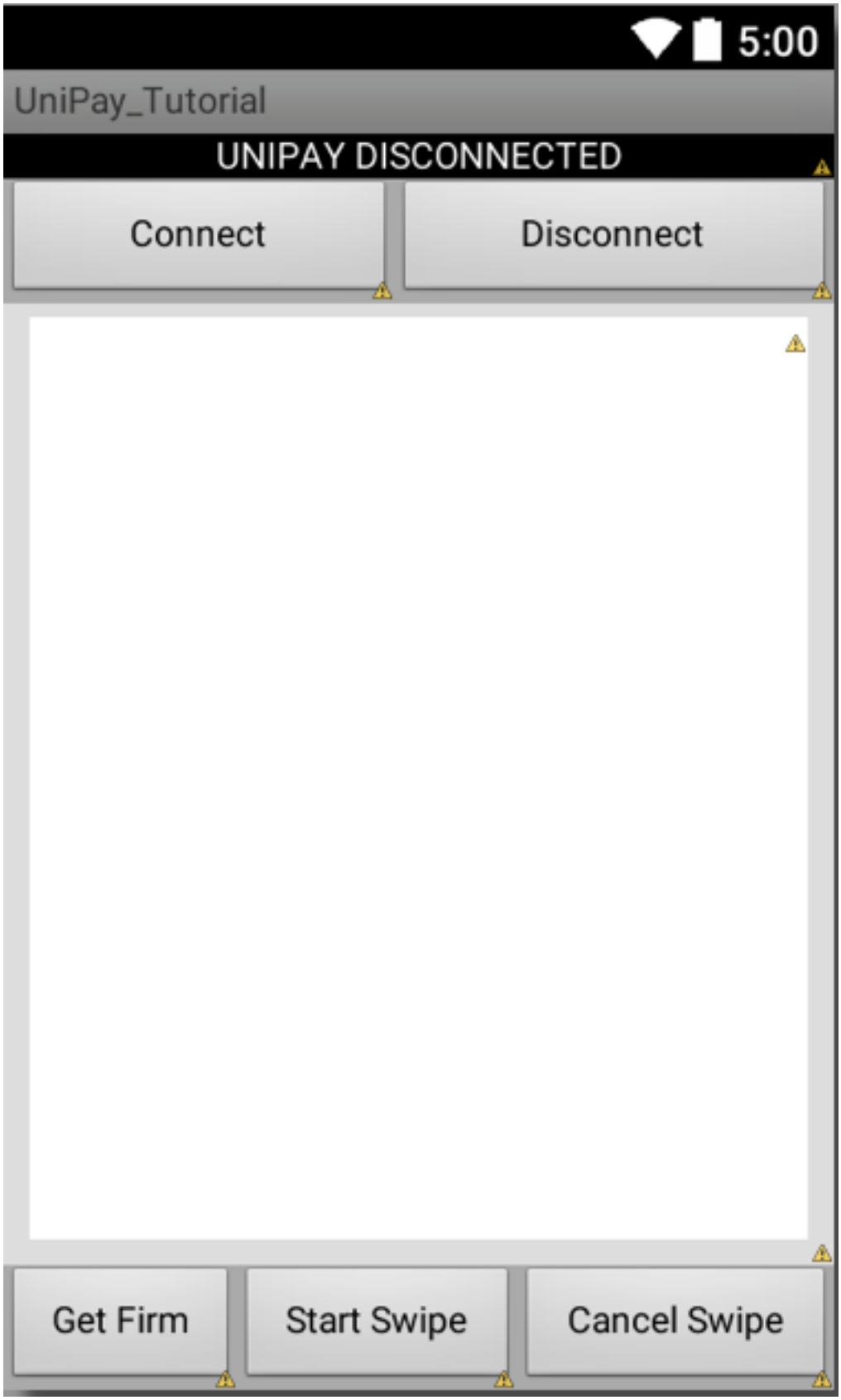
Import the necessary libraries

5.7.3 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 a TextView to communicate data from the UniPay. 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:
  - Connect
  - Disconnect
  - Get Firmware

- Start Swipe
- Cancel Swipe



5.7.4 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 text view, and the 5 buttons

Layout Source Code (uses theme "Light")

```
<?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="UNIPAY DISCONNECTED"
        android:textColor="#FFFFFF" />
    <LinearLayout
        android:id="@+id/linearLayoutBottom2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <Button
            android:id="@+id/btn_Connect"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:gravity="center_vertical|center_horizontal"
            android:text="Connect" />
        <Button
            android:id="@+id/btn_Disconnect"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Disconnect" />
    </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/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 Firm" />
        <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>
```

5.7.5 Step 5: Configure Method File

In the activity file, perform the following:

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

```
// declaring the instance of the UniPayReader;
private IDT_Device myUniPayReader = null;
private TextView connectStatusTextView;
private TextView textLog;
private Button btnGetFirmware;
private Button btnStartSwipe;
private Button btnCancelSwipe;
private Handler handler = new Handler();
private boolean isReaderConnected = false;
private String strCard = "";
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    handler = new Handler();
    btnConnect = (Button)findViewById(R.id.btn_Connect);
    btnDisconnect = (Button)findViewById(R.id.btn_Disconnect);
    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);
    connectStatusTextView = (TextView)findViewById(R.id.status_text);

    if(myUniPayReader!=null){
        myUniPayReader.unregisterListen();
        myUniPayReader.release();
        myUniPayReader = null;
    }
    myUniPayReader = new IDT_Device(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 doUpdateStatus = new Runnable()
{
    public void run()
    {
        if(!isReaderConnected){
            connectStatusTextView.setText("UNIPAY DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("UNIPAY CONNECTED");
        }
    }
```

```
};

@Override
public void deviceConnected() {
    isReaderConnected = true;
    handler.post(doUpdateStatus);
}

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

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

```
private Runnable m_CardRunnable = new Runnable()
{
    public void run()
    {
        textLog.setText(strCard);
    }
};
public void swipeMSRData(IDTMSRData card)
{
    String str = new String();
    if( card.event == EVENT_MSR_Types.EVENT_MSR_CARD_DATA ||card.event == EVENT_MSR_Types.
      EVENT_MSR_DATA_ERROR)
    {
        int len = card.cardData.length;
        int i = 0;
        str = "CardData:\r\n";
        for(i = 0; i < len; i++)
        {
            str += String.format("%02x ", card.cardData[i]);
        }
        if(card.track1 != "")
        {
            str += "\r\nTrack1 data:\r\n";
            str += card.track1;
        }
        if(card.track2 != "")
        {
            str += "\r\nTrack2 data:\r\n";
            str += card.track2;
        }
        if(card.track3 != "")
        {
            str += "\r\nTrack3 data:\r\n";
            str += card.track3;
        }
        len = card.encTrack1.length;
        if(len > 0)
        {
            str += "\r\nTrack1 encrypted data:\r\n";
            for(i = 0; i < len; i++)
            {
                str += String.format("%02X", card.encTrack1[i]);
            }
        }
        len = card.encTrack2.length;
        if(len > 0)
        {
            str += "\r\nTrack2 encrypted data:\r\n";
            for(i = 0; i < len; i++)
            {
                str += String.format("%02X", card.encTrack2[i]);
            }
        }
        len = card.encTrack3.length;
        if(len > 0)
        {
            str += "\r\nTrack3 encrypted data:\r\n";
            for(i = 0; i < len; i++)
            {
                str += String.format("%02X", card.encTrack3[i]);
            }
        }
        len = card.KSN.length;
        if(len > 0)
        {
            str += "\r\nKSN:";
            for(i = 0; i < len; i++)
            {
```

```
                str += String.format("%02x ", card.KSN[i]);
            }
        }
        str += "\r\nCard Type:";
        if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_ISOABA)
            str += String.format("ISO 7813/ISO 4909/ABA format.");
        else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_AAMVA)
            str += String.format("AAMVA format.");
        else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_Raw)
            str += String.format("Raw; un-decoded format.");
        else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_JisI_II)
            str += String.format("JIS I or JIS II.");
        else
            str += String.format("Other format.");
        str += "\r\nEncode Status:";
        str += String.format("%02x ", card.captureEncodeStatus);
        str += "\r\nEncrypt Type:";
        if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_TDES)
            str += String.format("TDES");
        if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_AES)
            str += String.format("AES");
        if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_NONE)
            str += String.format("None");
        if(card.serialNumber != "")
        {
            str += "\r\nReader Serial Number:";
            str += card.serialNumber;
        }


    }
    else if(card.event == EVENT_MSR_Types.EVENT_MSR_BACKSPACE_KEY)
        str = "Press Backspace Key.";
    else if(card.event == EVENT_MSR_Types.EVENT_MSR_CANCEL_KEY)
        str = "Press Cancel Key.";
    else if(card.event == EVENT_MSR_Types.EVENT_MSR_ENTER_KEY)
        str = "Press Enter Key.";
    else
    {
        str = "Swipe Card error, error code is ";
        str += String.format("%02X %02X ", card.cardData[0],card.cardData[1]);
    }
    strCard = str;
    handler.post(m_CardRunnable);
}
```

-Implement the button press methods

```
btnConnect.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        myUniPayReader.device_connect();
    }
});

btnDisconnect.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        myUniPayReader.device_disconnect();
    }
});

btnStartSwipe.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        String str = new String();
        int res = myUniPayReader.msr_startMSRSwipe();
        if(res == 0)
        {
            str = "Start to MSR Swipe successfully,please swipe card.";
        }
        else
            str = myUniPayReader.device_getResponseCodeString(res);
        textLog.setText(str);
    }
});
btnCancelSwipe.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        String str = new String();
        int res = myUniPayReader.msr_cancelMSRSwipe();
        if(res == 0)
        {
            str = "End to MSR Swipe successfully.";
        }
        else
            str = myUniPayReader.device_getResponseCodeString(res);
        textLog.setText(str);
    }
```

```
});
btnGetFirmware.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        StringBuilder version = new StringBuilder();
        int res = myUniPayReader.device_getFirmwareVersion(version);
        String str = new String();
        if(res == 0)
        {
            str = "Firmware = " + version.toString();
        }
        else
            str = myUniPayReader.device_getResponseCodeString(res);
        textLog.setText(str);
    }
});
```

5.7.6 Complete code listing

```
package com.example.unipay_tutorial;

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

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

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.DEVICE_INTERFACE_Types;
import com.idtechproducts.device.ReaderInfo.EVENT_MSR_Types;



public class MainActivity extends Activity implements  OnReceiverListener {

    // declaring the instance of the UniPayReader;
    private IDT_Device myUniPayReader = null;
    private TextView connectStatusTextView;
    private TextView textLog;
    private Button btnConnect;
    private Button btnDisconnect;
    private Button btnGetFirmware;
    private Button btnStartSwipe;
    private Button btnCancelSwipe;
    private Handler handler = new Handler();
    private boolean isReaderConnected = false;
    private String strCard = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        handler = new Handler();
        btnConnect = (Button)findViewById(R.id.btn_Connect);
        btnDisconnect = (Button)findViewById(R.id.btn_Disconnect);
        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);
        connectStatusTextView = (TextView)findViewById(R.id.status_text);

        if(myUniPayReader!=null){
            myUniPayReader.unregisterListen();
            myUniPayReader.release();
            myUniPayReader = null;
        }
        myUniPayReader = new IDT_Device(this,this);
        myUniPayReader.registerListen();

        loadXMLfile();


        btnConnect.setOnClickListener(new OnClickListener(){
```

```
            public void onClick(View v) {
                myUniPayReader.device_connect();
            }
        });

        btnDisconnect.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                myUniPayReader.device_disconnect();
            }
        });

        btnStartSwipe.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                int res = myUniPayReader.msr_startMSRSwipe();
                if(res == 0)
                {
                    str = "Start to MSR Swipe successfully,please swipe card.";
                }
                else
                    str = myUniPayReader.device_getResponseCodeString(res);
                textLog.setText(str);
            }
        });
        btnCancelSwipe.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                int res = myUniPayReader.msr_cancelMSRSwipe();
                if(res == 0)
                {
                    str = "End to MSR Swipe successfully.";
                }
                else
                    str = myUniPayReader.device_getResponseCodeString(res);
                textLog.setText(str);
            }
        });
        btnGetFirmware.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                StringBuilder version = new StringBuilder();
                int res = myUniPayReader.device_getFirmwareVersion(version);
                String str = new String();
                if(res == 0)
                {
                    str = "Firmware = " + version.toString();
                }
                else
                    str = myUniPayReader.device_getResponseCodeString(res);
                textLog.setText(str);
            }
        });




        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN, WindowManager.LayoutParams.
      FLAG_FULLSCREEN);
        myUniPayReader.log_setVerboseLoggingEnable(true);
    }


    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);
    }


    private Runnable doUpdateStatus = new Runnable()
    {
        public void run()
        {
            if(!isReaderConnected){
                connectStatusTextView.setText("UNIPAY DISCONNECTED");
            }
            else{
                connectStatusTextView.setText("UNIPAY CONNECTED");
            }
        }
    };

    @Override
    public void deviceConnected() {
        isReaderConnected = true;
        handler.post(doUpdateStatus);
    }

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

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

    }

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

    }


    @Override
    public boolean getUserGrant(USER_GRANT_TYPE arg0, String arg1) {
        // TODO Auto-generated method stub
        return false;
    }
```

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

}

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

}


@Override
public void plugStatusChange(DEVICE_INTERFACE_Types interfaceType, boolean deviceInserted) {
    // TODO Auto-generated method stub

}

private Runnable m_CardRunnable = new Runnable()
{
    public void run()
    {
        textLog.setText(strCard);
    }
};
public void swipeMSRData(IDTMSRData card)
{
    String str = new String();
    if( card.event == EVENT_MSR_Types.EVENT_MSR_CARD_DATA ||card.event == EVENT_MSR_Types.
  EVENT_MSR_DATA_ERROR)
    {
        int len = card.cardData.length;
        int i = 0;
        str = "CardData:\r\n";
        for(i = 0; i < len; i++)
        {
            str += String.format("%02x ", card.cardData[i]);
        }
        if(card.track1 != "")
        {
            str += "\r\nTrack1 data:\r\n";
            str += card.track1;
        }
        if(card.track2 != "")
        {
            str += "\r\nTrack2 data:\r\n";
            str += card.track2;
        }
        if(card.track3 != "")
        {
            str += "\r\nTrack3 data:\r\n";
            str += card.track3;
        }
        len = card.encTrack1.length;
        if(len > 0)
        {
            str += "\r\nTrack1 encrypted data:\r\n";
            for(i = 0; i < len; i++)
            {
                str += String.format("%02X", card.encTrack1[i]);
            }
        }
        len = card.encTrack2.length;
        if(len > 0)
        {
            str += "\r\nTrack2 encrypted data:\r\n";
            for(i = 0; i < len; i++)
            {
                str += String.format("%02X", card.encTrack2[i]);
            }
        }
        len = card.encTrack3.length;
        if(len > 0)
        {
            str += "\r\nTrack3 encrypted data:\r\n";
            for(i = 0; i < len; i++)
            {
                str += String.format("%02X", card.encTrack3[i]);
            }
        }
        len = card.KSN.length;
        if(len > 0)
        {
            str += "\r\nKSN:";
            for(i = 0; i < len; i++)
```

```
                {
                    str += String.format("%02x ", card.KSN[i]);
                }
            }
            str += "\r\nCard Type:";
            if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_ISOABA)
                str += String.format("ISO 7813/ISO 4909/ABA format.");
            else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_AAMVA)
                str += String.format("AAMVA format.");
            else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_Raw)
                str += String.format("Raw; un-decoded format.");
            else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_JisI_II)
                str += String.format("JIS I or JIS II.");
            else
                str += String.format("Other format.");
            str += "\r\nEncode Status:";
            str += String.format("%02x ", card.captureEncodeStatus);
            str += "\r\nEncrypt Type:";
            if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_TDES)
                str += String.format("TDES");
            if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_AES)
                str += String.format("AES");
            if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_NONE)
                str += String.format("None");
            if(card.serialNumber != "")
            {
                str += "\r\nReader Serial Number:";
                str += card.serialNumber;
            }


        }
        else if(card.event == EVENT_MSR_Types.EVENT_MSR_BACKSPACE_KEY)
            str = "Press Backspace Key.";
        else if(card.event == EVENT_MSR_Types.EVENT_MSR_CANCEL_KEY)
            str = "Press Cancel Key.";
        else if(card.event == EVENT_MSR_Types.EVENT_MSR_ENTER_KEY)
            str = "Press Enter Key.";
        else
        {
            str = "Swipe Card error, error code is ";
            str += String.format("%02X %02X ", card.cardData[0],card.cardData[1]);
        }
        strCard = str;
        handler.post(m_CardRunnable);
    }

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

    }

}
```

Chapter 6

Core Implementation UniPayEMV

UniPayEMV.jar combined with libemv_kernel.so is a Certified Level 2 EMV Kernel for UniPay on Android. It is an add-on library to the IDTechSDK.jar library. An existing UniPay project must be established with IDTechSDK.jar before UniPayEMV can be incorporated with it.

6.1 Integrating with IDTechSDK project

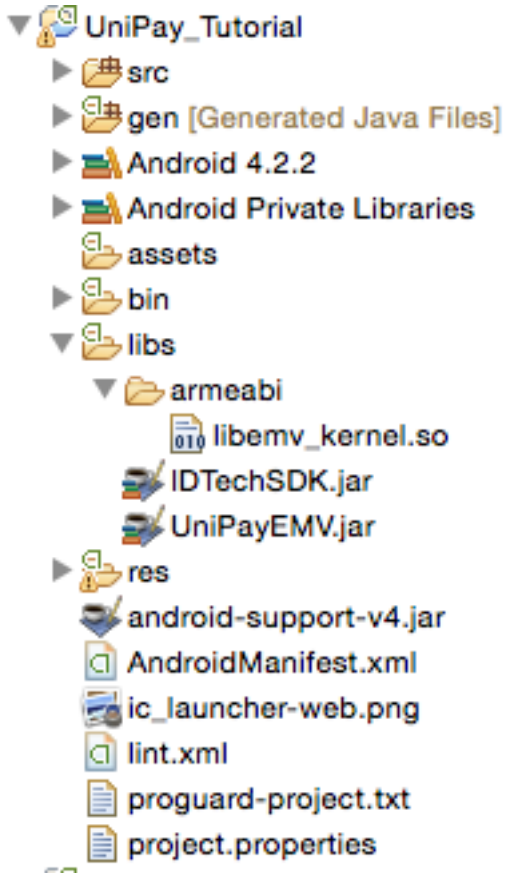
- Import the necessary libraries
- Add Import statements to utilize libraries
- Implement the UniPayEMVDelegates
- Implement optional delegate protocols
- Allocate/initialize IDT_BTPay objects
- Sample Project Tutorial

6.2 Import the necessary libraries

Adding EMV functionality to an existing UniPay project requires the following files to be added:

- UniPayEMV.jar
- libemv_kernel.so

The UniPayEMV.jar should be located in the libs folder, and the libemv_kernel.so must be placed in the armeabi folder withing the libs folder.



6.3 Add Import statements to utilize libraries

In the header files of the classes that will access the device, use import statement utilize the frameworks:

```
import com.idtechproducts.emv.UniPayEMV;
import com.idtechproducts.emv.UniPayEMV.IDTEMVData;
import com.idtechproducts.emv.UniPayEMV.MESSAGE_Types;
```

6.4 Implement the UniPayEMVDelegates:

In the activity that will be a delegate of UniPayEVM.framework, implement UniPayEMVDelegate and include all the methods:

```
public class MainActivity extends Activity implements  OnReceiverListener, UniPayEMV.UniPayEMVDelegate
```

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

}


@Override
public void confirmApplicationSelection(Vector<String> arg0, boolean arg1) {
    // TODO Auto-generated method stub

}
```

```
@Override
public void emvTransactionData(IDTEMVData arg0,
        com.idtechproducts.emv.UniPayEMV.EMV_RESULT_CODE_Types arg1,
        boolean arg2) {
    // TODO Auto-generated method stub

}


@Override
public void emvTransactionMessage(MESSAGE_Types arg0) {
    // TODO Auto-generated method stub

}


@Override
public void languagePreference(byte[] arg0) {
    // TODO Auto-generated method stub

}


@Override
public void swipeMSRDataEMV(IDTMSRData arg0, Map<String, byte[]> arg1) {
    // TODO Auto-generated method stub

}
```

6.5 Create an instance of the UniPayEMV class:

To use a device, create an instance of the UniPayEMV class by passing the activity context, the IDT_Device SDK instance, and the UniPayEMV delegate.

