IDTECH®

Value through Innovation

Apple iOS SDK Guide

for NEO2

#80152802-001

Rev. A

Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;NEO2;SDK;iOS	1/29/2018

Contents

Chapter 1

IDTech iOS SDK Reference Guide for NEO2

IDTech.framework is an Apple Framework that will be provided by IDTech as the main interface between iOS applications, the NEO2 and payment 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 iOS applications wishing to deploy with the payment application.

Chapter 2

Updating device firmware using PK Files

Certain models can be updated using an file archive with a .pk extension. A .pk file is a firmware update package that can be transferred to the device, and once the transfer is complete, the device will go into DFU mode and apply the files.
To send the file to the device, you use the following functions from the SDK:

```
//for a device connected with a cable (USB-C or Lightning)
-(RETURN_CODE) sendPKUpdate:(NSData*)pkFile;

//for a device connected wirelessly (BLE)
-(RETURN_CODE) device_sendPKUpdate:(NSData*)pkData;

//Example Usage
//Objective-C
[[IDT_Device sharedController] sendPKUpdate:fileData];   //lightning or usb-c
[[IDT_Device sharedController] device_sendPKUpdate:fileData];    //BLE

//Swift
IDT_Device.sharedController().sendPKUpdate(fileData)   //lightning or usb-c
IDT_Device.sharedController().device_sendPKUpdate(fileData)    //BLE
```

To monitor firmware update progress and final outcome, use the updateStatus delegate:

```
-(void) updateStatus:(PK_STATUS_Type)type currentBlock:(int)currentBlock totalBlocks:(int)totalBlocks error
      :(RETURN_CODE)error;

//Objective-C
-(void) updateStatus:(PK_STATUS_Type)type currentBlock:(int)currentBlock totalBlocks:(int)totalBlocks error
      :(RETURN_CODE)error{
    switch (type) {
        case PK_STATUS_COMPLETED:
            [self appendMessageToResults:@"PK Update Process Complete"];
            break;
        case PK_STATUS_FAILED:
            [self displayUpRet2: @"PK Update Process Failed: " returnValue: error];
            break;
        case PK_STATUS_STARTED:
            [self appendMessageToResults:@"PK Update Process Started"];
            break;
        case PK_STATUS_APPLYING_UPDATE:
            [self appendMessageToResults:@"PK Update Process Applying Update. Please wait..."];
            break;
        case PK_STATUS_SENDING_BLOCK:
            [self appendMessageToResults:[NSString stringWithFormat:@"Sending block %d of %d", currentBlock
      , totalBlocks]];
            break;
        default:
            break;
    }
}

//Swift
    func updateStatus(_ type: PK_STATUS_Type, currentBlock: Int32, totalBlocks: Int32, error: RETURN_CODE)
      {
        switch type {
        case PK_STATUS_COMPLETED:
            connection.appendMessageToDataField("PK Update Process Completed", field: resultsTextView)
            connection.pkUpdate = false
            return
        case PK_STATUS_FAILED:
```

```
            connection.appendMessageToDataField("PK Update Process Failed", field: resultsTextView)
            return
        case PK_STATUS_STARTED:
            connection.appendMessageToDataField("PK Update Process Started", field: resultsTextView)
            return
        case PK_STATUS_APPLYING_UPDATE:
            connection.appendMessageToDataField("PK Update Process Applying Update. Please wait...", field:
     resultsTextView)
            return
        case PK_STATUS_SENDING_BLOCK:
            connection.appendMessageToDataField(String(format: "Sending block %d of %d", currentBlock,
    totalBlocks), field: resultsTextView)
            return
        default:
            return
        }
    }
```

Chapter 3

Connecting to NEO2 BLE With Automatic Location

The SDK has the ability to scan the area for BLE devices, and present an UIAlertView to allow user selection of a found device.
The SDK must be told which device to filter for, how long to search for devices before presenting results, and what delegate to send the results to.
To accomplish this, you define the Service Characteristic of the NEO2, define your class as the delegate, and then enable the SDK device search. NOTE: The VP3600 does not have Service Characteristics you can filter for. For the VP3600, you must not define a service characteristic when using Automatic Location
First, make sure your class is defined as a delegate for the SDK

```
//Objective-C
@interface ViewController : UIViewController<IDT_NEO2_Delegate>

//Swift
class ViewController: UIViewController, IDT_NEO2_Delegate
```