```
private UniPayEMV myUniPayEMV = null;

   @Override
   protected void onCreate(Bundle savedInstanceState) {

       . . . . .

       if(myUniPayReader!=null){
           myUniPayReader.unregisterListen();
           myUniPayReader.release();
           myUniPayReader = null;
       }
       myUniPayReader = new IDT_Device(this,this);
       myUniPayReader.registerListen();

       if(myUniPayEMV!=null){
           myUniPayEMV = null;
       }
       myUniPayEMV = new UniPayEMV(this,myUniPayReader,this);

       . . . . .
   }
```

6.6 Sample Project Tutorial

We will add EMV capabilities to the previous sample project tutorial for UniPay:

- start EMV transaction
- complete EMV transaction
- cancel EMV transaction

Protocol Delegates:

- Protocol to report EMV card swipes
- Protocol to report EMV data

6.6.1 Step 1: Create Android Sample Project

Create the sample project for UniPay

Sample Project Tutorial

6.6.2 Step 2: Import libraries

Import the necessary libraries

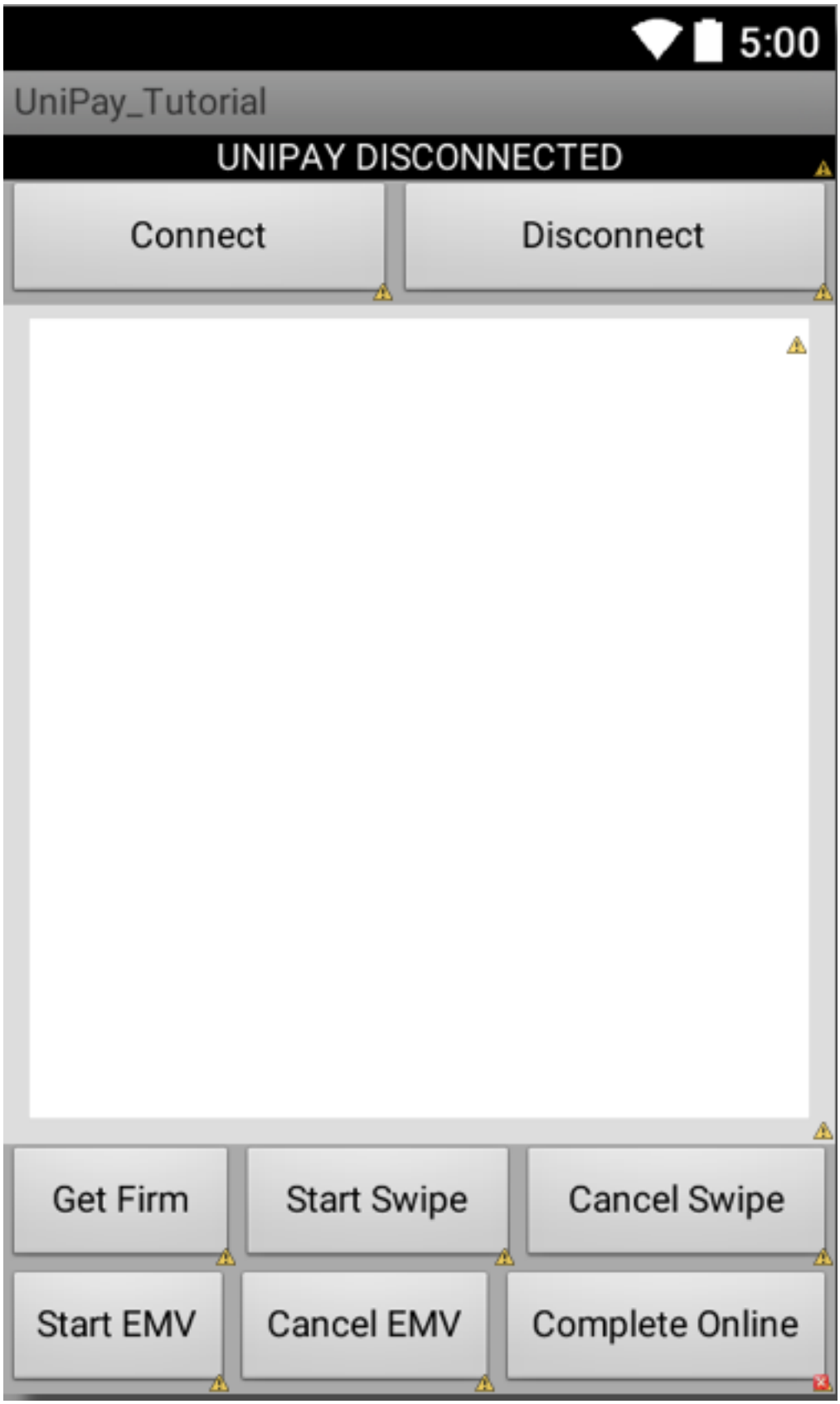
6.6.3 Step 3: Append Interface

Add to User Interface by editing the layout xml file
Open your main layout file, and add three more buttons:

- Add a button to start EMV transaction.

- Add a button to complete EMV transaction.

- Add a button to cancel emv transaction:




```
<?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="UNIPAY DISCONNECTED"
        android:textColor="#FFFFFF" />
    <LinearLayout
        android:id="@+id/linearLayoutBottom2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        >
        <Button
            android:id="@+id/btn_Connect"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:gravity="center_vertical|center_horizontal"
            android:text="Connect" />
        <Button
            android:id="@+id/btn_Disconnect"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Disconnect" />
    </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/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 Firm" />
        <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
        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" />
        <Button
            android:id="@+id/btn_completeEMV"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.53"
            android:text="Complete Online" />
    </LinearLayout>
</LinearLayout>
```

6.6.4 Step 4: Configure Activity

In the activity file, perform the following:

- Add Import statements to utilize libraries
- Implement the UniPayEMVDelegates
- Implement the button press methods

```
private Button btnStartEMV;
private Button btnCancelEMV;
private Button btnCompleteEMV;
```

```
btnStartEMV = (Button)findViewById(R.id.btn_startEMV);
btnCancelEMV = (Button)findViewById(R.id.btn_cancelEMV);
btnCompleteEMV = (Button)findViewById(R.id.btn_completeEMV);
```

```
btnStartEMV.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        String str = new String();
        myUniPayEMV.startEMVTransaction(1.00, 30, 0, null);
            str = "Start EMV Executing";
        textLog.setText(str);
    }
});
btnCancelEMV.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        String str = new String();
        myUniPayEMV.cancelTransaction();
            str = "Cancel EMV Sent";
        textLog.setText(str);
    }
});
btnCompleteEMV.setOnClickListener(new OnClickListener(){
    public void onClick(View v) {
        String str = new String();
        myUniPayEMV.completeOnlineEMVTransaction(EMV_COMPLETION_RESULT.EMV_COMPLETION_RESULT_ACCEPTED, "00"
      , "01020304050607083030", null);
            str = "Complete EMV Sent";
        textLog.setText(str);
    }
});
```

- Implement the EMV delegates

```
public void emvTransactionMessage(MESSAGE_Types message){
        switch (message) {
```

```
            case MESSAGE_INSERT_OR_SWIPE_CARD:
                appendMessageToResults("PLEASE INSERT OR SWIPE CARD");
                break;
            case MESSAGE_REMOVE_CARD:
                appendMessageToResults("PLEASE REMOVE CARD");
                //lcdDisplay.text = @"PLEASE REMOVE CARD";
                break;
            case MESSAGE_BAD_ICC:
                appendMessageToResults("CHIP READ ERROR-USE MSR\nPLEASE REMOVE CARD");
                break;
            case MESSAGE_TRANSACTION_CANCELLED:
                appendMessageToResults("TRANSACTION CANCELLED");
                break;
            case MESSAGE_FALLBACK_FAILED:
                appendMessageToResults("FALLBACK FAILED");
                break;
            case MESSAGE_USE_CHIP_READER:
                appendMessageToResults("MUST USE ICC READER");
                break;
            case MESSAGE_PROCESSING:
                appendMessageToResults("PROCESSING");
                break;
            case MESSAGE_READY:
                appendMessageToResults("UniPay Ready");
                break;
            case MESSAGE_USE_MSR:
                appendMessageToResults("Please Swipe Card");
                break;
            case MESSAGE_SDK_ERROR:
                appendMessageToResults("SDK Error: " + String.valueOf(myUniPayEMV.getLastSDKError()));
                break;
            case MESSAGE_NOT_ACCEPTED:
                appendMessageToResults("Not Accepted");
                break;

            default:
                break;
        }

    }

    public void swipeMSRDataEMV(IDTMSRData card, Map<String, byte[]> tags) {

        StringBuilder sb = new StringBuilder();

        sb.append("Swipe Data Captured for EMV transaction\n");
        sb.append("Track 1: ");
        if (card.track1 != null) if (card.track1.length() > 0) sb.append(card.track1);
        sb.append("\nTrack 2: ");
        if (card.track2 != null) if (card.track2.length() > 0) sb.append(card.track2);
        sb.append("\nTrack 3: ");
        if (card.track3 != null) if (card.track3.length() > 0) sb.append(card.track3);
        sb.append("\n");
        if (card.encTrack1 != null) if (card.encTrack1.length > 0) sb.append("Enc. Track 1: " +
      getHexStringFromBytes(card.encTrack1) + "\n");
        if (card.encTrack2 != null) if (card.encTrack2.length > 0) sb.append("Enc. Track 2: " +
      getHexStringFromBytes(card.encTrack2) + "\n");
        if (card.encTrack3 != null) if (card.encTrack3.length > 0) sb.append("Enc. Track 3: " +
      getHexStringFromBytes(card.encTrack3) + "\n");
        if (card.serialNumber != null) if (card.serialNumber.length() > 0) sb.append("Serial Nunber: " +
      card.serialNumber + "\n");
        if (card.KSN != null) if (card.KSN.length > 0) sb.append("KSN: " + getHexStringFromBytes(card.KSN)
      + "\n");

        sb.append("\n");
        sb.append("Tags:\n");
        for (Map.Entry<String, byte[]> entry : tags.entrySet())
        {
            sb.append(entry.getKey() + ": " + getHexStringFromBytes(entry.getValue()));
            sb.append("\n");
        }


    }

    public void emvTransactionData(IDTEMVData emvData, UniPayEMV.EMV_RESULT_CODE_Types errorCode, boolean
      performReversal) {
        byte[] b = {0};
        if (emvData != null){
            if (emvData.unencryptedTags != null){
                b = emvData.unencryptedTags.get("9F27");
                if (b[0] == 0x01) {

                    appendMessageToResults("ERROR: SERVICE NOT ALLOWED");

                    return;
                }
```

```
        }
    }


    if (performReversal) {
        appendMessageToResults("WARNING: TRANSACTIONS WAS APPROVED ONLINE, BUT TERMINAL DECLINED.
   PLEASE REVERSE/VOID ONLINE APPROVAL FOR THIS TRANSACTION");
    }

    boolean isError =  errorCode.toString().startsWith("EMV_RESULT_CODE_ERROR");

    if (!isError) {

            appendMessageToResults("");
            appendMessageToResults("GENERATE AC RESULTS");

    }
    else {
       appendMessageToResults("ERROR: TRANSACTION TERMINATED");
        if (errorCode == UniPayEMV.EMV_RESULT_CODE_Types.EMV_RESULT_CODE_ERROR_EMV_SERVICE_NOT_ALLOWED)
   {
            appendMessageToResults("NOT ACCEPTED");
        }
        appendMessageToResults("Result = " + errorCode.toString());


        return;
    }







    if ((errorCode == UniPayEMV.EMV_RESULT_CODE_Types.EMV_RESULT_CODE_GO_ONLINE) || (b[0] == 0x88) || (
  b[0] == 0x8B) || (b[0] == 0x8A)) {
         appendMessageToResults("Financial Transaction Request:");

         appendMessageToResults("GO ONLINE");



    }
    else{
        byte cvm[] = null;
        byte tvr[] = null;
        byte tsi[] = null;
        if (emvData != null){
        if (emvData.unencryptedTags != null){
            cvm = emvData.unencryptedTags.get("9F34");

            tvr = emvData.unencryptedTags.get("95");
            tsi = emvData.unencryptedTags.get("9B");
        }
        }
        String cvmString = "";
        String tvrString = "";
        String tsiString = "";
        if  (cvm != null) cvmString = getHexStringFromBytes(cvm);
        if  (tvr != null) tvrString = getHexStringFromBytes(tvr);
        if  (tsi != null) tsiString = getHexStringFromBytes(tsi);


            String results = "SUCCESS - TC";
            if ((b[0] & 0xf0) == 0) {
                results = "DECLINE - AAC";
               appendMessageToResults("Financial Transaction Confirmation:");
            }


    }

    if (emvData != null){
        if (emvData.unencryptedTags != null){
            Map<String, byte[]> treeMap = new TreeMap<String, byte[]>(emvData.unencryptedTags);
            appendMessageToResults("Tags:\n" + mapDescription(treeMap));
        }
        if (emvData.encryptedTags != null){
            Map<String, byte[]> treeMap = new TreeMap<String, byte[]>(emvData.encryptedTags);
            appendMessageToResults("Encrypted Tags:\n" + mapDescription(treeMap));
        }
        if (emvData.maskedTags != null){
            Map<String, byte[]> treeMap = new TreeMap<String, byte[]>(emvData.maskedTags);
            appendMessageToResults("Masked Tags:\n" + mapDescription(treeMap));
        }
    }
```

```
    }
```

Complete Code Listing

```
package com.example.unipay_tutorial;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.util.Map;
import java.util.TreeMap;
import java.util.Vector;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

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.DEVICE_INTERFACE_Types;
import com.idtechproducts.device.ReaderInfo.EVENT_MSR_Types;
import com.idtechproducts.emv.UniPayEMV;
import com.idtechproducts.emv.UniPayEMV.EMV_COMPLETION_RESULT;
import com.idtechproducts.emv.UniPayEMV.IDTEMVData;
import com.idtechproducts.emv.UniPayEMV.MESSAGE_Types;


public class MainActivity extends Activity implements  OnReceiverListener, UniPayEMV.UniPayEMVDelegate  {

    // declaring the instance of the UniPayReader;
    private IDT_Device myUniPayReader = null;
    private TextView connectStatusTextView;
    private TextView textLog;
    private Button btnConnect;
    private Button btnDisconnect;
    private Button btnGetFirmware;
    private Button btnStartSwipe;
    private Button btnCancelSwipe;
    private Button btnStartEMV;
    private Button btnCancelEMV;
    private Button btnCompleteEMV;
    private Handler handler = new Handler();
    private boolean isReaderConnected = false;
    private String strCard = "";

    private UniPayEMV myUniPayEMV = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        handler = new Handler();
        btnStartEMV = (Button)findViewById(R.id.btn_startEMV);
        btnCancelEMV = (Button)findViewById(R.id.btn_cancelEMV);
        btnCompleteEMV = (Button)findViewById(R.id.btn_completeEMV);
        btnConnect = (Button)findViewById(R.id.btn_Connect);
        btnDisconnect = (Button)findViewById(R.id.btn_Disconnect);
        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);
        connectStatusTextView = (TextView)findViewById(R.id.status_text);

        if(myUniPayReader!=null){
            myUniPayReader.unregisterListen();
            myUniPayReader.release();
            myUniPayReader = null;
        }
        myUniPayReader = new IDT_Device(this,this);
        myUniPayReader.registerListen();
        loadXMLfile();

        if(myUniPayEMV!=null){
```

```
            myUniPayEMV = null;
        }
        myUniPayEMV = new UniPayEMV(this,myUniPayReader,this);

        btnConnect.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                myUniPayReader.device_connect();
            }
        });

        btnDisconnect.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                myUniPayReader.device_disconnect();
            }
        });


        btnStartEMV.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                myUniPayEMV.startEMVTransaction(1.00, 30, 0, null);
                    str = "Start EMV Executing";
                textLog.setText(str);
            }
        });
        btnCancelEMV.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                myUniPayEMV.cancelTransaction();
                    str = "Cancel EMV Sent";
                textLog.setText(str);
            }
        });
        btnCompleteEMV.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                myUniPayEMV.completeOnlineEMVTransaction(EMV_COMPLETION_RESULT.
    EMV_COMPLETION_RESULT_ACCEPTED, "00", "11223344556677883030", "");
                    str = "Complete EMV Sent";
                textLog.setText(str);
            }
        });

        btnStartSwipe.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                int res = myUniPayReader.msr_startMSRSwipe();
                if(res == 0)
                {
                    str = "Start to MSR Swipe successfully,please swipe card.";
                }
                else
                    str = myUniPayReader.device_getResponseCodeString(res);
                textLog.setText(str);
            }
        });
        btnCancelSwipe.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                String str = new String();
                int res = myUniPayReader.msr_cancelMSRSwipe();
                if(res == 0)
                {
                    str = "End to MSR Swipe successfully.";
                }
                else
                    str = myUniPayReader.device_getResponseCodeString(res);
                textLog.setText(str);
            }
        });
        btnGetFirmware.setOnClickListener(new OnClickListener(){
            public void onClick(View v) {
                StringBuilder version = new StringBuilder();
                int res = myUniPayReader.device_getFirmwareVersion(version);
                String str = new String();
                if(res == 0)
                {
                    str = "Firmware = " + version.toString();
                }
                else
                    str = myUniPayReader.device_getResponseCodeString(res);
                textLog.setText(str);
            }
        });
```

```
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
    getWindow().setFlags(WindowManager.LayoutParams. FLAG_FULLSCREEN, WindowManager.LayoutParams.
  FLAG_FULLSCREEN);
    myUniPayReader.log_setVerboseLoggingEnable(true);
}


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);
}


private Runnable doUpdateStatus = new Runnable()
{
    public void run()
    {
        if(!isReaderConnected){
            connectStatusTextView.setText("UNIPAY DISCONNECTED");
        }
        else{
            connectStatusTextView.setText("UNIPAY CONNECTED");
        }
    }
};

@Override
public void deviceConnected() {
    isReaderConnected = true;
    handler.post(doUpdateStatus);
}

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

```
}

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

}

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

}


@Override
public boolean getUserGrant(USER_GRANT_TYPE arg0, String arg1) {
    // TODO Auto-generated method stub
    return false;
}

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

}

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

}


@Override
public void plugStatusChange(DEVICE_INTERFACE_Types interfaceType, boolean deviceInserted) {
    // TODO Auto-generated method stub

}

private Runnable m_CardRunnable = new Runnable()
{
    public void run()
    {
        textLog.setText(strCard);
    }
};
public void swipeMSRData(IDTMSRData card)
{
    String str = new String();
    if( card.event == EVENT_MSR_Types.EVENT_MSR_CARD_DATA ||card.event == EVENT_MSR_Types.
  EVENT_MSR_DATA_ERROR)
    {
        int len = card.cardData.length;
        int i = 0;
        str = "CardData:\r\n";
        for(i = 0; i < len; i++)
        {
            str += String.format("%02x ", card.cardData[i]);
        }
        if(card.track1 != "")
        {
            str += "\r\nTrack1 data:\r\n";
            str += card.track1;
        }
        if(card.track2 != "")
        {
            str += "\r\nTrack2 data:\r\n";
            str += card.track2;
        }
        if(card.track3 != "")
        {
            str += "\r\nTrack3 data:\r\n";
            str += card.track3;
        }
```

```
            len = card.encTrack1.length;
            if(len > 0)
            {
                str += "\r\nTrack1 encrypted data:\r\n";
                for(i = 0; i < len; i++)
                {
                    str += String.format("%02X", card.encTrack1[i]);
                }
            }
            len = card.encTrack2.length;
            if(len > 0)
            {
                str += "\r\nTrack2 encrypted data:\r\n";
                for(i = 0; i < len; i++)
                {
                    str += String.format("%02X", card.encTrack2[i]);
                }
            }
            len = card.encTrack3.length;
            if(len > 0)
            {
                str += "\r\nTrack3 encrypted data:\r\n";
                for(i = 0; i < len; i++)
                {
                    str += String.format("%02X", card.encTrack3[i]);
                }
            }
            len = card.KSN.length;
            if(len > 0)
            {
                str += "\r\nKSN:";
                for(i = 0; i < len; i++)
                {
                    str += String.format("%02x ", card.KSN[i]);
                }
            }
            str += "\r\nCard Type:";
            if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_ISOABA)
                str += String.format("ISO 7813/ISO 4909/ABA format.");
            else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_AAMVA)
                str += String.format("AAMVA format.");
            else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_Raw)
                str += String.format("Raw; un-decoded format.");
            else if(card.cardType == CAPTURE_ENCODE_TYPE.CAPTURE_ENCODE_TYPE_JisI_II)
                str += String.format("JIS I or JIS II.");
            else
                str += String.format("Other format.");
            str += "\r\nEncode Status:";
            str += String.format("%02x ", card.captureEncodeStatus);
            str += "\r\nEncrypt Type:";
            if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_TDES)
                str += String.format("TDES");
            if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_AES)
                str += String.format("AES");
            if(card.captureEncryptType == CAPTURE_ENCRYPT_TYPE.CAPTURE_ENCRYPT_TYPE_NONE)
                str += String.format("None");
            if(card.serialNumber != "")
            {
                str += "\r\nReader Serial Number:";
                str += card.serialNumber;
            }


        }
        else if(card.event == EVENT_MSR_Types.EVENT_MSR_BACKSPACE_KEY)
            str = "Press Backspace Key.";
        else if(card.event == EVENT_MSR_Types.EVENT_MSR_CANCEL_KEY)
            str = "Press Cancel Key.";
        else if(card.event == EVENT_MSR_Types.EVENT_MSR_ENTER_KEY)
            str = "Press Enter Key.";
        else
        {
            str = "Swipe Card error, error code is ";
            str += String.format("%02X %02X ", card.cardData[0],card.cardData[1]);
        }
        strCard = str;
        handler.post(m_CardRunnable);
    }

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

    }


    @Override
```

```
public void confirmApplicationSelection(Vector<String> arg0, boolean arg1) {
    // TODO Auto-generated method stub

}




@Override
public void languagePreference(byte[] arg0) {
    // TODO Auto-generated method stub

}


private String getHexStringFromBytes(byte []data)
{
    if(data==null) return null;
    if(data.length<=0)
        return null;
    StringBuffer hexString = new StringBuffer();
    String fix = null;
    for (int i = 0; i < data.length; i++) {
        fix = Integer.toHexString(0xFF & data[i]);
        if(fix.length()==1)
            fix = "0"+fix;
        hexString.append(fix);
    }
    fix = null;
    fix = hexString.toString();
    return fix;
}


private void appendMessageToResults(String str){


    final String result = str + "\n";
    runOnUiThread(new Runnable() {
        public void run() {
            textLog.append(result);
        }
    });

}

public void emvTransactionMessage(MESSAGE_Types message){
    switch (message) {
        case MESSAGE_INSERT_OR_SWIPE_CARD:
            appendMessageToResults("PLEASE INSERT OR SWIPE CARD");
            break;
        case MESSAGE_REMOVE_CARD:
            appendMessageToResults("PLEASE REMOVE CARD");
            //lcdDisplay.text = @"PLEASE REMOVE CARD";
            break;
        case MESSAGE_BAD_ICC:
            appendMessageToResults("CHIP READ ERROR-USE MSR\nPLEASE REMOVE CARD");
            break;
        case MESSAGE_TRANSACTION_CANCELLED:
            appendMessageToResults("TRANSACTION CANCELLED");
            break;
        case MESSAGE_FALLBACK_FAILED:
            appendMessageToResults("FALLBACK FAILED");
            break;
        case MESSAGE_USE_CHIP_READER:
            appendMessageToResults("MUST USE ICC READER");
            break;
        case MESSAGE_PROCESSING:
            appendMessageToResults("PROCESSING");
            break;
        case MESSAGE_READY:
            appendMessageToResults("UniPay Ready");
            break;
        case MESSAGE_USE_MSR:
            appendMessageToResults("Please Swipe Card");
            break;
        case MESSAGE_SDK_ERROR:
            appendMessageToResults("SDK Error: " + String.valueOf(myUniPayEMV.getLastSDKError()));
            break;
        case MESSAGE_NOT_ACCEPTED:
            appendMessageToResults("Not Accepted");
            break;

        default:
            break;
```

```
        }

    }

    public void swipeMSRDataEMV(IDTMSRData card, Map<String, byte[]> tags) {

        StringBuilder sb = new StringBuilder();

        sb.append("Swipe Data Captured for EMV transaction\n");
        sb.append("Track 1: ");
        if (card.track1 != null) if (card.track1.length() > 0) sb.append(card.track1);
        sb.append("\nTrack 2: ");
        if (card.track2 != null) if (card.track2.length() > 0) sb.append(card.track2);
        sb.append("\nTrack 3: ");
        if (card.track3 != null) if (card.track3.length() > 0) sb.append(card.track3);
        sb.append("\n");
        if (card.encTrack1 != null) if (card.encTrack1.length > 0) sb.append("Enc. Track 1: " +
      getHexStringFromBytes(card.encTrack1) + "\n");
        if (card.encTrack2 != null) if (card.encTrack2.length > 0) sb.append("Enc. Track 2: " +
      getHexStringFromBytes(card.encTrack2) + "\n");
        if (card.encTrack3 != null) if (card.encTrack3.length > 0) sb.append("Enc. Track 3: " +
      getHexStringFromBytes(card.encTrack3) + "\n");
        if (card.serialNumber != null) if (card.serialNumber.length() > 0) sb.append("Serial Nunber: " +
      card.serialNumber + "\n");
        if (card.KSN != null) if (card.KSN.length > 0) sb.append("KSN: " + getHexStringFromBytes(card.KSN)
      + "\n");

        sb.append("\n");
        sb.append("Tags:\n");
        for (Map.Entry<String, byte[]> entry : tags.entrySet())
        {
            sb.append(entry.getKey() + ": " + getHexStringFromBytes(entry.getValue()));
            sb.append("\n");
        }


    }

    public void emvTransactionData(IDTEMVData emvData, UniPayEMV.EMV_RESULT_CODE_Types errorCode, boolean
      performReversal) {
        byte[] b = {0};
        if (emvData != null){
            if (emvData.unencryptedTags != null){
                b = emvData.unencryptedTags.get("9F27");
                if (b[0] == 0x01) {

                    appendMessageToResults("ERROR: SERVICE NOT ALLOWED");

                    return;
                }
            }
        }


        if (performReversal) {
            appendMessageToResults("WARNING: TRANSACTIONS WAS APPROVED ONLINE, BUT TERMINAL DECLINED.
       PLEASE REVERSE/VOID ONLINE APPROVAL FOR THIS TRANSACTION");
        }

        boolean isError =  errorCode.toString().startsWith("EMV_RESULT_CODE_ERROR");

        if (!isError) {

                appendMessageToResults("");
                appendMessageToResults("GENERATE AC RESULTS");

        }
        else {
           appendMessageToResults("ERROR: TRANSACTION TERMINATED");
            if (errorCode == UniPayEMV.EMV_RESULT_CODE_Types.EMV_RESULT_CODE_ERROR_EMV_SERVICE_NOT_ALLOWED)
       {
                appendMessageToResults("NOT ACCEPTED");
            }
            appendMessageToResults("Result = " + errorCode.toString());


            return;
        }




        if ((errorCode == UniPayEMV.EMV_RESULT_CODE_Types.EMV_RESULT_CODE_GO_ONLINE) || (b[0] == 0x88) || (
      b[0] == 0x8B) || (b[0] == 0x8A)) {
```

```
            appendMessageToResults("Financial Transaction Request:");

            appendMessageToResults("GO ONLINE");



        }
        else{
            byte cvm[] = null;
            byte tvr[] = null;
            byte tsi[] = null;
            if (emvData != null){
            if (emvData.unencryptedTags != null){
                cvm = emvData.unencryptedTags.get("9F34");

                tvr = emvData.unencryptedTags.get("95");
                tsi = emvData.unencryptedTags.get("9B");
            }
            }
            String cvmString = "";
            String tvrString = "";
            String tsiString = "";
            if  (cvm != null) cvmString = getHexStringFromBytes(cvm);
            if  (tvr != null) tvrString = getHexStringFromBytes(tvr);
            if  (tsi != null) tsiString = getHexStringFromBytes(tsi);


                String results = "SUCCESS - TC";
                if ((b[0] & 0xf0) == 0) {
                    results = "DECLINE - AAC";
                   appendMessageToResults("Financial Transaction Confirmation:");
                }


        }

        if (emvData != null){
            if (emvData.unencryptedTags != null){
                Map<String, byte[]> treeMap = new TreeMap<String, byte[]>(emvData.unencryptedTags);
                appendMessageToResults("Tags:\n" + mapDescription(treeMap));
            }
            if (emvData.encryptedTags != null){
                Map<String, byte[]> treeMap = new TreeMap<String, byte[]>(emvData.encryptedTags);
                appendMessageToResults("Encrypted Tags:\n" + mapDescription(treeMap));
            }
            if (emvData.maskedTags != null){
                Map<String, byte[]> treeMap = new TreeMap<String, byte[]>(emvData.maskedTags);
                appendMessageToResults("Masked Tags:\n" + mapDescription(treeMap));
            }
        }

    }


    private String mapDescription(Map <String,byte[]> tags){
        if (tags == null) return "";
        StringBuilder sb = new StringBuilder();
        for (Map.Entry<String, byte[]> entry : tags.entrySet())
        {
            sb.append(entry.getKey() + ": " + getHexStringFromBytes(entry.getValue()));
            sb.append("\n");
        }
        return sb.toString();

    }


}
```

Chapter 7

Enumeration Reference

IDTechEMV

```
public enum EMV_RESULT_CODE_Types{
        EMV_RESULT_CODE_APPROVED(0X00),
        EMV_RESULT_CODE_DECLINED(0X01),
        EMV_RESULT_CODE_GO_ONLINE(0X02),
        EMV_RESULT_CODE_FAILED(0X03),
        EMV_RESULT_CODE_SYSTEM_ERROR(0X05),
        EMV_RESULT_CODE_NOT_ACCEPT(0X07),
        EMV_RESULT_CODE_FALLBACK(0X0A),
        EMV_RESULT_CODE_CANCEL(0X0C),
        EMV_RESULT_CODE_OTHER_ERROR(0X0F),
        EMV_RESULT_CODE_TIME_OUT(0X0D),
        EMV_RESULT_CODE_OFFLINE_APPROVED(0X10),
        EMV_RESULT_CODE_OFFLINE_DECLINED(0X11),
        EMV_RESULT_CODE_REFERRAL_PROCESSING(0X12),
        EMV_RESULT_CODE_ERROR_APP_PROCESSING(0X13),
        EMV_RESULT_CODE_ERROR_APP_READING(0X14),
        EMV_RESULT_CODE_ERROR_DATA_AUTH(0X15),
        EMV_RESULT_CODE_ERROR_PROCESSING_RESTRICTIONS(0X16),
        EMV_RESULT_CODE_ERROR_CVM_PROCESSING(0X17),
        EMV_RESULT_CODE_ERROR_RISK_MGMT(0X18),
        EMV_RESULT_CODE_ERROR_TERM_ACTION_ANALYSIS(0X19),
        EMV_RESULT_CODE_ERROR_CARD_ACTION_ANALYSIS(0X1A),
        EMV_RESULT_CODE_ERROR_APP_SELECTION_TIMEOUT(0X1B),
        EMV_RESULT_CODE_ERROR_DATA_LEN_INCORRECT(0X1C),
        EMV_RESULT_CODE_CALL_YOUR_BANK(0X1D),
        EMV_RESULT_CODE_NO_ICC_ON_CARD(0X1E),
        EMV_RESULT_CODE_NEW_SELECTION(0X1F),
        EMV_RESULT_CODE_START_TRANSACTION_SUCCESS(0X20),
        EMV_RESULT_CODE_APPROVED_WITH_ADVISE_NO_REASON(0X21),
        EMV_RESULT_CODE_APPROVED_WITH_ADVISE_IA_FAILED(0X22),
        EMV_RESULT_CODE_ERROR_POWER_CARD_ERROR(0x23),
        EMV_RESULT_CODE_ERROR_TRANSACTION_AMOUNT_INCORRECT(0x24),
        EMV_RESULT_CODE_PROCESS_OK(0x100),
        EMV_RESULT_CODE_ERROR_INVALID_ARG(0x101),
        EMV_RESULT_CODE_ERROR_FILE_OPEN_FAILED(0x301),
        EMV_RESULT_CODE_ERROR_FILE_OPERATION_FAILED(0x302),
        EMV_RESULT_CODE_ERROR_FILE_MEMORY_NOT_ENOUGH(0x351),
        EMV_RESULT_CODE_SMARTCARD_OK(0x401),
        EMV_RESULT_CODE_ERROR_SMARTCARD_FAIL(0x402),
        EMV_RESULT_CODE_ERROR_SMARTCARD_INIT_FAILED(0x403),
        EMV_RESULT_CODE_ERROR_FALLBACK_SITUATION(0x404),
        EMV_RESULT_CODE_ERROR_EMV_PARSING_TAGS_FAILED(0x601),
        EMV_RESULT_CODE_ERROR_EMV_DUPLICATE_CARD_DATA_ELEMENT(0x602),
        EMV_RESULT_CODE_ERROR_EMV_DATA_FORMAT_INCORRECT(0x603),
        EMV_RESULT_CODE_ERROR_EMV_NO_TERM_APP(0x604),
        EMV_RESULT_CODE_ERROR_EMV_NO_MATCHING_APP(0x605),
        EMV_RESULT_CODE_ERROR_EMV_MISSING_MANDATORY_OBJECT(0x606),
        EMV_RESULT_CODE_ERROR_EMV_APP_SELECTION_RETRY(0x607),
        EMV_RESULT_CODE_ERROR_EMV_GET_AMOUNT_ERROR(0x608),
        EMV_RESULT_CODE_ERROR_EMV_CARD_REJECTED(0x609),
        EMV_RESULT_CODE_ERROR_EMV_AIP_NOT_RECEIVED(0x610),
        EMV_RESULT_CODE_ERROR_EMV_AFL_NOT_RECEIVED(0x611),
        EMV_RESULT_CODE_ERROR_EMV_AFL_LEN_OUT_OF_RANGE(0x612),
        EMV_RESULT_CODE_ERROR_EMV_SFI_OUT_OF_RANGE(0x613),
        EMV_RESULT_CODE_ERROR_EMV_AFL_INCORRECT(0x614),
        EMV_RESULT_CODE_ERROR_EMV_EXP_DATE_INCORRECT(0x615),
        EMV_RESULT_CODE_ERROR_EMV_EFF_DATE_INCORRECT(0x616),
        EMV_RESULT_CODE_ERROR_EMV_ISS_COD_TBL_OUT_OF_RANGE(0x617),
```

```
        EMV_RESULT_CODE_ERROR_EMV_CRYPTOGRAM_TYPE_INCORRECT(0x618),
        EMV_RESULT_CODE_ERROR_EMV_PSE_NOT_SUPPORTED_BY_CARD(0x619),
        EMV_RESULT_CODE_ERROR_EMV_USER_SELECTED_LANGUAGE(0x620),
        EMV_RESULT_CODE_ERROR_EMV_SERVICE_NOT_ALLOWED(0x621),
        EMV_RESULT_CODE_ERROR_EMV_NO_TAG_FOUND(0x622),
        EMV_RESULT_CODE_ERROR_EMV_CARD_BLOCKED(0x623),
        EMV_RESULT_CODE_ERROR_EMV_LEN_INCORRECT(0x624),
        EMV_RESULT_CODE_ERROR_CARD_COM_ERROR(0x625),
        EMV_RESULT_CODE_ERROR_EMV_TSC_NOT_INCREASED(0x626),
        EMV_RESULT_CODE_ERROR_EMV_HASH_INCORRECT(0x627),
        EMV_RESULT_CODE_ERROR_EMV_NO_ARC(0x628),
        EMV_RESULT_CODE_ERROR_EMV_INVALID_ARC(0x629),
        EMV_RESULT_CODE_ERROR_EMV_NO_ONLINE_COMM(0x630),
        EMV_RESULT_CODE_ERROR_TRAN_TYPE_INCORRECT(0x631),
        EMV_RESULT_CODE_ERROR_TRAN_AMOUNT_INCORRECT(0x632),
        EMV_RESULT_CODE_ERROR_CVM_TYPE_UNKNOWN(0x701),
        EMV_RESULT_CODE_ERROR_CVM_AIP_NOT_SUPPORTED(0x702),
        EMV_RESULT_CODE_ERROR_CVM_TAG_8E_MISSING(0x703),
        EMV_RESULT_CODE_ERROR_CVM_TAG_8E_FORMAT_ERROR(0x704),
        EMV_RESULT_CODE_ERROR_CVM_CODE_IS_NOT_SUPPORTED(0x705),
        EMV_RESULT_CODE_ERROR_CVM_COND_CODE_IS_NOT_SUPPORTED(0x706),
        EMV_RESULT_CODE_ERROR_NO_MORE_CVM(0x707),
        EMV_RESULT_CODE_ERROR_PIN_BYPASSED_BEFORE(0x708),
        EMV_RESULT_CODE_ERROR_PK_BUFFER_SIZE_TOO_BIG(0x801),
        EMV_RESULT_CODE_ERROR_PK_FILE_WRITE_ERROR(0x802),
        EMV_RESULT_CODE_ERROR_PK_HASH_ERROR(0x803),
        EMV_RESULT_CODE_ERROR_EMV_APP_SELECTION_RETRY2(0x1101),
        EMV_RESULT_CODE_ERROR_NO_CARD_HOLDER_CONFIRMATION(0x1102),
        EMV_RESULT_CODE_ERROR_GET_ONLINE_PIN(0x1103);
    };
```

```
public enum MESSAGE_Types{
    MESSAGE_INSERT_OR_SWIPE_CARD(0),
    MESSAGE_REMOVE_CARD(1),
    MESSAGE_BAD_ICC(2),
    MESSAGE_TRANSACTION_CANCELLED(3),
    MESSAGE_FALLBACK_FAILED(4),
    MESSAGE_USE_CHIP_READER(5),
    MESSAGE_PROCESSING(6),
    MESSAGE_READY(7),
    MESSAGE_USE_MSR(8),
    MESSAGE_NOT_ACCEPTED(9),
    MESSAGE_SDK_ERROR(10);
};
```

```
public enum EMV_COMPLETION_RESULT{
    EMV_COMPLETION_RESULT_ACCEPTED(0X00),
    EMV_COMPLETION_RESULT_UNABLE_TO_GO_ONLINE(0X01),
    EMV_COMPLETION_RESULT_TECHNICAL_ISSUE(0X02),
    EMV_COMPLETION_RESULT_DECLINED(0X03),
    EMV_COMPLETION_RESULT_ISSUER_REFERAL(0X04);
} ;
```

IDTMSRData

```
public 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;
```

```
public enum CAPTURE_ENCRYPT_TYPE{
    CAPTURE_ENCRYPT_TYPE_TDES(0),
    CAPTURE_ENCRYPT_TYPE_AES(1);
} CAPTURE_ENCRYPT_TYPE;
```