Then execute the following code. In this example, we are asking SDK to search for 2 seconds for all NEO2 in the vacinity

```
//The first set of examples is filtering ON for all NEO2 BLE devices that are NOT VP3600
//The second set example is filtering OFF for locating VP3600

//Objective-C, search for VP3320, VP3350
    NSString* SPS_SERVICE_3320 =  @"49535343-FE7D-4AE5-8FA9-9FAFD205E455";
    NSString* SPS_SERVICE_3350 =  @"0783b03e-8535-b5a0-7140-a304d2495cb7";
    [[IDT_NEO2 sharedController] setDelegate:self];
    [[IDT_NEO2 sharedController] scanForBLEDevices:2.0 serviceUUIDs:@[[CBUUID UUIDWithString:
      SPS_SERVICE_3320],[CBUUID UUIDWithString:SPS_SERVICE_3350]] options:nil];

//Swift, search for VP3320, VP3350
    IDT_NEO2.sharedController().delegate = self
    let svc3320 = CBUUID.init(string: "49535343-FE7D-4AE5-8FA9-9FAFD205E455")
    let svc3350 = CBUUID.init(string: "0783b03e-8535-b5a0-7140-a304d2495cb7")
    let svcArray = Array<CBUUID>.init(arrayLiteral: svc3320, svc3350)
    IDT_NEO2.sharedController()?.scan(forBLEDevices: 2.0, serviceUUIDs: svcArray,
      options: nil)

//Objective-C, search ANY device, including VP3600
    [[IDT_NEO2 sharedController] setDelegate:self];
    [[IDT_NEO2 sharedController] scanForBLEDevices:2.0 serviceUUIDs:nil options:nil];

//Swift, search ANY device, including VP3600
    IDT_NEO2.sharedController().delegate = self
    IDT_NEO2.sharedController()?.scan(forBLEDevices: 2.0, serviceUUIDs: nil,
      options: nil)
```

After the scan is complete, the SDK will return the alertView to the bluetoothPickerAlert. You simply show this view:

```
//Objective-C
- (void) bluetoothPickerAlert:(UIAlertView*)view{
    [view show];
    }
//Swift
func bluetoothPickerAlert(_ view: UIAlertView!) {
        view.show()
    }
}
```

Any device selected in that UIAlertView will be automatically connected. There is no further coding needed to process the selection, as the SDK is the delegate for that UIAlertView and will automtically process the selection.

Chapter 4

Connecting to NEO2 BLE With Manual Location

A version of NEO2 uses Bluetooth 4.0, also known as Bluetooth BLE (Bluetooth Low Energy). Unlike previous version of Bluetooth, with BLE you don't need to pair first through the Bluetooth setting on your iOS device. Once BLE scanning is enabled (via a SDK call), the Apple iOS scans and locates all BLE devices in range and presents a list for the SDK to choose from.

Unlike other operating systems where you can locate a BLE device by it's MAC address, due to security reasons Apple does NOT allow you to specify a device by this address. Instead, once a device is selected (by its friendly name), the Apple iOS will calculate a unique identifier (using parameters as the BLE device MAC address and the iOS device UID) so any further connections can be made directly to that device.

The NEO2 has a default friendly name of "Enzytek SPS_S". This is the default name the SDK uses to connect to the first NEO2 it encounters when no other friendly name is set in the SDK, or if the iOS generated device identifier is not provided.

To see the current default BLE name the SDK will use: device_getBLEFriendlyName (IDT_NEO2)

If the NEO2 friendly name is different than "Enzytek SPS-S", then the SDK can be set to recognize this new name: device_setBLEFriendlyName: (IDT_NEO2)

The API method to start BLE scanning : device_enableBLEDeviceSearch: (IDT_NEO2)

The very first time a NEO2 is used with a particular iOS device, it must be located by it's friendly name (as the device identifier hasn't been calculated by the iOS). If there are multiple NEO2's in range, it will stop at the first one located.

```
[[IDT_NEO2 sharedController] device_enableBLEDeviceSearch:nil];
```

Once the device is connected, its UUID can be retrieved with device_connectedBLEDevice (IDT_NEO2)

```
NSUUID* identifier = [[IDT_NEO2 sharedController] device_connectedBLEDevice];
```

That identifier can be stored in persistent memory and used for future connection to that device on the same iOS device

```
[[IDT_NEO2 sharedController] device_enableBLEDeviceSearch:identifier];
```

Scanning for BLE devices has an impact on battery life. Once you start scanning with IDT_NEO2::device_enable↩ BLEDeviceSearch:identifier:(), it will continue to scan until either a device is found (friendly name match, or identifier match) OR a cancel BLE scanning is executed with device_disableBLEDeviceSearch (IDT_NEO2)

As an alternate indicator of scanning progress, and of any devices the iOS found, every BLE device located is reported back to deviceMessage delegate. This information can be used to confirm if a device is in range, verify the device friendly name, or display the iOS calculated UUID for that device. The message will start with "BLE DEVICE FOUND:...."

```
BLE DEVICE FOUND: Enzytek SPS_S (B469AGF6-A9C0-4FF2-AF2A-2ECD18EDB506)
```

Chapter 5

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.