Chapter 8

UniPay Error Code Reference

0000	No error, beginning task
0001	No response from reader
0002	Invalid response data
0003	Time out for task or CMD
0004	Wrong parameter
0005	SDK is doing MSR or ICC task
0006	SDK is doing PINPad task
0007	SDK is doing Other task
0300	Key Type(TDES) of Session Key is not same as the related Master Key.
0400	Related Key was not loaded.
0500	Key Same.
0702	PAN is Error Key.
0D00	This Key had been loaded.
0E00	Base Time was loaded.
1800	Send "Cancel Command" after send "Get Encrypted PIN" &"Get Numeric "& "Get Amount"
1900	Press "Cancel" key after send "Get Encrypted PIN" &"Get Numeric "& "Get Amount"
30FF	Security Chip is not connect
3000	Security Chip is deactivation & Device is In Removal Legally State.
3101	Security Chip is activation & Device is In Removal Legally State.
5500	No Admin DUKPT Key.
5501	Admin DUKPT Key STOP.
5502	Admin DUKPT Key KSN is Error.
5503	Get Authentication Code1 Failed.
5504	Validate Authentication Code Error.
5505	Encrypt or Decrypt data failed.
5506	Not Support the New Key Type.
5507	New Key Index is Error.
5508	Step Error.
550F	Other Error.
6000	Save or Config Failed / Or Read Config Error.
6200	No Serial Number.
6900	Invalid Command - Protocol is right, but task ID is invalid.
6A00	Unsupported Command - Protocol and task ID are right, but command is invalid.
6B00	Unknown parameter in command - Protocol task ID and command are right, but parameter is invalid.
7200	Device is suspend (MKSK suspend or press password suspend).
7300	PIN DUKPT is STOP (21 bit 1).
7400	Device is Busy.
E100	Can not enter sleep mode.
E200	File has existed.
E300	File has not existed.
E400	Open File Error.
E500	SmartCard Error.
E600	Get MSR Card data is error.
E700	Command time out.
E800	File read or write is error.
E900	Active 1850 error!
EA00	Load bootloader error.
EF00	Protocol Error- STX or ETX or check error.
EB00	Picture is not exist.
2C06	no card seated to request ATR
2D01	Card Not Supported,
2D03	Card Not Supported, wants CRC
690D	Command not supported on reader without ICC support
8100	ICC error time out on power-up
8200	invalid TS character received
8500	pps confirmation error
8600	Unsupported F, D, or combination of F and D
8700	protocol not supported EMV TD1 out of range
8800	power not at proper level
8900	ATR length too long
8B01	EMV invalid TA1 byte value
8B02	EMV TB1 required
8B03	EMV Unsupported TB1 only 00 allowed

8B04	EMV Card Error, invalid BWI or CWI
8B06	EMV TB2 not allowed in ATR
8B07	EMV TC2 out of range
8B08	EMV TC2 out of range
8B09	per EMV96 TA3 must be > 0xF
8B10	ICC error on power-up
8B11	EMV T=1 then TB3 required
8B12	Card Error, invalid BWI or CWI
8B13	Card Error, invalid BWI or CWI
8B17	EMV TC1/TB3 conflict*
8B20	EMV TD2 out of range must be T=1
8C00	TCK error
A304	connector has no voltage setting
A305	ICC error on power-up invalid (SBLK(IFSD) exchange
E301	ICC error after session start
FF00	Request to go online
FF01	EMV: Accept the offline transaction
FF02	EMV: Decline the offline transaction
FF03	EMV: Accept the online transaction
FF04	EMV: Decline the online transaction
FF05	EMV: Application may fallback to magstripe technology
FF06	EMV: ICC detected tah the conditions of use are not satisfied
FF07	EMV: ICC didn't accept transaction
FF08	EMV: Transaction was cancelled
FF09	EMV: Application was not selected by kernel or ICC format error or ICC missing data error
FF0A	EMV: Transaction is terminated
FF0B	EMV: Other EMV Error
FFFF	NO RESPONSE

0008	err response or data
0009	no reader attached
000A	did connection
000B	mono audio is enabled
000C	audio volume is too low
000D	task or CMD be canceled

0E00	Authorization Accepted
0E01	Unable to go online
0E02	Technical Issue
0E03	Declined
0E04	Issuer Referral transaction

0F00	Accept the online transaction
0F01	Decline the online transaction
0F02	Request to go online
0F03	Transaction is terminated
0F05	Application was not selected by kernel or ICC format error or ICC missing data error
0F07	ICC didn't accept transaction
0F0A	Application may fallback to magstripe technology
0F0C	Transaction was cancelled
0F0D	Timeout
0F0F	Other EMV Error
0F10	Accept the offline transaction
0F11	Decline the offline transaction

0F21	ICC detected tah the conditions of use are not satisfied
0F22	No app were found on card matching terminal configuration
0F23	Terminal file does not exist
0F24	CAPK file does not exist
0F25	CRL Entry does not exist
0FFE	Return code when blocking is disabled
0FFF	Return code when command is not applicable on the selected device

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)
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
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
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)
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
CE	Card Issuer Action Code (PayPass) – Online
CF	Card Issuer Action Code (PayPass) – Decline
D1	Currency conversion table
D2	Integrated Data Storage Directory (IDSD)
D3	Additional check table
D5	Application Control
D6	Default ARPC response code
D7	Application Control (PayPass)
D8	AIP (PayPass)
D9	AFL (PayPass)
DA	Static CVC3-TRACK1
DB	Static CVC3-TRACK2
DC	IVCVC3-TRACK1
DD	IVCVC3-TRACK2
DF01	Encrypted PIN Block in Tag 9F62 – ISO 95641 Format 0
DF02	PEK Version Number
DF03	PIN Try Limit
DF04	PIN Try Counter (VSDC Application)
DF05	AIP - For VISA Contactless
DF06	Products permitted
DF07	Offline checks mandated
DF08	UDKmac
DF09	UDKenc
DF0B	Retries Permitted Limit
DF0C	Script Message Update
DF0D	Fleet Issuer Action Code - Default

DF0E	Fleet Issuer Action Code - Denial
DF0F	Fleet Issuer Action Code - Online
DF12	Vehicle Registration Number
DF13	DDA Public Modulus
DF14	Driver Name
DF15	Driver ID
DF16	Max Fill Volume
DF17	DDA Public Modulus Length
DF18	Mileage
DF20	Issuer Proprietary Bitmap (IPB)
DF21	Internet Authentication Flag (IAF)
DF22	Encrypted PEK - RFU
DF23	PEK Key Check Value - RFU
DF24	MDK - Key derivation Index
DF25	VISA DPA – MDK - Key derivation Index
DF26	Encrypted PIN Block – ISO 9564-1 Format 1 PIN Block (Thales P3 Format 05)
DF40	qVSDC AIP
DF41	VSDC AIP
DF42	UDKac
DF43	UDKmac
DF44	UDKenc
DF47	UDKcvc
DF48	UDKac KCV
DF49	UDKmac KCV
DF4A	UDKenc KCV
DF4B	UDKcvc KCV
DF4B	POS Cardholder Interaction Information
DF51	Grand Parent AC
DF52	Parent AC
DF53	Grand Parent MAC
DF54	Parent MAC
DF55	Grand Parent ENC
DF56	Parent ENC/Terminal Action Code - Default
DF57	Terminal Action Code - Decline
DF60	DS Input (Card)
DF60	DDA Component P
DF61	DDA Component Q
DF61	DS Digest H
DF62	DS ODS Info
DF62	DDA Component D1
DF63	DDA Component D2
DF63	DS ODS Term
DF64	DDA Component Q Minus 1 Mod P
DF65	DDA Private Exponent
DF6B	Paypass Contactless
DF79	Dynamic Data Authentication Keys

DF8101	DS Summary 2
DF8102	DS Summary 3
DF8104	Balance Read Before Gen AC
DF8105	Balance Read After Gen AC
DF8106	Data Needed
DF8107	CDOL1 Related Data
DF8108	DS AC Type
DF8109	DS Input (Term)
DF810A	DS ODS Info For Reader
DF810B	DS Summary Status
DF810C	Kernel ID
DF810D	DSVN Term
DF810E	Post-Gen AC Put Data Status
DF810F	Pre-Gen AC Put Data Status
DF8110	Proceed To First Write Flag
DF8111	PDOL Related Data
DF8112	Tags To Read
DF8113	DRDOL Related Data
DF8114	Reference Control Parameter
DF8115	Error Indication
DF8116	User Interface Request Data
DF8117	Card Data Input Capability
DF8118	CVM Capability – CVM Required
DF8119	CVM Capability – No CVM Required
DF811A	Default UDOL
DF811B	Kernel Configuration
DF811C	Max Lifetime of Torn Transaction Log Record
DF811D	Max Number of Torn Transaction Log Records
DF811E	Mag-stripe CVM Capability – CVM Required
DF811F	Security Capability
DF8120	Terminal Action Code – Default
DF8121	Terminal Action Code – Denial
DF8122	Terminal Action Code – Online
DF8123	Reader Contactless Floor Limit
DF8124	Reader Contactless Transaction Limit (No On-device CVM)
DF8125	Reader Contactless Transaction Limit (On-device CVM)
DF8126	Reader CVM Required Limit
DF8127	Time Out Value
DF8128	IDS Status
DF8129	Outcome Parameter Set
DF812A	DD Card (Track1)
DF812B	DD Card (Track2)
DF812C	Mag-stripe CVM Capability – No CVM Required
DF812D	Message Hold Time
DF8130	Hold Time Value
DF8131	Phone Message Table

FF60	Visa International
FF62	Visa Magnetic Stripe
FF63	Visa Quick VSDC
FF8101	Torn Record
FF8102	Tags To Write Before Gen AC
FF8103	Tags To Write After Gen AC
FF8104	Data To Send
FF8105	Data Record
FF8106	Discretionary Data

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 11

Data Structure Index

11.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 12

Data Structure Documentation

12.1 com.idtechproducts.device.APDUResponseStruct Class Reference

Data Fields

- byte SW1
- byte SW2
- boolean hasKSN
- boolean hasEncryption
- byte[ ] response
- byte[ ] ksn

12.1.1 Detailed Description

The class for APDU Response.

12.1.2 Field Documentation

12.1.2.1 boolean com.idtechproducts.device.APDUResponseStruct.hasEncryption

APDU response is encrypted.

12.1.2.2 boolean com.idtechproducts.device.APDUResponseStruct.hasKSN

KSN data read.

12.1.2.3 byte [ ] com.idtechproducts.device.APDUResponseStruct.ksn

Key Seral Number.

12.1.2.4 byte [ ] com.idtechproducts.device.APDUResponseStruct.response

APDU response.

12.1.2.5 byte com.idtechproducts.device.APDUResponseStruct.SW1

APDU response status bytes,2 bytes SW1 and SW2.

12.1.2.6 byte com.idtechproducts.device.APDUResponseStruct.SW2

APDU response status bytes,2 bytes SW1 and SW2.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/APDUResponseStruct.java

12.2 com.idtechproducts.emv.UniPayEMV.CAPTURE_ENCODE_TYPE Enum Reference

Public Member Functions

- String **toString** ()

Data Fields

- **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)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.3 com.idtechproducts.device.ReaderInfo.CAPTURE_ENCODE_TYPE Enum Reference

Data Fields

- **CAPTURE_ENCODE_TYPE_ISOABA**
- **CAPTURE_ENCODE_TYPE_AAMVA**
- **CAPTURE_ENCODE_TYPE_Other**
- **CAPTURE_ENCODE_TYPE_Raw**
- **CAPTURE_ENCODE_TYPE_JisI_II**

12.3.1 Detailed Description

Define Card type.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.4 com.idtechproducts.device.ReaderInfo.CAPTURE_ENCRYPT_TYPE Enum Reference

Data Fields

- **CAPTURE_ENCRYPT_TYPE_TDES**

- **CAPTURE_ENCRYPT_TYPE_AES**
- **CAPTURE_ENCRYPT_TYPE_NONE**

12.4.1 Detailed Description

Define encrypted type.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.5 com.idtechproducts.emv.UniPayEMV.CAPTURE_ENCRYPT_TYPE Enum Reference

Public Member Functions

- String **toString** ()

Data Fields

- **CAPTURE_ENCRYPT_TYPE_TDES** =(0)
- **CAPTURE_ENCRYPT_TYPE_AES** =(1)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.6 com.idtechproducts.emv.UniPayEMV.CAPTURE_TYPE Enum Reference

Public Member Functions

- String **toString** ()

Data Fields

- **CAPTURE_TYPE_CONTACT** =(0)
- **CAPTURE_TYPE_CONTACTLESS** =(1)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.7 com.idtechproducts.device.ReaderInfo.DEVICE_INTERFACE_Types Enum Reference

Data Fields

- **DEVICE_UNKNOWN**
- **DEVICE_AUDIO_JACK**
- **DEVICE_USB**

12.7.1 Detailed Description

Device interface type.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.8 com.idtechproducts.emv.UniPayEMV.EMV_COMPLETION_RESULT Enum Reference

Public Member Functions

- String **toString** ()

Data Fields

- **EMV_COMPLETION_RESULT_ACCEPTED** =(0X00)
- **EMV_COMPLETION_RESULT_UNABLE_TO_GO_ONLINE** =(0X01)
- **EMV_COMPLETION_RESULT_TECHNICAL_ISSUE** =(0X02)
- **EMV_COMPLETION_RESULT_DECLINED** =(0X03)
- **EMV_COMPLETION_RESULT_ISSUER_REFERAL** =(0X04)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

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

Data Fields

- **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:

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/OnReceiverListener.java

12.10 com.idtechproducts.emv.UniPayEMV.EMV_RESULT_CODE_Types Enum Reference

Public Member Functions

- int **getValue** ()
- String **toString** ()

Static Public Member Functions

- static EMV_RESULT_CODE_Types **toValue** (int val)

Data Fields

- **EMV_RESULT_CODE_APPROVED** =(0X00)
- **EMV_RESULT_CODE_DECLINED** =(0X01)
- **EMV_RESULT_CODE_GO_ONLINE** =(0X02)
- **EMV_RESULT_CODE_FAILED** =(0X03)
- **EMV_RESULT_CODE_SYSTEM_ERROR** =(0X05)
- **EMV_RESULT_CODE_NOT_ACCEPT** =(0X07)
- **EMV_RESULT_CODE_FALLBACK** =(0X0A)
- **EMV_RESULT_CODE_CANCEL** =(0X0C)
- **EMV_RESULT_CODE_OTHER_ERROR** =(0X0F)
- **EMV_RESULT_CODE_TIME_OUT** =(0X0D)
- **EMV_RESULT_CODE_OFFLINE_APPROVED** =(0X10)
- **EMV_RESULT_CODE_OFFLINE_DECLINED** =(0X11)
- **EMV_RESULT_CODE_REFERRAL_PROCESSING** =(0X12)
- **EMV_RESULT_CODE_ERROR_APP_PROCESSING** =(0X13)
- **EMV_RESULT_CODE_ERROR_APP_READING** =(0X14)
- **EMV_RESULT_CODE_ERROR_DATA_AUTH** =(0X15)
- **EMV_RESULT_CODE_ERROR_PROCESSING_RESTRICTIONS** =(0X16)
- **EMV_RESULT_CODE_ERROR_CVM_PROCESSING** =(0X17)
- **EMV_RESULT_CODE_ERROR_RISK_MGMT** =(0X18)
- **EMV_RESULT_CODE_ERROR_TERM_ACTION_ANALYSIS** =(0X19)
- **EMV_RESULT_CODE_ERROR_CARD_ACTION_ANALYSIS** =(0X1A)
- **EMV_RESULT_CODE_ERROR_APP_SELECTION_TIMEOUT** =(0X1B)
- **EMV_RESULT_CODE_ERROR_DATA_LEN_INCORRECT** =(0X1C)
- **EMV_RESULT_CODE_CALL_YOUR_BANK** =(0X1D)
- **EMV_RESULT_CODE_NO_ICC_ON_CARD** =(0X1E)
- **EMV_RESULT_CODE_NEW_SELECTION** =(0X1F)
- **EMV_RESULT_CODE_START_TRANSACTION_SUCCESS** =(0X20)
- **EMV_RESULT_CODE_APPROVED_WITH_ADVISE_NO_REASON** =(0X21)
- **EMV_RESULT_CODE_APPROVED_WITH_ADVISE_IA_FAILED** =(0X22)
- **EMV_RESULT_CODE_ERROR_POWER_CARD_ERROR** =(0x23)
- **EMV_RESULT_CODE_ERROR_TRANSACTION_AMOUNT_INCORRECT** =(0x24)
- **EMV_RESULT_CODE_PROCESS_OK** =(0x100)
- **EMV_RESULT_CODE_ERROR_INVALID_ARG** =(0x101)
- **EMV_RESULT_CODE_ERROR_FILE_OPEN_FAILED** =(0x301)
- **EMV_RESULT_CODE_ERROR_FILE_OPERATION_FAILED** =(0x302)
- **EMV_RESULT_CODE_ERROR_FILE_MEMORY_NOT_ENOUGH** =(0x351)
- **EMV_RESULT_CODE_SMARTCARD_OK** =(0x401)
- **EMV_RESULT_CODE_ERROR_SMARTCARD_FAIL** =(0x402)
- **EMV_RESULT_CODE_ERROR_SMARTCARD_INIT_FAILED** =(0x403)

- **EMV_RESULT_CODE_ERROR_FALLBACK_SITUATION** =(0x404)
- **EMV_RESULT_CODE_ERROR_EMV_PARSING_TAGS_FAILED** =(0x601)
- **EMV_RESULT_CODE_ERROR_EMV_DUPLICATE_CARD_DATA_ELEMENT** =(0x602)
- **EMV_RESULT_CODE_ERROR_EMV_DATA_FORMAT_INCORRECT** =(0x603)
- **EMV_RESULT_CODE_ERROR_EMV_NO_TERM_APP** =(0x604)
- **EMV_RESULT_CODE_ERROR_EMV_NO_MATCHING_APP** =(0x605)
- **EMV_RESULT_CODE_ERROR_EMV_MISSING_MANDATORY_OBJECT** =(0x606)
- **EMV_RESULT_CODE_ERROR_EMV_APP_SELECTION_RETRY** =(0x607)
- **EMV_RESULT_CODE_ERROR_EMV_GET_AMOUNT_ERROR** =(0x608)
- **EMV_RESULT_CODE_ERROR_EMV_CARD_REJECTED** =(0x609)
- **EMV_RESULT_CODE_ERROR_EMV_AIP_NOT_RECEIVED** =(0x610)
- **EMV_RESULT_CODE_ERROR_EMV_AFL_NOT_RECEIVED** =(0x611)
- **EMV_RESULT_CODE_ERROR_EMV_AFL_LEN_OUT_OF_RANGE** =(0x612)
- **EMV_RESULT_CODE_ERROR_EMV_SFI_OUT_OF_RANGE** =(0x613)
- **EMV_RESULT_CODE_ERROR_EMV_AFL_INCORRECT** =(0x614)
- **EMV_RESULT_CODE_ERROR_EMV_EXP_DATE_INCORRECT** =(0x615)
- **EMV_RESULT_CODE_ERROR_EMV_EFF_DATE_INCORRECT** =(0x616)
- **EMV_RESULT_CODE_ERROR_EMV_ISS_COD_TBL_OUT_OF_RANGE** =(0x617)
- **EMV_RESULT_CODE_ERROR_EMV_CRYPTOGRAM_TYPE_INCORRECT** =(0x618)
- **EMV_RESULT_CODE_ERROR_EMV_PSE_NOT_SUPPORTED_BY_CARD** =(0x619)
- **EMV_RESULT_CODE_ERROR_EMV_USER_SELECTED_LANGUAGE** =(0x620)
- **EMV_RESULT_CODE_ERROR_EMV_SERVICE_NOT_ALLOWED** =(0x621)
- **EMV_RESULT_CODE_ERROR_EMV_NO_TAG_FOUND** =(0x622)
- **EMV_RESULT_CODE_ERROR_EMV_CARD_BLOCKED** =(0x623)
- **EMV_RESULT_CODE_ERROR_EMV_LEN_INCORRECT** =(0x624)
- **EMV_RESULT_CODE_ERROR_CARD_COM_ERROR** =(0x625)
- **EMV_RESULT_CODE_ERROR_EMV_TSC_NOT_INCREASED** =(0x626)
- **EMV_RESULT_CODE_ERROR_EMV_HASH_INCORRECT** =(0x627)
- **EMV_RESULT_CODE_ERROR_EMV_NO_ARC** =(0x628)
- **EMV_RESULT_CODE_ERROR_EMV_INVALID_ARC** =(0x629)
- **EMV_RESULT_CODE_ERROR_EMV_NO_ONLINE_COMM** =(0x630)
- **EMV_RESULT_CODE_ERROR_TRAN_TYPE_INCORRECT** =(0x631)
- **EMV_RESULT_CODE_ERROR_TRAN_AMOUNT_INCORRECT** =(0x632)
- **EMV_RESULT_CODE_ERROR_CVM_TYPE_UNKNOWN** =(0x701)
- **EMV_RESULT_CODE_ERROR_CVM_AIP_NOT_SUPPORTED** =(0x702)
- **EMV_RESULT_CODE_ERROR_CVM_TAG_8E_MISSING** =(0x703)
- **EMV_RESULT_CODE_ERROR_CVM_TAG_8E_FORMAT_ERROR** =(0x704)
- **EMV_RESULT_CODE_ERROR_CVM_CODE_IS_NOT_SUPPORTED** =(0x705)
- **EMV_RESULT_CODE_ERROR_CVM_COND_CODE_IS_NOT_SUPPORTED** =(0x706)
- **EMV_RESULT_CODE_ERROR_NO_MORE_CVM** =(0x707)
- **EMV_RESULT_CODE_ERROR_PIN_BYPASSED_BEFORE** =(0x708)
- **EMV_RESULT_CODE_ERROR_PK_BUFFER_SIZE_TOO_BIG** =(0x801)
- **EMV_RESULT_CODE_ERROR_PK_FILE_WRITE_ERROR** =(0x802)
- **EMV_RESULT_CODE_ERROR_PK_HASH_ERROR** =(0x803)
- **EMV_RESULT_CODE_ERROR_EMV_APP_SELECTION_RETRY2** =(0x1101)
- **EMV_RESULT_CODE_ERROR_NO_CARD_HOLDER_CONFIRMATION** =(0x1102)
- **EMV_RESULT_CODE_ERROR_GET_ONLINE_PIN** =(0x1103)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.11 com.idtechproducts.device.ErrorCode Class Reference

Static Public Attributes

- static final int SUCCESS =0x0000
- static final int NOT_CONNECT =0x0001
- static final int INVALID_REPONSE_DATA =0x0002
- static final int TIMEOUT_TASK =0x0003
- static final int WRONG_PARAMETER =0x0004
- static final int MSR_MODEL =0x0005
- static final int PIN_MODEL =0x0006
- static final int EMVL2_MODEL =0x0007
- static final int NOTIFYRESPONSE_MODEL =0x0008
- static final int UNKNOW_ERROR =0x0009
- static final int AUTO_CONFIG_MODEL =0x0010
- static final int NO_CONFIG =0x0011
- static final int NO_READER =0x0012
- static final int SDK_BUZY =0x0013
- static final int NOT_SUPPORTED =0x0014
- static final int NO_RESPONSE =0x0015
- static final int KEY_TYPE_NOT_MATCH =0x0300
- static final int KEY_NOT_LOAD =0x0400
- static final int SAME_KEY_ERROR =0x0500
- static final int KEY_ALL_ZERO =0x0501
- static final int TR31_FORMAT_ERROR =0x0502
- static final int PAN_ERROR =0x0702
- static final int KEY_HAS_LOADED =0x0D00
- static final int TIME_HAS_LOADED =0x0E00
- static final int ENCRYPT_DECRYPT_FAILED =0x0F00
- static final int LOW_BATTERY =0x1000
- static final int ACTION_CANCELED =0x1800
- static final int ACTION_ABORTED =0x1900
- static final int ICC_CARD_NOT_SEATED =0x2C02
- static final int NO_SMARTCARD_REQUEST =0x2C06
- static final int CARD_NOT_SUPPORTED =0x2D01
- static final int CARD_NOT_SUPPORTED_WANTS_CRC =0x2D03
- static final int CHIP_IS_DEACTIVATION =0x3000
- static final int CHIP_NOT_CONNECT =0x30FF
- static final int CHIP_IS_ACTIVATION =0x3101
- static final int NOT_ADMIN_KEY =0x5500
- static final int ADMIN_KEY_STOP =0x5501
- static final int ADMIN_KSN_ERROR =0x5502
- static final int GET_AUTHENTICATION_CODE_FAIL =0x5503
- static final int AUTHENTICATION_CODE_ERROR =0x5504
- static final int ENCRYPTED_DECRYPTED_DATA_ERROR =0x5505
- static final int KEY_TYPE_NOT_SUPPORTED =0x5506
- static final int KEY_INDEX_ERROR =0x5507
- static final int STEP_ERROR =0x5508
- static final int KSN_ERROR =0x5509
- static final int MAC_ERROR =0x550A
- static final int KEY_USAGE_ERROR =0x550B
- static final int MODE_ERROR =0x550C
- static final int OTHER_ERROR =0x550F
- static final int LANGUAGE_CONFIG_FAIL =0x6000

- static final int SERIAL_NUMBER_NOT_EXIST =0x6200
- static final int INVALID_TASKID =0x6900
- static final int WITHOUT_ICC_SUPPORTED =0x690D
- static final int UNSUPPORTED_COMMAND =0x6A00
- static final int UNSUPPORTED_COMMAND_IN_STATE =0x6A01
- static final int INVALID_PARAMETER =0x6B00
- static final int INVALID_COMMAND_LENGTH =0x6C00
- static final int DEVICE_SUSPEND =0x7200
- static final int PIN_STOP =0x7300
- static final int DEVICE_BUSY =0x7400
- static final int TIMEOUT =0x8100
- static final int INVALID_TS =0x8200
- static final int NO_CARD_DATA =0x8200
- static final int TRIMAGII_NO_RESPONSE =0x8200
- static final int PPS_CONFIRMATION_ERROR =0x8500
- static final int UNSUPPORTED_FD =0x8600
- static final int PROTOCOL_NOT_SUPPORTED_EMV =0x8700
- static final int POWER_NOT_POWERLEVEL =0x8800
- static final int ATR_LENGTH_EXCEED =0x8900
- static final int EMV_INVALID_TA1 =0x8B01
- static final int EMV_TB1_REQUIRED =0x8B02
- static final int EMV_UNSUPPORTED_TB1 =0x8B03
- static final int EMV_CARD_ERROR =0x8B04
- static final int EMV_TB2_NOT_ALLOWED =0x8B06
- static final int EMV_TC2_EXCEED =0x8B07
- static final int EMV_TC2_OUT_RANGE =0x8B08
- static final int EMV96_TA3_ERROR =0x8B09
- static final int ICC_ERROR_ON_POWERUP =0x8B10
- static final int EMV_TB3_REQUIRED =0x8B11
- static final int INVALID_BWIORCWI =0x8B12
- static final int CARD_ERROR_INVALID_BWIORCWI =0x8B13
- static final int EMV_TCI_TB3_CONFLICT =0x8B17
- static final int EMV_TD2_OUT_RANGE =0x8B20
- static final int TCK_ERROR =0x8C00
- static final int CONNECTOR_NO_VOLTAGE_SETTING =0xA30C
- static final int INVALID_SBLK_IFSD_EXCHANGE =0xA305
- static final int DATA_NOT_EXIST =0xD000
- static final int DATA_ACCESS_ERROR =0xD001
- static final int INDEX_NOT_EXIST =0xD102
- static final int MAXIMUM_EXCEEDED =0xD200
- static final int HASH_ERROR =0xD201
- static final int Other_ERROR =0xD204
- static final int SYSTEM_BUSY =0xD205
- static final int CANNOT_SLEEP_MODE =0xE100
- static final int FILE_HAS_EXISTED =0xE200
- static final int FILE_NOT_EXIST =0xE300
- static final int ICC_ERROR_AFTER_SESSION_START =0xE301
- static final int IO_LINE_LOW =0xE313
- static final int OPEN_FILE_ERROR =0xE400
- static final int SMARTCARD_ERROR =0xE500
- static final int CARD_DATA_ERROR =0xE600
- static final int TIMEOUT_COMMAND =0xE700
- static final int FILE_READ_WRITE_ERROR =0xE800
- static final int ACTIVE_1850_ERROR =0xE900
- static final int BOOTLOADER_ERROR =0xEA00

- static final int PICTURE_NOT_EXIST =0xEB00
- static final int PROCOTOL_ERROR =0xEF00
- static final int ICC_COMMUNICATION_TIMEOUT =0xF002
- static final int ICC_COMMUNICATION_ERROR =0xF003
- static final int ICC_NEED_DISABLE_MSR =0xF00F
- static final int AID_NOT_EXIST =0xF200
- static final int TERMINATE_DATA_NOT_EXIST =0xF201
- static final int TLV_FORMAT_ERROR =0xF202
- static final int AID_LIST_OVERFLOW =0xF203
- static final int CAKEY_NOT_EXIST =0xF204
- static final int CAKEY_RID_NOT_EXIST =0xF205
- static final int CAKEY_INDEX_NOT_EXIST =0xF206
- static final int CAKEY_LIST_OVERFLOW =0xF207
- static final int CAKEY_HASH_ERROR =0xF208
- static final int TRANSACTION_FORMAT_ERROR =0xF209
- static final int COMMAND_NOT_PROCESS =0xF20A
- static final int CRL_NOT_EXIST =0xF20B
- static final int CRL_LIST_OVERFLOW =0xF20C
- static final int MISS_AMOUNT_OTHERAMOUNT_TRANXTYPE =0xF20D
- static final int ID_ALGORITHM_MISTAKE =0xF20E
- static final int NO_FINANCIAL_CARD =0xF20F
- static final int TLV_BUFFER_OVERFLOW =0xF210

12.11.1 Field Documentation

12.11.1.1 final int com.idtechproducts.device.ErrorCode.ACTION_ABORTED =0x1900 [static]

0x1900

12.11.1.2 final int com.idtechproducts.device.ErrorCode.ACTION_CANCELED =0x1800 [static]

0x1800

12.11.1.3 final int com.idtechproducts.device.ErrorCode.ACTIVE_1850_ERROR =0xE900 [static]

0xE900

12.11.1.4 final int com.idtechproducts.device.ErrorCode.ADMIN_KEY_STOP =0x5501 [static]

0x5501

12.11.1.5 final int com.idtechproducts.device.ErrorCode.ADMIN_KSN_ERROR =0x5502 [static]

0x5502

12.11.1.6 final int com.idtechproducts.device.ErrorCode.AID_LIST_OVERFLOW =0xF203 [static]

0xF203

12.11.1.7 final int com.idtechproducts.device.ErrorCode.AID_NOT_EXIST =0xF200 [static]

0xF200

12.11.1.8 final int com.idtechproducts.device.ErrorCode.ATR_LENGTH_EXCEED =0x8900 [static]

0x8900

12.11.1.9 final int com.idtechproducts.device.ErrorCode.AUTHENTICATION_CODE_ERROR =0x5504 [static]

0x5504

12.11.1.10 final int com.idtechproducts.device.ErrorCode.AUTO_CONFIG_MODEL =0x0010 [static]

0x0010

12.11.1.11 final int com.idtechproducts.device.ErrorCode.BOOTLOADER_ERROR =0xEA00 [static]

0xEA00

12.11.1.12 final int com.idtechproducts.device.ErrorCode.CAKEY_HASH_ERROR =0xF208 [static]

0xF208

12.11.1.13 final int com.idtechproducts.device.ErrorCode.CAKEY_INDEX_NOT_EXIST =0xF206 [static]

0xF206

12.11.1.14 final int com.idtechproducts.device.ErrorCode.CAKEY_LIST_OVERFLOW =0xF207 [static]

0xF207

12.11.1.15 final int com.idtechproducts.device.ErrorCode.CAKEY_NOT_EXIST =0xF204 [static]

0xF204

12.11.1.16 final int com.idtechproducts.device.ErrorCode.CAKEY_RID_NOT_EXIST =0xF205 [static]

0xF205

12.11.1.17 final int com.idtechproducts.device.ErrorCode.CANNOT_SLEEP_MODE =0xE100 [static]

0xE100

12.11.1.18 final int com.idtechproducts.device.ErrorCode.CARD_DATA_ERROR =0xE600 [static]

0xE600

12.11.1.19 final int com.idtechproducts.device.ErrorCode.CARD_ERROR_INVALID_BWIORCWI =0x8B13 [static]

0x8B13

12.11.1.20 final int com.idtechproducts.device.ErrorCode.CARD_NOT_SUPPORTED =0x2D01 [static]

0x2D01

12.11.1.21 final int com.idtechproducts.device.ErrorCode.CARD_NOT_SUPPORTED_WANTS_CRC =0x2D03 [static]

0x2D03

12.11.1.22 final int com.idtechproducts.device.ErrorCode.CHIP_IS_ACTIVATION =0x3101 [static]

0x3101

12.11.1.23 final int com.idtechproducts.device.ErrorCode.CHIP_IS_DEACTIVATION =0x3000 [static]

0x3000

12.11.1.24 final int com.idtechproducts.device.ErrorCode.CHIP_NOT_CONNECT =0x30FF [static]

0x30FF

12.11.1.25 final int com.idtechproducts.device.ErrorCode.COMMAND_NOT_PROCESS =0xF20A [static]

0xF20A

12.11.1.26 final int com.idtechproducts.device.ErrorCode.CONNECTOR_NO_VOLTAGE_SETTING =0xA30C [static]

0xA304

12.11.1.27 final int com.idtechproducts.device.ErrorCode.CRL_LIST_OVERFLOW =0xF20C [static]

0xF20C

12.11.1.28 final int com.idtechproducts.device.ErrorCode.CRL_NOT_EXIST =0xF20B [static]

0xF20B

12.11.1.29 final int com.idtechproducts.device.ErrorCode.DATA_ACCESS_ERROR =0xD001 [static]

0xD001

12.11.1.30 final int com.idtechproducts.device.ErrorCode.DATA_NOT_EXIST =0xD000 [static]

0xD000

12.11.1.31 final int com.idtechproducts.device.ErrorCode.DEVICE_BUSY =0x7400 [static]

0x7400

12.11.1.32 final int com.idtechproducts.device.ErrorCode.DEVICE_SUSPEND =0x7200 [static]

0x7200

12.11.1.33 final int com.idtechproducts.device.ErrorCode.EMV96_TA3_ERROR =0x8B09 [static]

0x8B09

12.11.1.34 final int com.idtechproducts.device.ErrorCode.EMV_CARD_ERROR =0x8B04 [static]

0x8B04

12.11.1.35 final int com.idtechproducts.device.ErrorCode.EMV_INVALID_TA1 =0x8B01 [static]

0x8B01

12.11.1.36 final int com.idtechproducts.device.ErrorCode.EMV_TB1_REQUIRED =0x8B02 [static]

0x8B02

12.11.1.37 final int com.idtechproducts.device.ErrorCode.EMV_TB2_NOT_ALLOWED =0x8B06 [static]

0x8B06

12.11.1.38 final int com.idtechproducts.device.ErrorCode.EMV_TB3_REQUIRED =0x8B11 [static]

0x8B11

12.11.1.39 final int com.idtechproducts.device.ErrorCode.EMV_TC2_EXCEED =0x8B07 [static]

0x8B07

12.11.1.40 final int com.idtechproducts.device.ErrorCode.EMV_TC2_OUT_RANGE =0x8B08 [static]

0x8B08

12.11.1.41 final int com.idtechproducts.device.ErrorCode.EMV_TCI_TB3_CONFLICT =0x8B17 [static]

0x8B17

12.11.1.42 final int com.idtechproducts.device.ErrorCode.EMV_TD2_OUT_RANGE =0x8B20 [static]

0x8B20

12.11.1.43 final int com.idtechproducts.device.ErrorCode.EMV_UNSUPPORTED_TB1 =0x8B03 [static]

0x8B03

12.11.1.44 final int com.idtechproducts.device.ErrorCode.EMVL2_MODEL =0x0007 [static]

0x0007

12.11.1.45 final int com.idtechproducts.device.ErrorCode.ENCRYPT_DECRYPT_FAILED =0x0F00 [static]

0x0F00

12.11.1.46 final int com.idtechproducts.device.ErrorCode.ENCRYPTED_DECRYPTED_DATA_ERROR =0x5505 [static]

0x5505

12.11.1.47 final int com.idtechproducts.device.ErrorCode.FILE_HAS_EXISTED =0xE200 [static]

0xE200

12.11.1.48 final int com.idtechproducts.device.ErrorCode.FILE_NOT_EXIST =0xE300 [static]

0xE300

12.11.1.49 final int com.idtechproducts.device.ErrorCode.FILE_READ_WRITE_ERROR =0xE800 [static]

0xE800

12.11.1.50 final int com.idtechproducts.device.ErrorCode.GET_AUTHENTICATION_CODE_FAIL =0x5503 [static]

0x5503

12.11.1.51 final int com.idtechproducts.device.ErrorCode.HASH_ERROR =0xD201 [static]

0xD201

12.11.1.52 final int com.idtechproducts.device.ErrorCode.ICC_CARD_NOT_SEATED =0x2C02 [static]

0x2C02

12.11.1.53 final int com.idtechproducts.device.ErrorCode.ICC_COMMUNICATION_ERROR =0xF003 [static]

0xF003

12.11.1.54 final int com.idtechproducts.device.ErrorCode.ICC_COMMUNICATION_TIMEOUT =0xF002 [static]

0xF002

12.11.1.55 final int com.idtechproducts.device.ErrorCode.ICC_ERROR_AFTER_SESSION_START =0xE301 [static]

0xE301

12.11.1.56 final int com.idtechproducts.device.ErrorCode.ICC_ERROR_ON_POWERUP =0x8B10 [static]

0x8B10

12.11.1.57 final int com.idtechproducts.device.ErrorCode.ICC_NEED_DISABLE_MSR =0xF00F [static]

0xF00F

12.11.1.58 final int com.idtechproducts.device.ErrorCode.ID_ALGORITHM_MISTAKE =0xF20E [static]

0xF20E

12.11.1.59 final int com.idtechproducts.device.ErrorCode.INDEX_NOT_EXIST =0xD102 [static]

0xD102

12.11.1.60 final int com.idtechproducts.device.ErrorCode.INVALID_BWIORCWI =0x8B12 [static]

0x8B12

12.11.1.61 final int com.idtechproducts.device.ErrorCode.INVALID_COMMAND_LENGTH =0x6C00 [static]

0x6C00

12.11.1.62 final int com.idtechproducts.device.ErrorCode.INVALID_PARAMETER =0x6B00 [static]

0x6B00

12.11.1.63 final int com.idtechproducts.device.ErrorCode.INVALID_REPONSE_DATA =0x0002 [static]

0x0002

12.11.1.64 final int com.idtechproducts.device.ErrorCode.INVALID_SBLK_IFSD_EXCHANGE =0xA305 [static]

0xA305

12.11.1.65 final int com.idtechproducts.device.ErrorCode.INVALID_TASKID =0x6900 [static]

0x6900

12.11.1.66 final int com.idtechproducts.device.ErrorCode.INVALID_TS =0x8200 [static]

0x8200

12.11.1.67 final int com.idtechproducts.device.ErrorCode.IO_LINE_LOW =0xE313 [static]

0xE313

12.11.1.68 final int com.idtechproducts.device.ErrorCode.KEY_ALL_ZERO =0x0501 [static]

0x0501

12.11.1.69 final int com.idtechproducts.device.ErrorCode.KEY_HAS_LOADED =0x0D00 [static]

0x0D00

12.11.1.70 final int com.idtechproducts.device.ErrorCode.KEY_INDEX_ERROR =0x5507 [static]

0x5507

12.11.1.71 final int com.idtechproducts.device.ErrorCode.KEY_NOT_LOAD =0x0400 [static]

0x0400

12.11.1.72 final int com.idtechproducts.device.ErrorCode.KEY_TYPE_NOT_MATCH =0x0300 [static]

Reader Status code 0x0300

12.11.1.73 final int com.idtechproducts.device.ErrorCode.KEY_TYPE_NOT_SUPPORTED =0x5506 [static]

0x5506

12.11.1.74 final int com.idtechproducts.device.ErrorCode.KEY_USAGE_ERROR =0x550B [static]

0x550B

12.11.1.75 final int com.idtechproducts.device.ErrorCode.KSN_ERROR =0x5509 [static]

0x5509

12.11.1.76 final int com.idtechproducts.device.ErrorCode.LANGUAGE_CONFIG_FAIL =0x6000 [static]

0x6000

12.11.1.77 final int com.idtechproducts.device.ErrorCode.LOW_BATTERY =0x1000 [static]

0x1000

12.11.1.78 final int com.idtechproducts.device.ErrorCode.MAC_ERROR =0x550A [static]

0x550A

12.11.1.79 final int com.idtechproducts.device.ErrorCode.MAXIMUM_EXCEEDED =0xD200 [static]

0xD200

12.11.1.80 final int com.idtechproducts.device.ErrorCode.MISS_AMOUNT_OTHERAMOUNT_TRANXTYPE =0xF20D [static]