5.1 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.

5.2 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.

5.3 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.

5.4 PA-DSS Guidelines

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

1. Sensitive Data Storage Guidelines

Do not retain full magnetic stripe, card validation code or value (CAV2, CID, CVC2, CVV2), or PIN block data. 1.1 Do not store sensitive authentication data after authorization (even if encrypted): Sensitive authentication data includes the data as cited in the following Requirements 1.1.1 through 1.1.3. PCI Data Security Standard Requirement 3.2

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

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

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

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

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

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

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

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

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

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

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

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

2. Protect stored cardholder data

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

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

Notes:

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

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

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

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

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

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

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

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

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

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

3. Provide secure authentication features

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

Note: These password controls are not intended to apply to employees who only have access to one card number at a time to facilitate a single transaction. These controls are applicable for access by employees with administrative capabilities, for access to servers with cardholder data, and for access controlled by the payment application. This requirement applies to the payment application and all associated tools used to view or access cardholder data. 3.1.10 If a payment application session has been idle for more than 15 minutes, the application requires the user to re-authenticate. PCI Data Security Standard Requirement 8.5.15.

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

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

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

4. Log payment application activity

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

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

5. Develop secure payment applications

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

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

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

5.1.2 Separate development/test, and production environments

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

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

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

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

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

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

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

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

- Back-out or product de-installation procedures

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

6. Protect wireless transmissions

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

7. Test payment applications to address vulnerabilities

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

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

8. Facilitate secure network implementation

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

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

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

10. Facilitate secure remote software updates

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

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

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

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

11. Encrypt sensitive traffic over public networks

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

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

12. Encrypt all non-console administrative access

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

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

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

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

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

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

5.5 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 6

NEO2 Main Transaction Commands

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

6.1 EMV Methods

Start EMV Transaction

emv_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback: (IDT_NEO2)

Start CTLS Transaction (EMV or MSD)

IDT_NEO2::ctls_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:()

Start Any Interface Transaction

IDT_NEO2::device_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:()

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

emv_authenticateTransaction: (IDT_NEO2)

For contact EMV only. By default, auto-authenticate is ON and this step does not need to be performed. If auto-authenticate is OFF (emv_disableAutoAuthenticateTransaction: (IDT_NEO2)), when the results come back as EMV_RESULT_CODE.EMV_RESULT_CODE_AUTHENTICATE_TRANSACTION, this method must be called to continue the EMV transaction.

Complete Online EMV Transaction

emv_completeOnlineEMVTransaction:hostResponseTags: (IDT_NEO2)

For contact EMV only. After receiving a host response, pass host tags (minimum 8A Authorization Response Code) as a TLV stream through the tags parameter. EMV tags can be parsed returned pointer.

If there was a communication error with host, you must still finish the EMV transaction by passing "FALSE" for isSuccess, and nil for tags.

Terminal Configuration

emv_retrieveTerminalData: (IDT_NEO2)
emv_removeTerminalData (IDT_NEO2)
emv_setTerminalData: (IDT_NEO2)
ctls_setTerminalData: (IDT_NEO2)
ctls_setConfigurationGroup: (IDT_NEO2)

ctls_removeConfigurationGroup: (IDT_NEO2)
ctls_getConfigurationGroup:response: (IDT_NEO2)

Methods for terminal configuration. When setting the terminal data, you populate and pass and TerminalFile structure.

AID Management

emv_retrieveApplicationData:response: (IDT_NEO2)
emv_removeApplicationData: (IDT_NEO2)
emv_setApplicationData:configData: (IDT_NEO2)
emv_retrieveAIDList: (IDT_NEO2)
ctls_retrieveApplicationData:response: (IDT_NEO2)
ctls_removeApplicationData: (IDT_NEO2)
ctls_setApplicationData: (IDT_NEO2)
ctls_retrieveAIDList: (IDT_NEO2)

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

CAPK Management

emv_retrieveCAPK:index:response: (IDT_NEO2)
emv_removeCAPK:index: (IDT_NEO2)
emv_setCAPK: (IDT_NEO2)
emv_retrieveCAPKList: (IDT_NEO2)
ctls_retrieveCAPK:key: (IDT_NEO2)
ctls_removeCAPK: (IDT_NEO2)
ctls_setCAPK: (IDT_NEO2)
IDT_NEO2::ctsl_retrieveCAPKList:()
Methods for Certificate Authority Public Key management. When setting the CAPK, you populate and pass the key as a sequence of ordered bytes. When specifying a CAPK to retrieve or remove, you populate the name in the NSData parameter. When retrieving the CAPK list, the list of RID/Index can be retrieved from the ordered NSData stream, 6 bytes each, bytes 1-5 RID, byte 6 index.