0xF20D

12.11.1.81 final int com.idtechproducts.device.ErrorCode.MODE_ERROR =0x550C [static]

0x550C

12.11.1.82 final int com.idtechproducts.device.ErrorCode.MSR_MODEL =0x0005 [static]

0x0005

12.11.1.83 final int com.idtechproducts.device.ErrorCode.NO_CARD_DATA =0x8200 [static]

0x8300

12.11.1.84 final int com.idtechproducts.device.ErrorCode.NO_CONFIG =0x0011 [static]

0x0011

12.11.1.85 final int com.idtechproducts.device.ErrorCode.NO_FINANCIAL_CARD =0xF20F [static]

0xF20F

12.11.1.86 final int com.idtechproducts.device.ErrorCode.NO_READER =0x0012 [static]

0x0012

12.11.1.87 final int com.idtechproducts.device.ErrorCode.NO_RESPONSE =0x0015 [static]

0x0015

12.11.1.88 final int com.idtechproducts.device.ErrorCode.NO_SMARTCARD_REQUEST =0x2C06 [static]

0x2C06

12.11.1.89 final int com.idtechproducts.device.ErrorCode.NOT_ADMIN_KEY =0x5500 [static]

0x5500

12.11.1.90 final int com.idtechproducts.device.ErrorCode.NOT_CONNECT =0x0001 [static]

0x0001

12.11.1.91 final int com.idtechproducts.device.ErrorCode.NOT_SUPPORTED =0x0014 [static]

0x0014

12.11.1.92 final int com.idtechproducts.device.ErrorCode.NOTIFYRESPONSE_MODEL =0x0008 [static]

0x0008

12.11.1.93 final int com.idtechproducts.device.ErrorCode.OPEN_FILE_ERROR =0xE400 [static]

0xE400

12.11.1.94 final int com.idtechproducts.device.ErrorCode.OTHER_ERROR =0x550F [static]

0x550F

12.11.1.95 final int com.idtechproducts.device.ErrorCode.Other_ERROR =0xD204 [static]

0xD204

12.11.1.96 final int com.idtechproducts.device.ErrorCode.PAN_ERROR =0x0702 [static]

0x0702

12.11.1.97 final int com.idtechproducts.device.ErrorCode.PICTURE_NOT_EXIST =0xEB00 [static]

0xEB00

12.11.1.98 final int com.idtechproducts.device.ErrorCode.PIN_MODEL =0x0006 [static]

0x0006

12.11.1.99 final int com.idtechproducts.device.ErrorCode.PIN_STOP =0x7300 [static]

0x7300

12.11.1.100 final int com.idtechproducts.device.ErrorCode.POWER_NOT_POWERLEVEL =0x8800 [static]

0x8800

12.11.1.101 final int com.idtechproducts.device.ErrorCode.PPS_CONFIRMATION_ERROR =0x8500 [static]

0x8500

12.11.1.102 final int com.idtechproducts.device.ErrorCode.PROCOTOL_ERROR =0xEF00 [static]

0xEF00

12.11.1.103 final int com.idtechproducts.device.ErrorCode.PROTOCOL_NOT_SUPPORTED_EMV =0x8700 [static]

0x8700

12.11.1.104 final int com.idtechproducts.device.ErrorCode.SAME_KEY_ERROR =0x0500 [static]

0x0500

12.11.1.105 final int com.idtechproducts.device.ErrorCode.SDK_BUZY =0x0013 [static]

0x0013

12.11.1.106 final int com.idtechproducts.device.ErrorCode.SERIAL_NUMBER_NOT_EXIST =0x6200 [static]

0x6200

12.11.1.107 final int com.idtechproducts.device.ErrorCode.SMARTCARD_ERROR =0xE500 [static]

0xE500

12.11.1.108 final int com.idtechproducts.device.ErrorCode.STEP_ERROR =0x5508 [static]

0x5508

12.11.1.109 final int com.idtechproducts.device.ErrorCode.SUCCESS =0x0000 [static]

SDK Status code The code description can be retrieved from the API device_getResponseCodeString. 0x0000

12.11.1.110 final int com.idtechproducts.device.ErrorCode.SYSTEM_BUSY =0xD205 [static]

0xD205

12.11.1.111 final int com.idtechproducts.device.ErrorCode.TCK_ERROR =0x8C00 [static]

0x8C00

12.11.1.112 final int com.idtechproducts.device.ErrorCode.TERMINATE_DATA_NOT_EXIST =0xF201 [static]

0xF201

12.11.1.113 final int com.idtechproducts.device.ErrorCode.TIME_HAS_LOADED =0x0E00 [static]

0x0E00

12.11.1.114 final int com.idtechproducts.device.ErrorCode.TIMEOUT =0x8100 [static]

0x8100

12.11.1.115 final int com.idtechproducts.device.ErrorCode.TIMEOUT_COMMAND =0xE700 [static]

0xE700

12.11.1.116 final int com.idtechproducts.device.ErrorCode.TIMEOUT_TASK =0x0003 [static]

0x0003

12.11.1.117 final int com.idtechproducts.device.ErrorCode.TLV_BUFFER_OVERFLOW =0xF210 [static]

0xF210

12.11.1.118 final int com.idtechproducts.device.ErrorCode.TLV_FORMAT_ERROR =0xF202 [static]

0xF202

12.11.1.119 final int com.idtechproducts.device.ErrorCode.TR31_FORMAT_ERROR =0x0502 [static]

0x0502

12.11.1.120 final int com.idtechproducts.device.ErrorCode.TRANSACTION_FORMAT_ERROR =0xF209 [static]

0xF209

12.11.1.121 final int com.idtechproducts.device.ErrorCode.TRIMAGII_NO_RESPONSE =0x8200 [static]

0x8400

12.11.1.122 final int com.idtechproducts.device.ErrorCode.UNKNOW_ERROR =0x0009 [static]

0x0009

12.11.1.123 final int com.idtechproducts.device.ErrorCode.UNSUPPORTED_COMMAND =0x6A00 [static]

0x6A00

12.11.1.124 final int com.idtechproducts.device.ErrorCode.UNSUPPORTED_COMMAND_IN_STATE =0x6A01 [static]

0x6A01

12.11.1.125 final int com.idtechproducts.device.ErrorCode.UNSUPPORTED_FD =0x8600 [static]

0x8600

12.11.1.126 final int com.idtechproducts.device.ErrorCode.WITHOUT_ICC_SUPPORTED =0x690D [static]

0x690D

12.11.1.127 final int com.idtechproducts.device.ErrorCode.WRONG_PARAMETER =0x0004 [static]

0x0004

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ErrorCode.java

12.12 com.idtechproducts.emv.UniPayEMV.EVENT_MSR_Types Enum Reference

Public Member Functions

- String **toString** ()

Data Fields

- **EVENT_MSR_UNKNOWN** =(31)
- **EVENT_MSR_CARD_DATA** =(32)
- **EVENT_MSR_CANCEL_KEY** =(33)
- **EVENT_MSR_BACKSPACE_KEY** =(34)
- **EVENT_MSR_ENTER_KEY** =(35)
- **EVENT_MSR_DATA_ERROR** =(36)
- **EVENT_MSR_ICC_START** =(37)
- **EVENT_BTPAY_CARD_DATA** =(38)
- **EVENT_UNIPAYII_EMV_NO_ICC_MSR_DATA** =(39)
- **EVENT_UNIPAYII_EMV_FALLBACK_DATA** =(40)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.13 com.idtechproducts.device.ReaderInfo.EVENT_MSR_Types Enum Reference

Data Fields

- **EVENT_MSR_UNKNOWN**
- **EVENT_MSR_CARD_DATA**
- **EVENT_MSR_CANCEL_KEY**
- **EVENT_MSR_BACKSPACE_KEY**
- **EVENT_MSR_ENTER_KEY**
- **EVENT_MSR_DATA_ERROR**

12.13.1 Detailed Description

Define MSR type.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.14 com.idtechproducts.device.ICCReaderStatusStruct Class Reference

Data Fields

- boolean iccPower
- boolean cardSeated
- boolean latchClosed
- boolean cardPresent
- boolean magneticDataPresent

12.14.1 Detailed Description

The class for ICC Reader Status.

12.14.2 Field Documentation

12.14.2.1 boolean com.idtechproducts.device.ICCReaderStatusStruct.cardPresent

If device has a latch, determines if the card is present in device.If the device does not have a latch,value is always FALSE.

12.14.2.2 boolean com.idtechproducts.device.ICCReaderStatusStruct.cardSeated

Determines if card is inserted.

12.14.2.3 boolean com.idtechproducts.device.ICCReaderStatusStruct.iccPower

Determines if ICC has been powered up.

12.14.2.4 boolean com.idtechproducts.device.ICCReaderStatusStruct.latchClosed

Determines if Card Latch is engaged.If device does not have a latch,value is always FALSE.

12.14.2.5 boolean com.idtechproducts.device.ICCReaderStatusStruct.magneticDataPresent

True = Magnetic data present,False = No Magnetic Data.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ICCReaderStatusStruct.java

12.15 com.idtechproducts.device.ICCSettingStruct Class Reference

Data Fields

- byte keyTypeOption
- byte encryptionModeOption
- byte iccVoltageOption
- byte mainCardTypeOption
- byte prePANID
- byte postPANID
- byte maskCharWithASCII
- byte maskCharWithBCD
- byte iccWithNotiticationOption
- byte ConnectorOption
- byte TransactionTimeoutL1

12.15.1 Detailed Description

Response data for all settings of ICC status.

NOTE:

From firmware version 1.01.029, support following setting item:
keyTypeOption
encryptionModeOption
mainCardTypeOption
iccWithNotiticationOption
TransactionTimeoutL1

Before firmware version 1.01.029, support following setting item:
keyTypeOption
encryptionModeOption
mainCardTypeOption
iccWithNotiticationOption

12.15.2 Field Documentation

12.15.2.1 byte com.idtechproducts.device.ICCSettingStruct.ConnectorOption

ICC Connector Option,one byte,default value:0x00; (0x00: User Card Connector,0x01: SAM Connector).

12.15.2.2 byte com.idtechproducts.device.ICCSettingStruct.encryptionModeOption

Encryption Mode Option:

NOTE: It is for UniPay option

12.15.2.3 byte com.idtechproducts.device.ICCSettingStruct.iccVoltageOption

ICC Voltage Option: All Voltages (3V/5V) 0xFF, Only 5V 0x00. NOTE: All EMV cards support 5V voltage. So default power could be Only 5V NOTE: It is for UniPay II option

12.15.2.4 byte com.idtechproducts.device.ICCSettingStruct.iccWithNotiticationOption

ICC Notification status option, one byte,default value:0x01 (0x01: open the ICC notification, 0xFF: close the ICC notification).

12.15.2.5 byte com.idtechproducts.device.ICCSettingStruct.keyTypeOption

Key Type Option:

NOTE: It is for UniPay option

12.15.2.6 byte com.idtechproducts.device.ICCSettingStruct.mainCardTypeOption

Main Card Type Option: EMV Card 0xFF, ISO Card 0x00 NOTE: It is for UniPay II option

12.15.2.7 byte com.idtechproducts.device.ICCSettingStruct.maskCharWithASCII

Character used to mask PAN with ASCII,one byte,default value:0x2A(0x20∼0x7A). NOTE: It is for UniPay II option

12.15.2.8 byte com.idtechproducts.device.ICCSettingStruct.maskCharWithBCD

Character used to mask PAN with BCD,one byte,default value:0x0C(0x0A∼0x0F). NOTE: It is for UniPay II option

12.15.2.9 byte com.idtechproducts.device.ICCSettingStruct.postPANID

Last PAN digits to display,one byte,default value:0x04(0x00∼0x04). NOTE: It is for UniPay II option

12.15.2.10 byte com.idtechproducts.device.ICCSettingStruct.prePANID

Leading PAN digits to display,one byte, default value:0x04(0x00∼0x06). NOTE: It is for UniPay II option

12.15.2.11 byte com.idtechproducts.device.ICCSettingStruct.TransactionTimeoutL1

L1 Transaction Timeout, the value is the time to power off the reader from power on.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ICCSettingStruct.java

12.16 com.idtechproducts.device.IDT_Device Class Reference

Public Member Functions

- IDT_Device (OnReceiverListener callback, Context contex)
- void registerListen ()
- void unregisterListen ()
- void release ()
- String config_getXMLVersionInfo ()
- String phone_getInfoManufacture ()
- String phone_getInfoModel ()
- void log_setVerboseLoggingEnable (boolean enableShowLog)
- void log_setSaveLogEnable (boolean enable)
- int log_deleteLogs ()
- void config_setXMLFileNameWithPath (String xmlFilename)
- boolean config_loadingConfigurationXMLFile (boolean updateAutomatically)
- boolean device_connect ()
- void device_disconnect ()
- boolean device_isConnected ()
- boolean device_connectWithProfile (StructConfigParameters profile)
- boolean msr_isSwipeCardRunning ()
- int config_getSDKVersion (StringBuilder sdkVersion)
- int config_getModelNumber (StringBuilder modNumber)
- int config_getSerialNumber (StringBuilder serialNumber)
- int config_setSerialNumber (String serNumber)
- int config_getDateTime (StringBuilder dateTime)
- int config_setDateTime (String dateTime)
- int config_getInterfaceDeviceSN (StringBuilder serialNumber)

- int config_setInterfaceDeviceSN (String serNumber)
- int config_getTerminalIdentification (StringBuilder termIden)
- int config_setTerminalIdentification (String termIden)
- int device_getFirmwareVersion (StringBuilder version)
- int device_getBatteryVoltage (StringBuilder batteryInfo)
- int device_ReviewAudioJackSetting (int response[])
- int device_rebootDevice ()
- int device_enableBeep (boolean enable)
- int device_sendBeep (int frequency, int duration)
- int device_sendCommandDirectIO (String hexCommand, ResDataStruct respData)
- String device_getResponseCodeString (int errorCode)
- int icc_getICCReaderStatus (ICCReaderStatusStruct ICCStatus)
- int icc_powerOnICC (PowerOnStructure options, ResDataStruct atrPPS)
- int icc_powerOffICC ()
- int icc_exchangeAPDU (byte[] dataAPDU, APDUResponseStruct response)
- int icc_exchangeEncryptedAPDU (byte[] dataAPDU, byte[] ksn, APDUResponseStruct response)
- int icc_exchangeMultiAPDU (byte[] dataAPDU, ResDataStruct respData)
- int icc_getCurrentKSN (ResDataStruct resKSN)
- int icc_Set3VThen5V ()
- int icc_Set5V ()
- int icc_enableNotification (boolean enableNotifyICCStatus)
- int icc_setICCConnector (byte connector)
- int icc_setProAndPostPanLength (int prePanCtrlDataLen, int postPanCtrlDataLen)
- int icc_setAsciiMask (byte maskWithAscii)
- int icc_setBCDMask (byte maskWithBCD)
- int icc_defaultSetting ()
- int icc_reviewAllSetting (ICCSettingStruct iccSetting)
- int lcd_clearDisplay (byte control)
- int lcd_savePromptDisplay (int numPrompt, String message)
- int lcd_displayPrompt (int numLine, int numPrompt)
- int lcd_displayMessage (int numLine, String message)
- int lcd_enableSleepMode (boolean sleepMode)
- int lcd_defaultSetting ()
- int lcd_getSetting (boolean[] sleepMode)
- int msr_reviewAllSetting (MSRSettingStruct msrSetting)
- int msr_defaultAllSetting ()
- int msr_getSingleSetting (byte funcID, byte[] response)
- int msr_setSingleSetting (byte funcID, byte setData)
- int msr_startMSRSwipe ()
- int msr_cancelMSRSwipe ()
- int autoConfig_start (String strXMLFilename, boolean fastSearch)
- void autoConfig_stop ()
- void stopWaitingTask ()
- boolean setWaitingMSRResponseTimeout (int timeoutValue)
- boolean setWaitingExchangeAPDUResponseTimeout (int timeoutValue)
- boolean setWaitingPINResponseTimeout (int timeoutValue)
- void setEMVListener (OnReceiverListener callback)
- ReaderType getAttachedReaderType ()
- SupportStatus getSupportStatus (ReaderType readerType)
- SupportStatus getSupportStatus ()

Data Fields

- UMLog **LOG**

12.16.1 Detailed Description

The IDTech Framework is an Android Framework that will be provided by IDTech as the main interface between Android applications and UniPay reader processing solutions. The purpose of this document is to describe the requirements of the frameworks as well as the interface definitions and requirements needed for any Android applications wishing to deploy with the payment application.

12.16.2 Constructor & Destructor Documentation

12.16.2.1 com.idtechproducts.device.IDT_Device.IDT_Device (OnReceiverListener *callback,* Context *contex*) [inline]

It is the constructor of the main class IDT_Device. When it is called, the SDK will create the Instance for UniPay reader device. The interface UniPayReaderMsg as the callback function need be implemented in the application.

It's better to call the constructor When android activity is created.

12.16.3 Member Function Documentation

12.16.3.1 int com.idtechproducts.device.IDT_Device.autoConfig_start (String *strXMLFilename,* boolean *fastSearch*) [inline]

start Auto Config to search the profile.

Parameters

strXMLFilename	Input the customized XML file as the templates to search the profile.
fastSearch	There are two steps to search the profile: First to search in the templates inside of XML file, Second is to search with emumerate posible templates by experiense. If fastSearch is true, the search will skip the second step. If fastSearch is false, the search will cover the two steps.

Returns

success or error code.

See also

ErrorCode

12.16.3.2 void com.idtechproducts.device.IDT_Device.autoConfig_stop ( ) [inline]

stop Auto Config.

Returns

null.

12.16.3.3 int com.idtechproducts.device.IDT_Device.config_getDateTime (StringBuilder *dateTime*) [inline]

Get the date and time of device.

Parameters

dateTime	the device's date and time, 20121123122459.

Returns

success or error code.

See also

ErrorCode

12.16.3.4 int com.idtechproducts.device.IDT_Device.config_getInterfaceDeviceSN (StringBuilder *serialNumber*) [inline]

Get interface device's serial number.

Parameters

serialNumber	for interface device's serial number string.

Returns

success or error code.

See also

ErrorCode

12.16.3.5 int com.idtechproducts.device.IDT_Device.config_getModelNumber (StringBuilder *modNumber*) [inline]

Get the model number of device.

Parameters

modNumber	for Model Number.

Returns

success or error code.

See also

ErrorCode

12.16.3.6 int com.idtechproducts.device.IDT_Device.config_getSDKVersion (StringBuilder *sdkVersion*) [inline]

READER CONFIG API LIST Get the version of SDK.

Parameters

sdkVersion	for version string.

Returns

success or error code.

See also

ErrorCode

12.16.3.7 int com.idtechproducts.device.IDT_Device.config_getSerialNumber (StringBuilder *serialNumber*) [inline]

Get the serial number of device.

Parameters

serialNumber	for Serial Number.

Returns

success or error code.

See also

ErrorCode

12.16.3.8 int com.idtechproducts.device.IDT_Device.config_getTerminalIdentification (StringBuilder *termIden*) [inline]

Get Get terminal identification.

Parameters

termIden	for device terminal identification string.

Returns

success or error code.

See also

ErrorCode

12.16.3.9 String com.idtechproducts.device.IDT_Device.config_getXMLVersionInfo ( ) [inline]

Get XML configuration version.

Returns

the version info.

12.16.3.10 boolean com.idtechproducts.device.IDT_Device.config_loadingConfigurationXMLFile (boolean *updateAutomatically*) [inline]

Load XML Configuration File.

Parameters

xmlFilename,XML	Configuration File Name.

Returns

none

12.16.3.11 int com.idtechproducts.device.IDT_Device.config_setDateTime (String *dateTime*) [inline]

Set the date and time to device.

Parameters

dateTime	the date and time in hex string,12 characters.If date is: 2012/11/23 12:24:59, dateTime should be 121123122459.

Returns

success or error code.

See also

ErrorCode

12.16.3.12 int com.idtechproducts.device.IDT_Device.config_setInterfaceDeviceSN (String *serNumber*) [inline]

Set Interface device's serial number. EMV serial Number can be set only once.

Parameters

serNumber	for Serial Number string;8 bytes '0' ∼ '9', or 'a' ∼ 'z', or 'A' ∼ 'Z'.

Returns

success or error code.

See also

ErrorCode

12.16.3.13 int com.idtechproducts.device.IDT_Device.config_setSerialNumber (String *serNumber*) [inline]

Set the serial number to device.

Parameters

serNumber	for Serial Number string; 9 (UniPay) or 10 (UniPay II) bytes ASCII, from 0x20 to 0x7F.

Returns

success or error code.

See also

ErrorCode

12.16.3.14 int com.idtechproducts.device.IDT_Device.config_setTerminalIdentification (String *termIden*) [inline]

Set terminal identification.

Parameters

termIden	for terminal identification string;8 bytes '0' ∼ '9', or 'a' ∼ 'z', or 'A' ∼ 'Z'.

Returns

success or error code.

See also

ErrorCode

12.16.3.15 void com.idtechproducts.device.IDT_Device.config_setXMLFileNameWithPath (String *xmlFilename*) [inline]

set XML Configuration File Name with the full path.

Parameters

xmlFilename,X↩ ML	Configuration File Name.

Returns

none

12.16.3.16 boolean com.idtechproducts.device.IDT_Device.device_connect ( ) [inline]

Connect the device.

Returns

success or error code.

12.16.3.17 boolean com.idtechproducts.device.IDT_Device.device_connectWithProfile (StructConfigParameters *profile*) [inline]

connect the device with Profile.

Parameters

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

Returns

none

12.16.3.18 void com.idtechproducts.device.IDT_Device.device_disconnect ( ) [inline]

Disconnect the device.

Returns

none

12.16.3.19 int com.idtechproducts.device.IDT_Device.device_enableBeep (boolean *enable*) [inline]

Control the device's beeper.

Parameters

enable	true: enable beep, false: disable to beep.

Returns

success or error code.

See also

ErrorCode

12.16.3.20 int com.idtechproducts.device.IDT_Device.device_getBatteryVoltage (StringBuilder *batteryInfo*) [inline]

DEVICE INFO API Get the Battery Voltage information of device.

Parameters

batteryInfo	for battery Voltage Info string.

Returns

success or error code.

See also

ErrorCode

12.16.3.21 int com.idtechproducts.device.IDT_Device.device_getFirmwareVersion (StringBuilder *version*) [inline]

DEVICE INFO API Get the firmware version of device.

Parameters

version	for version string.

Returns

success or error code.

See also

ErrorCode

12.16.3.22 String com.idtechproducts.device.IDT_Device.device_getResponseCodeString (int *errorCode*) [inline]

Get Response Code String.

Parameters

errorCode	Error code returns by other APIs.

Returns

the string for error code.

12.16.3.23 boolean com.idtechproducts.device.IDT_Device.device_isConnected ( ) [inline]

get the status if the device connected.

Returns

true: connected, false: disconnected

12.16.3.24 int com.idtechproducts.device.IDT_Device.device_rebootDevice ( ) [inline]

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

Returns

success or error code.

See also

ErrorCode

12.16.3.25 int com.idtechproducts.device.IDT_Device.device_ReviewAudioJackSetting (int *response[]*) [inline]

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

12.16.3.26 int com.idtechproducts.device.IDT_Device.device_sendBeep (int *frequency,* int *duration*) [inline]

Control the device's beeper.

Parameters

frequency	the Frequency of beeper;the range from 1000 to 20000.
duration	the duration of beeper; the range from 16 to 65536.

Returns

success or error code.

See also

ErrorCode

12.16.3.27 int com.idtechproducts.device.IDT_Device.device_sendCommandDirectIO (String *hexCommand,* ResDataStruct *respData*) [inline]

Send the command body directly.

Parameters

hexCommand	the command body to send to the device. For example, get the firmware version command: "724601"
response	the response from the device.

Usage about this API, for example to get the firmware version:

byte[] response = null;
device_sendCommandDirectIO("724601",response);

Returns

success or error code.

See also

ErrorCode

12.16.3.28 ReaderType com.idtechproducts.device.IDT_Device.getAttachedReaderType ( ) [inline]

Return the type of reader currently attached and powered up.

Returns UNKNOWN If reader is not attached or not powered up, or if the reader type cannot be determined. Returns the reader type.

12.16.3.29 SupportStatus com.idtechproducts.device.IDT_Device.getSupportStatus (ReaderType *readerType*) [inline]

Return return the reader support status as specified in the configuration readerType: UNIPAYII file the SDK loaded. Returns the support status.

12.16.3.30 SupportStatus com.idtechproducts.device.IDT_Device.getSupportStatus ( ) [inline]

Return return the reader support status as specified in the configuration file the SDK loaded. Returns the support status.

12.16.3.31 int com.idtechproducts.device.IDT_Device.icc_defaultSetting ( ) [inline]

Default ICC group all setting.

Returns

success or error code.

See also

ErrorCode

12.16.3.32 int com.idtechproducts.device.IDT_Device.icc_enableNotification (boolean *enableNotifyICCStatus*) [inline]

Enable / Disable the ICC notification. When enable the notification, the SDK will always post the event when ICC card plug in or out.

Parameters

enableNotifyICCStatus,true	enable the ICC status notification; false:disable the ICC status notification.

Returns

success or error code.

See also

ErrorCode

12.16.3.33 int com.idtechproducts.device.IDT_Device.icc_exchangeAPDU (byte[] *dataAPDU,* APDUResponseStruct *response*) [inline]

Exchange APDU between the currently selected microprocessor card in the ICC reader. The max length of APDU is 512 bytes and the C-APDU is Plaintext.

Parameters

dataAPDU	Plaintext APDU data buffer.
response	the class for response APDU,please see class APDUResponseStruct for more information. The format of Raw response APDU Data Structure of [n bytes Encrypted R-APDU without Status Bytes] is below: 2 bytes Valid R-APDU Length + x bytes Valid R-APDU without Status Bytes + y bytes Padding.

See also

APDUResponseStruct

Returns

success or error code.

See also

ErrorCode

12.16.3.34 int com.idtechproducts.device.IDT_Device.icc_exchangeEncryptedAPDU (byte[] *dataAPDU,* byte[] *ksn,* APDUResponseStruct *response*) [inline]

Exchange APDU between the currently selected microprocessor card in the ICC reader.
The max length of APDU is 512 bytes and the C-APDU is Encrypted.

Parameters

dataAPDU	Encrypted APDU data buffer. The format of Raw C-APDU Data Structure of [m bytes Encrypted C-APDU] is below: m = 2 bytes Valid C-APDU Length + x bytes Valid C-APDU + y bytes Padding (0x00).
ksn	10 bytes Key Serial Number.
response	the class for response APDU,please see class APDUResponseStruct for more information. The format of Raw response APDU Data Structure of [n bytes Encrypted R-APDU] is below:. 2 bytes Valid R-APDU Length + x bytes Valid R-APDU + y bytes Padding.

See also

APDUResponseStruct

Returns

success or error code.

See also

ErrorCode

12.16.3.35 int com.idtechproducts.device.IDT_Device.icc_exchangeMultiAPDU (byte[] *dataAPDU,* ResDataStruct *respData*) [inline]

Exchange Multiple APDUs between the currently selected microprocessor card in the ICC reader.
This API is for UniPay device only.

Parameters

dataAPDU	Plaintext APDU data buffer.
respData	the data buffer respond from the device RPDU Packet format as follows: [Number of APDU commands] [R-APDU 1 length] [R-APDU 1 Command] ' , [R-APDU n length] [R-APDU n command]

See also

ResDataStruct

Returns

success or error code.

See also

ErrorCode

12.16.3.36 int com.idtechproducts.device.IDT_Device.icc_getCurrentKSN (ResDataStruct *resKSN*) [inline]

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

resKSN	the class for current KSN.

Returns

success or error code.

See also

ErrorCode

12.16.3.37 int com.idtechproducts.device.IDT_Device.icc_getICCReaderStatus (ICCReaderStatusStruct *ICCStatus*) [inline]

Get ICC Reader microprocessor card status.

Parameters

ICCStatus	the class for the microprocessor card status.please see ICCReaderStatusStruct class for more information.

See also

ICCReaderStatusStruct

Returns

success or error code.

See also

ErrorCode

12.16.3.38 int com.idtechproducts.device.IDT_Device.icc_powerOffICC ( ) [inline]

Power down the currently selected microprocessor card in the ICC reader.

Returns

success or error code.

See also

ErrorCode

12.16.3.39 int com.idtechproducts.device.IDT_Device.icc_powerOnICC (PowerOnStructure *options*, ResDataStruct *atrPPS*) [inline]

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

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; 23. 8B17: EMV TC1/TB3 conflict∗;

Returns

success or error code.

See also

ErrorCode

12.16.3.40 int com.idtechproducts.device.IDT_Device.icc_reviewAllSetting (ICCSettingStruct *iccSetting*) [inline]

Review all setting of ICC status.

Parameters

iccSetting	for ICC setting. please see class ICCSettingStruct for more information.

See also

ICCSettingStruct

Returns

success or error code.

See also

ErrorCode

12.16.3.41 int com.idtechproducts.device.IDT_Device.icc_Set3VThen5V ( ) [inline]

Internal command Set ICC Voltage option, icc works on 3V then 5V

Returns

success or error code.

See also

ErrorCode

12.16.3.42 int com.idtechproducts.device.IDT_Device.icc_Set5V ( ) [inline]

Internal command Set ICC Voltage option, icc works only on 5V

Returns

success or error code.

See also

ErrorCode

12.16.3.43 int com.idtechproducts.device.IDT_Device.icc_setAsciiMask (byte *maskWithAscii*) [inline]

Set ASCII Mask Data. This setting will be valid in the display of TAG 57/5A at the EMV Level 2 transaction processing.

Parameters

maskWithAscii,can	be from 0x20 to 0x7E, default is 0x2A(∗)

Returns

success or error code.

See also

ErrorCode

12.16.3.44 int com.idtechproducts.device.IDT_Device.icc_setBCDMask (byte *maskWithBCD*) [inline]

Set BCD Mask Data. This setting will be valid in the display of TAG 57/5A at the EMV Level 2 transaction processing.

Parameters

maskWithAscii,can	be from 0x0A to 0x0F, default is 0x0C(∗)

Returns

success or error code.

See also

ErrorCode

12.16.3.45 int com.idtechproducts.device.IDT_Device.icc_setICCConnector (byte *connector*) [inline]

Set the ICC or SAM connector.

Parameters

connector,the	value to switch the ICC or SAM connector 0x00: use the ICC connector. 0x01: use the SAM connector.

Returns

success or error code.

See also

ErrorCode

12.16.3.46 int com.idtechproducts.device.IDT_Device.icc_setProAndPostPanLength (int *prePanCtrlDataLen,* int *postPanCtrlDataLen*) [inline]

Set Pre/Post Pan Ctrl Data length. This setting will be valid in the display of TAG 57/5A at the EMV Level 2 transaction processing.

Parameters

prePanCtrl↩ DataLen,the	length to show the pre PAN character with plaintext
postPanCtrl↩ DataLen,the	length to show the post PAN character with plaintext

Returns

success or error code.

See also

ErrorCode

12.16.3.47 int com.idtechproducts.device.IDT_Device.lcd_clearDisplay (byte *control*) [inline]

LCD API LIST Clear Display.

Parameters

control,0	first line, 1: second line, 0xFF: clear all screen

Returns

success or error code.

See also

ErrorCode

12.16.3.48 int com.idtechproducts.device.IDT_Device.lcd_defaultSetting ( ) [inline]

Default LCD group setting.

Returns

success or error code.

See also

ErrorCode

12.16.3.49 int com.idtechproducts.device.IDT_Device.lcd_displayMessage (int *numLine,* String *message*) [inline]

Display message.

Parameters

numLine,the	index of line number, 0 or 1
message	Display line 2, up to 16 characters.

Returns

success or error code.

See also

ErrorCode

12.16.3.50 int com.idtechproducts.device.IDT_Device.lcd_displayPrompt (int *numLine,* int *numPrompt*) [inline]

Display the prompt message.

Parameters

numLine,the	index of line number, 0 or 1
numPrompt,the	number of prompt, from 0 to 9

Returns

success or error code.

See also

ErrorCode

12.16.3.51 int com.idtechproducts.device.IDT_Device.lcd_enableSleepMode (boolean *sleepMode*) [inline]

Enable the LCD to enter into sleep mode.

Parameters

sleep↩ Mode,true:set	the device to sleep mode; false:exit sleep mode.

Returns

success or error code.

See also

ErrorCode

12.16.3.52 int com.idtechproducts.device.IDT_Device.lcd_getSetting (boolean[] *sleepMode*) [inline]

Review LCD Setting.

Parameters

sleep↩ Mode,sleep↩ Mode[0]	means the sleep mode true: the LCD is on the sleep mode, the back light turned off false: the LCD exist the sleep mode, the back light turned on

Returns

success or error code.

See also

ErrorCode

12.16.3.53 int com.idtechproducts.device.IDT_Device.lcd_savePromptDisplay (int *numPrompt,* String *message*) [inline]

Save prompt display message.

Parameters

numPrompt,the	number of prompt, from 0 to 9
message	to Display, up to 16 characters.

Returns

success or error code.

See also

ErrorCode

12.16.3.54 int com.idtechproducts.device.IDT_Device.log_deleteLogs ( ) [inline]

delete the log in the root path of SD card.

Returns

none

See also

log_setSaveLogEnable

12.16.3.55 void com.idtechproducts.device.IDT_Device.log_setSaveLogEnable (boolean *enable*) [inline]

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

Parameters

enableShow↩ Log,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

12.16.3.56 void com.idtechproducts.device.IDT_Device.log_setVerboseLoggingEnable (boolean *enableShowLog*) [inline]

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

Parameters

enableShow↩ Log,true	enable to show the log in the logcat view window. false: disable to show the log in the logcat view window.

Returns

none

12.16.3.57 int com.idtechproducts.device.IDT_Device.msr_cancelMSRSwipe ( ) [inline]

Disable MSR swipe card.

Returns

success or error code.

See also

ErrorCode

12.16.3.58 int com.idtechproducts.device.IDT_Device.msr_defaultAllSetting ( ) [inline]

Default all setting of Mask and Encryption.

Returns

success or error code.

See also

ErrorCode

12.16.3.59 int com.idtechproducts.device.IDT_Device.msr_getSingleSetting (byte *funcID,* byte[] *response*) [inline]

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.

See also

ErrorCode

12.16.3.60 boolean com.idtechproducts.device.IDT_Device.msr_isSwipeCardRunning ( ) [inline]

get the status if the swipe card task running.

Returns

true: swipe card task is running, false: swipe card task not running

12.16.3.61 int com.idtechproducts.device.IDT_Device.msr_reviewAllSetting (MSRSettingStruct *msrSetting*) [inline]

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.

See also

ErrorCode

12.16.3.62 int com.idtechproducts.device.IDT_Device.msr_setSingleSetting (byte *funcID*, byte *setData*) [inline]

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.

See also

ErrorCode

12.16.3.63 int com.idtechproducts.device.IDT_Device.msr_startMSRSwipe ( ) [inline]

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.

See also

ErrorCode

12.16.3.64 String com.idtechproducts.device.IDT_Device.phone_getInfoManufacture ( ) [inline]

Get manufacture version.

Returns

the manufacture info

12.16.3.65 String com.idtechproducts.device.IDT_Device.phone_getInfoModel ( ) [inline]

Get phones's model number information.

Returns

the model number information.

12.16.3.66 void com.idtechproducts.device.IDT_Device.registerListen ( ) [inline]

General API:registerListen.

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

12.16.3.67 void com.idtechproducts.device.IDT_Device.release ( ) [inline]

release, make the SDK in the idle status.

12.16.3.68 void com.idtechproducts.device.IDT_Device.setEMVListener (OnReceiverListener *callback*) [inline]

Allows the EMV library to receive notifications from the OnReceiverListener

12.16.3.69 boolean com.idtechproducts.device.IDT_Device.setWaitingExchangeAPDUResponseTimeout (int *timeoutValue*) [inline]

set the timeout for waiting Response from the reader.
the timeout works on the waiting for Exchange APDU,

Parameters

timeoutValue	: the seconds value to wait the response, valid from 5 seconds to 10 seconds,

Returns

true or false. true: the timoutValue is valid, false: the timeoutValue is invalid,

12.16.3.70 boolean com.idtechproducts.device.IDT_Device.setWaitingMSRResponseTimeout (int *timeoutValue*) [inline]

set the timeout for waiting Response from the reader.
the timeout works on the waiting for swiping MSR.

Parameters

timeoutValue	: the seconds value to wait the response, valid from 30 seconds to 35 seconds.

Returns

true or false. true: the timoutValue is valid, false: the timeoutValue is invalid,

12.16.3.71 boolean com.idtechproducts.device.IDT_Device.setWaitingPINResponseTimeout (int *timeoutValue*) [inline]

set the timeout for waiting Response from the reader.
the timeout works on the waiting for getting the PIN/Numeric.

Parameters

timeoutValue	: the seconds value to wait the response, valid from 30 seconds to 60 seconds,

Returns

true or false. true: the timoutValue is valid, false: the timeoutValue is invalid,

12.16.3.72 void com.idtechproducts.device.IDT_Device.stopWaitingTask ( ) [inline]

Stop the waiting task which wait the device response.

Returns

null

12.16.3.73 void com.idtechproducts.device.IDT_Device.unregisterListen ( ) [inline]

unregisterListen to disable the detect

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/IDT_Device.java

12.17 com.idtechproducts.emv.UniPayEMV.IDTEMVData Class Reference

Public Member Functions

- void **Clear** ()
- CAPTURE_ENCRYPT_TYPE **getCaptureEncryptType** ()
- CAPTURE_TYPE **getCaptureType** ()
- Map< String, byte[]> **getUnencrytpedTags** ()
- Map< String, byte[]> **getEncryptedTags** ()
- Map< String, byte[]> **getMaskedTags** ()
- byte[] **getKSN** ()
- void **setCaptureEncryptType** (CAPTURE_ENCRYPT_TYPE val)
- void **setCaptureType** (CAPTURE_TYPE val)
- void **setUnencrytpedTags** (Map< String, byte[]> val)
- void **setEncryptedTags** (Map< String, byte[]> val)
- void **setMaskedTags** (Map< String, byte[]> val)
- void **setKSN** (byte[] val)

Data Fields

- CAPTURE_ENCRYPT_TYPE **captureEncryptType**
- CAPTURE_TYPE **captureType**
- Map< String, byte[]> **unencryptedTags**
- Map< String, byte[]> **encryptedTags**
- Map< String, byte[]> **maskedTags**
- byte[] **KSN**

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.18 com.idtechproducts.device.IDTMSRData Class Reference

Data Fields

- EVENT_MSR_Types event
- byte[ ] cardData
- byte[ ] encTrack1
- byte[ ] encTrack2
- byte[ ] encTrack3
- String track1
- String track2
- String track3
- String serialNumber
- byte[ ] KSN
- int track1Length
- int track2Length
- int track3Length
- CAPTURE_ENCODE_TYPE cardType
- byte captureEncodeStatus
- CAPTURE_ENCRYPT_TYPE captureEncryptType

12.18.1 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.

12.18.2 Field Documentation

12.18.2.1 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;

12.18.2.2 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;

12.18.2.3 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.

12.18.2.4 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

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

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

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

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

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

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

12.18.2.8 EVENT_MSR_Types com.idtechproducts.device.IDTMSRData.event

MSR type,please see EVENT_MSR_Types for more information.

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

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

12.18.2.10 String com.idtechproducts.device.IDTMSRData.serialNumber

Get the Reader Serial Number.

12.18.2.11 String com.idtechproducts.device.IDTMSRData.track1

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

12.18.2.12 int com.idtechproducts.device.IDTMSRData.track1Length

Get the swiped card length of Track1 data.

12.18.2.13 String com.idtechproducts.device.IDTMSRData.track2

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

12.18.2.14 int com.idtechproducts.device.IDTMSRData.track2Length

Get the swiped card length of Track2 data.

12.18.2.15 String com.idtechproducts.device.IDTMSRData.track3

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

12.18.2.16 int com.idtechproducts.device.IDTMSRData.track3Length

Get the swiped card length of Track3 data.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/IDTMSRData.java

12.19 com.idtechproducts.emv.UniPayEMV.MESSAGE_Types Enum Reference

Public Member Functions

- String **toString** ()

Data Fields

- **MESSAGE_INSERT_OR_SWIPE_CARD** =(0)
- **MESSAGE_REMOVE_CARD** =(1)
- **MESSAGE_BAD_ICC** =(2)
- **MESSAGE_TRANSACTION_CANCELLED** =(3)
- **MESSAGE_FALLBACK_FAILED** =(4)
- **MESSAGE_USE_CHIP_READER** =(5)
- **MESSAGE_PROCESSING** =(6)
- **MESSAGE_READY** =(7)
- **MESSAGE_USE_MSR** =(8)
- **MESSAGE_NOT_ACCEPTED** =(9)
- **MESSAGE_SDK_ERROR** =(10)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.20 com.idtechproducts.device.MSRSettingStruct Class Reference

Data Fields

- byte prePANID

- byte postPANID
- byte maskCharID
- byte dispExpDateID
- byte EnOptionID
- byte MaskOptID
- byte EncryptionType
- byte Security_LevelID

12.20.1 Detailed Description

Response data for all settings of Mask and Encryption.

UniPay support prePANID,EncryptionType,dispExpDateID,Security_LevelID,EnOptionID,MaskOptID.
UniPay II support prePANID,postPANID,maskCharID,EncryptionType,dispExpDateID,Security_LevelID,EnOptionI↩
D,MaskOptID.

12.20.2 Field Documentation

12.20.2.1 byte com.idtechproducts.device.MSRSettingStruct.dispExpDateID

Mask or display expiration date,one byte,default value:0x31 don't mask expiration date(0x30 or 0x31).

12.20.2.2 byte com.idtechproducts.device.MSRSettingStruct.EncryptionType

Encryption type ('1'-'2'). '1' 3DES, '2' AES.

12.20.2.3 byte com.idtechproducts.device.MSRSettingStruct.EnOptionID

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

12.20.2.4 byte com.idtechproducts.device.MSRSettingStruct.maskCharID

Character used to mask PAN,one byte,default value:0x2A(0x20∼0x7A).

12.20.2.5 byte com.idtechproducts.device.MSRSettingStruct.MaskOptID

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

12.20.2.6 byte com.idtechproducts.device.MSRSettingStruct.postPANID

Last PAN digits to display,one byte,default value:0x04(0x00∼0x04).

12.20.2.7 byte com.idtechproducts.device.MSRSettingStruct.prePANID

Leading PAN digits to display,one byte, default value:0x04(0x00∼0x06).

12.20.2.8 byte com.idtechproducts.device.MSRSettingStruct.Security_LevelID

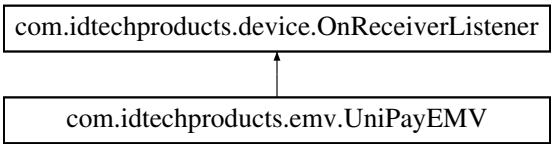
Security Level ID

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/MSRSettingStruct.java

12.21 com.idtechproducts.device.OnReceiverListener Interface Reference

Inheritance diagram for com.idtechproducts.device.OnReceiverListener:



Data Structures

- enum EMV_RESULT_CODE_Types
- enum USER_GRANT_TYPE

Public Member Functions

- void swipeMSRData (IDTMSRData card)
- void deviceConnected ()
- void deviceDisconnected ()
- void plugStatusChange (DEVICE_INTERFACE_Types deviceInterface, boolean deviceInserted)
- void timeout (String msgInfo)
- void AutoConfigCompleted (StructConfigParameters profile)
- void AutoConfigProgress (int progressValue)
- void ICCNotifyInfo (byte[] dataNotify, String strMessage)
- boolean getUserGrant (USER_GRANT_TYPE nType, String strMessage)
- void LoadXMLConfigFileInfo (int index, String strMessage)
- void msgToConnectDevice ()
- void msgAudioVolumeAjustFailed (String strMessage)

12.21.1 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.

12.21.2 Member Function Documentation

12.21.2.1 void com.idtechproducts.device.OnReceiverListener.AutoConfigCompleted (StructConfigParameters *profile*)

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

12.21.2.2 void com.idtechproducts.device.OnReceiverListener.AutoConfigProgress (int *progressValue*)

The auto config process percent value.

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

Fires when device connects.

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

Fires when device disconnects.

12.21.2.5 boolean com.idtechproducts.device.OnReceiverListener.getUserGrant (USER_GRANT_TYPE *nType,* String *strMessage*)

Get the user grant to continue process ,

Parameters

nType	0: grantConnectDevice: get the grant to connect the device, post when the phone detect the phone jack plug in the device. 1: grantUpdateXML: get the grant to update the configuration XML file. 2: grantOverwriteXML:get the grant to overwrite current configuration XML file. 3: grantTryAutoConfig: get the grant to search the profile through Auto config mode. 4: grantReportToIdtech: get the grant to report to ID TECH supports.
strMessage,the	grant type message information.

12.21.2.6 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.

12.21.2.7 void com.idtechproducts.device.OnReceiverListener.LoadXMLConfigFileInfo (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.

12.21.2.8 void com.idtechproducts.device.OnReceiverListener.msgAudioVolumeAjustFailed (String *strMessage*)

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.

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

The message notify the application to connect the device.

12.21.2.10 void com.idtechproducts.device.OnReceiverListener.plugStatusChange (DEVICE_INTERFACE_Types *deviceInterface,* boolean *deviceInserted*)

Notify the plug status of phone jack.

12.21.2.11 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.

12.21.2.12 void com.idtechproducts.device.OnReceiverListener.timeout (String *msgInfo*)

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:

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/OnReceiverListener.java

12.22 com.idtechproducts.device.PowerOnStructure Class Reference

Data Fields

- boolean sendIFS
- boolean explicitPPS
- boolean disableAutoPPS
- boolean disableResponseCheck
- byte[] pps
- byte ppsLength

12.22.1 Detailed Description

The class for PowerOn Option.

12.22.2 Field Documentation

12.22.2.1 boolean com.idtechproducts.device.PowerOnStructure.disableAutoPPS

true:No auto PPS for negotiate mode; false:Auto PPS.

12.22.2.2 boolean com.idtechproducts.device.PowerOnStructure.disableResponseCheck

true:No check on response of S(IFS) request; false:IFS response check.

12.22.2.3 boolean com.idtechproducts.device.PowerOnStructure.explicitPPS

true:Explicit PPSError: Reference source not found; false:No Explicit PPS.

12.22.2.4 byte [ ] com.idtechproducts.device.PowerOnStructure.pps

pps is used to set the Protocol and Parameters Selection between card and reader, only Di <= 4 are supported. pps must follow the structure specified in ISO 7816-3 as PPS0, [PPS1], [PPS2], and [PPS3]. For more information see ISO 7816-3 section 7.2.

12.22.2.5 byte com.idtechproducts.device.PowerOnStructure.ppsLength

the length of pps data.

12.22.2.6 boolean com.idtechproducts.device.PowerOnStructure.sendIFS

true:Send S(IFS) request if T=1 protocolError: Reference source not found; false:no IFS.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/PowerOnStructure.java

12.23 com.idtechproducts.device.ReaderInfo Class Reference

Data Structures

- enum CAPTURE_ENCODE_TYPE
- enum CAPTURE_ENCRYPT_TYPE
- enum DEVICE_INTERFACE_Types
- enum EVENT_MSR_Types
- enum ReaderType
- enum SupportStatus

Static Public Attributes

- static DEVICE_INTERFACE_Types **detectedDeviceInterface** = DEVICE_INTERFACE_Types.DEVICE_U↩ NKNOWN
- static DEVICE_INTERFACE_Types **connectedInterface** = DEVICE_INTERFACE_Types.DEVICE_UNKN↩ OWN

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.24 com.idtechproducts.device.ReaderInfo.ReaderType Enum Reference

Data Fields

- **UNKNOWN**
- **UM_OR_PRO**
- **UM**
- **UM_PRO**
- **UM_II**
- **SHUTTLE**
- **UNIPAY**
- **UNIPAYII**

12.24.1 Detailed Description

Define the reader type.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.25 com.idtechproducts.device.ReaderInfo.SupportStatus Enum Reference

Data Fields

- **UNSUPPORTED**
- **SUPPORTED**
- **MAYBE_SUPPORTED**

12.25.1 Detailed Description

Define the phone support status.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/ReaderInfo.java

12.26 com.idtechproducts.emv.UniPay_ApplicationID Class Reference

Data Fields

- byte[ ] acquirerIdentifier = new byte[6]

  Indicates which acquirer/processor processes the corresponding AID. Tag 9F01.
- byte[ ] aid = new byte[16]

  AID value as per payment networks. Tag 9F06.
- byte[ ] aidLen = new byte[1]

  AID's length.
- byte[ ] applicationVersionNumber = new byte[2]

  EMV application version number. Tag 9F09.
- byte[ ] tacDefault = new byte[5]

  Terminal Action Code Denial. Tag DF13.
- byte[ ] tacDenial = new byte[5]

  Terminal Action Code Denial. Tag DF14.
- byte[ ] tacOnline = new byte[5]

  Terminal Action Code Denial. Tag DF15.
- byte[ ] transactionCurrencyCode = new byte[2]

  AID's currency. Example: For US, {0x08,0x40}. Tag 5F2A.
- byte[ ] transactionCurrencyExponent = new byte[1]

  Transaction Currency Exponent. Example: Amount $4.53 is managed as 453. Tag 5F36.
- byte[ ] useTACDefault = new byte[1]

  Indicates if tacDefault value should be used.
- byte[ ] useTACDenial = new byte[1]

  Indicates if tacDefault value should be used.
- byte[ ] useTACOnline = new byte[1]

  Indicates if tacDefault value should be used.
- byte[ ] applicationSelectionIndicator = new byte[1]

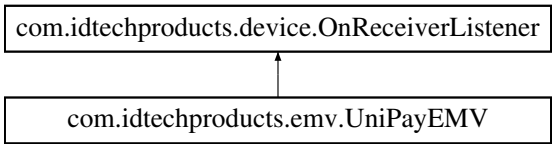
  Indicates if partial AID matching is allowed. 0x01 = allowed DF62.

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPay_ApplicationID.java

12.27 com.idtechproducts.emv.UniPayEMV Class Reference

Inheritance diagram for com.idtechproducts.emv.UniPayEMV:



Data Structures

- enum CAPTURE_ENCODE_TYPE
- enum CAPTURE_ENCRYPT_TYPE
- enum CAPTURE_TYPE
- enum EMV_COMPLETION_RESULT
- enum EMV_RESULT_CODE_Types
- enum EVENT_MSR_Types
- class IDTEMVData
- enum MESSAGE_Types
- interface UniPayEMVDelegate

Public Member Functions

- void accelerateRead (boolean enabled)
- UniPayEMV (Context context, IDT_Device reader, UniPayEMVDelegate del)
- void setTerminalData (UniPayTerminalDataStruct data)
- void confirmApplicationCancel ()
- void confirmApplication (int index)
- void cancelTransaction ()
- void startEMVTransaction (double amount, int timeout, int type, Map< String, byte[]> tags)
- String getEMVKernelVersion ()
- boolean completeOnlineEMVTransaction (EMV_COMPLETION_RESULT result, String resultCode, String iad, String scripts)
- UniPayTerminalDataStruct retrieveTerminalData ()
- void setApplicationData (UniPay_ApplicationID data)
- UniPay_ApplicationID retrieveApplicationData (String data)
- void removeApplicationData (String data)
- Vector< String > retrieveAIDList ()

12.27.1 Constructor & Destructor Documentation

12.27.1.1 com.idtechproducts.emv.UniPayEMV.UniPayEMV (Context *context,* IDT_Device *reader,* UniPayEMVDelegate *del*) [inline]

UniPay EMV

Initiates a UniPay EMV instance.

Parameters

context	Context of application calling method
reader	Instance of IDT_Device SDK
del	Pass instance of delegate that will receive UniPayEMV messages

12.27.2 Member Function Documentation

12.27.2.1 void com.idtechproducts.emv.UniPayEMV.accelerateRead (boolean *enabled*) [inline]

Accelerate Read Data

Enables multi-record reading from ICC to accelerate EMV transaction time. Enabled by default.

Parameters

enabled	TRUE = use accelerated reading, FALSE = use standard reading

12.27.2.2 void com.idtechproducts.emv.UniPayEMV.cancelTransaction ( ) [inline]

Cancel Transaction

Cancels the current transaction

12.27.2.3 boolean com.idtechproducts.emv.UniPayEMV.completeOnlineEMVTransaction (EMV_COMPLETION_RESULT *result,* String *resultCode,* String *iad,* String *scripts*) [inline]

Complete EMV Transaction Online Request

Completes an online EMV transaction request by the card

The tags will be returned in the emvTransactionData delegate protocol.

Parameters

result	EMV_COMPLETION_RESULT: Used to specify if contacting online host was successful or other problem occurred
resultCode	Result Code from host. Manditory. 2 characters ASCII value. Example "00"
iad	Issuer Authentication Data. Optional. 10 bytes, 20 Hex Characters representing data. Example "11223344556677883030"
scripts	Issuer Scripts. Optional. Data represented by Hex Characters. TLV Format. Must start with 71 or 72, followed by length, followed by data. Example "7110001122334455667788 99AABBCCDDEEFF". Can string multiple scripts, both 71 and 72.

Results are returned on delegate protocol emvTransactionData.

12.27.2.4 void com.idtechproducts.emv.UniPayEMV.confirmApplication (int *index*) [inline]

Set Application

During and EMV transaction flow, if there are multiple applications to chose from, or the terminal settings indicate cardholder confirmation for applicaiton selection, the delegate IDTechEMV_Delegate::confirmApplicationSelection:() will receive an array with all the available applications to choose from. The selected index of the application must be passed back to this method to continue the EMV transaction flow

Parameters

index	The index of the selected app from the application array passed back from confirmApplicationSelection:()

12.27.2.5 void com.idtechproducts.emv.UniPayEMV.confirmApplicationCancel ( ) [inline]

Cancel Set Application

During and EMV transaction flow, if there are multiple applications to chose from, or the terminal settings indicate cardholder confirmation for application selection, the delegate IDTechEMV_Delegate::confirmApplicationSelection:() will receive an array with all the available applications to choose from. If no application selection is performed, this routine must be called to cancel the transaction

12.27.2.6 String com.idtechproducts.emv.UniPayEMV.getEMVKernelVersion ( ) [inline]

Returns SDK EMV Kernel Version

Returns

response Kernel Version

12.27.2.7 void com.idtechproducts.emv.UniPayEMV.removeApplicationData (String *data*) [inline]

Remove Application Data by AID

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

Parameters

AID	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters

12.27.2.8 Vector<String> com.idtechproducts.emv.UniPayEMV.retrieveAIDList ( ) [inline]

Retrieve AID list

Returns all the AID names on the terminal.

Return values

response	Returns a Vector<String> of AID Names

12.27.2.9 UniPay_ApplicationID com.idtechproducts.emv.UniPayEMV.retrieveApplicationData (String *data*) [inline]

Retrieve Application Data by AID

Retrieves the Application Data as specified by the AID name passed as a parameter. If aidLen = 0, then requested AID was not found.

Parameters

data	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters

Returns

responseAID The AID returned as UniPay_ApplicationID

12.27.2.10 UniPayTerminalDataStruct com.idtechproducts.emv.UniPayEMV.retrieveTerminalData ( ) [inline]

Retrieve Terminal Data

Retrieves the Terminal Data. The data will be UniPayTerminalDataStruct

Returns

response Response returned as a TerminalData

12.27.2.11 void com.idtechproducts.emv.UniPayEMV.setApplicationData (UniPay_ApplicationID *data*) [inline]

Set Application Data by AID

Sets the Application Data as specified by the UniPay_ApplicationID structure passed as a class

Parameters

data	UniPay_ApplicationID configuration file

12.27.2.12 void com.idtechproducts.emv.UniPayEMV.setTerminalData (UniPayTerminalDataStruct *data*) [inline]

Set Terminal Data

Sets the Terminal Data as specified by the UniPayTerminalDataStruct class passed as a parameter

Parameters

data	TerminalData configuration file

12.27.2.13 void com.idtechproducts.emv.UniPayEMV.startEMVTransaction (double *amount*, int *timeout*, int *type*, Map< String, byte[]> *tags*) [inline]

Start EMV Transaction Request

Authorizes the EMV transaction amounts for an ICC card

The tags will be returned in the emvTransactionData delegate protocol.

Parameters

amount	Transaction amount value (tag value 9F02)
timeout	Timeout value in seconds.
type	Transaction Type.
tags	Any other optional tags to be included in the request. Passed as a Map<String,byte[]>.

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.28 com.idtechproducts.emv.UniPayEMV.UniPayEMVDelegate Interface Reference

Public Member Functions

- void **emvTransactionMessage** (UniPayEMV.MESSAGE_Types message)
- void **confirmApplicationSelection** (Vector< String > lableArray, boolean tryAgain)
- void **languagePreference** (byte[] lang)
- void **emvTransactionData** (IDTEMVData emvData, UniPayEMV.EMV_RESULT_CODE_Types errorCode, boolean performReversal)
- void **swipeMSRDataEMV** (IDTMSRData cardData, Map< String, byte[]> emvData)

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayEMV.java

12.29 com.idtechproducts.emv.UniPayTerminalDataStruct Class Reference

Data Fields

- byte[] terminalCountryCode = new byte[2]
- byte[] merchantCategoryCode = new byte[2]

- byte[ ] merchantID = new byte[15]
- byte[ ] terminalID = new byte[8]
- byte[ ] defaultTACDenial = new byte[5]
- byte[ ] defaultTACOnline = new byte[5]
- byte[ ] defaultTACDefault = new byte[5]
- byte[ ] useDefaultTACDefault = new byte[1]
- byte[ ] useDefaultTACDenial = new byte[1]
- byte[ ] useDefaultTACOnline = new byte[1]
- byte[ ] terminalLocation = new byte[36]

12.29.1 Field Documentation

12.29.1.1 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.defaultTACDefault = new byte[5]

Default TAC Default value. See EMVCo book IV.

12.29.1.2 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.defaultTACDenial = new byte[5]

IDefault TAC Denial value. See EMVCo book IV.

12.29.1.3 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.defaultTACOnline = new byte[5]

Default TAC Online value. See EMVCo book IV.

12.29.1.4 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.merchantCategoryCode = new byte[2]

The Terminal's Merchant Category Code. Classifies the type of business being done by the mechant, represented according to ISO 8583:1993.

12.29.1.5 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.merchantID = new byte[15]

When concatenated with the Acquirer Identifer, uniquely identifies a given merchant.

12.29.1.6 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.terminalCountryCode = new byte[2]

The Terminal's Country Code,example:{0x08,0x40}.

12.29.1.7 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.terminalID = new byte[8]

Designates the unique location of a terminal at a merchant

12.29.1.8 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.terminalLocation = new byte[36]

Terminal Location tag 9F4E.

12.29.1.9 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.useDefaultTACDefault = new byte[1]

Indicates if tacDefault value should be used as terminal default value.

12.29.1.10 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.useDefaultTACDenial = new byte[1]

Indicates if tacDenial value should be used as terminal default value.

12.29.1.11 byte [ ] com.idtechproducts.emv.UniPayTerminalDataStruct.useDefaultTACOnline = new byte[1]

Indicates if tacObline value should be used as terminal default value.

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

- /Users/randy/Repo/Android/UniPayEMV/src/com/idtechproducts/emv/UniPayTerminalDataStruct.java

12.30 com.idtechproducts.device.OnReceiverListener.USER_GRANT_TYPE Enum Reference

Data Fields

- **grantConnectDevice**
- **grantUpdateXML**
- **grantOverwriteXML**
- **grantTryAutoConfig**
- **grantReportToIdtech**

12.30.1 Detailed Description

Define the User grant type. grantConnectDevice: get the grant to connect the device, post when the phone detect the phone jack plug in the device. grantUpdateXML: get the grant to update the configuration XML file. grant↩ OverwriteXML:get the grant to overwrite current configuration XML file. grantTryAutoConfig: get the grant to search the profile through Auto config mode. grantReportToIdtech: get the grant to report to ID TECH supports.

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

- /Users/randy/Repo/Android/IDTechSDK/src/com/idtechproducts/device/OnReceiverListener.java