CRL Management

emv_retrieveCRLList: (IDT_NEO2)
emv_removeCRLList (IDT_NEO2)
emv_setCRLEntries: (IDT_NEO2)

Methods for Certificate Revocation List management.

APDU Communication

device_setPassThrough: (IDT_NEO2)
icc_powerOnICC: (IDT_NEO2)
icc_powerOffICC: (IDT_NEO2)

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

6.2 MSR/CTLS MSD

Request Swipe or Tap

msr_startMSRSwipe (IDT_NEO2)
IDT_NEO2::ctls_startTransaction()
Both methods perform identical operation. Enables MSR to receive Swipe and CTLS to receive tap. If swipe captured, returns IDTMSRData instance to deviceDelegate::swipeMSRData:(). If CTLS captured, returns IDTEMVData to deviceDelegate::emvTransactionData:()

Cancel Swipe

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

Chapter 7

Sending Direct Commands

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

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

device_sendIDGCommand:subCommand:data:response: (IDT_NEO2)

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

Chapter 8

Core Implementation NEO2: Objective-C

IDTech Framework includes class libraries to interface with the NEO2. This guide assume a fair understanding of Xcode 5.0+ and general Apple iOS programming knowledge.

8.1 Integrating with IDTech framework

- Import the necessary framework/libraries
- Add Import statements to utilize frameworks
- Amend the view controller interface
- Implement optional delegate protocols
- Allocate/initialize IDT_NEO2 objects
- Sample Project Tutorial

8.2 Import the necessary framework/libraries

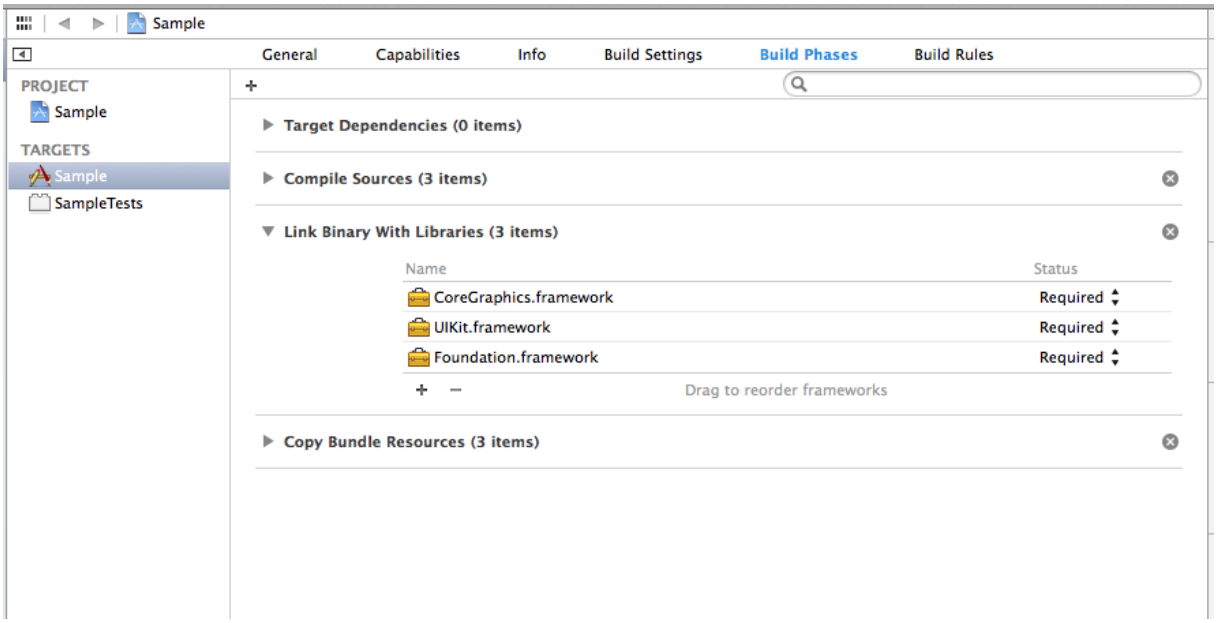
Communicating with IDTech Devices requires the following framework/libraries to be imported into the project:

- IDTech.framework
- ExternalAccessory.framework
- MediaPlayer.framework
- AVFoundation.framework
- AudioToolbox.framework
- CFNetwork.framework
- CoreBluetooth.framework

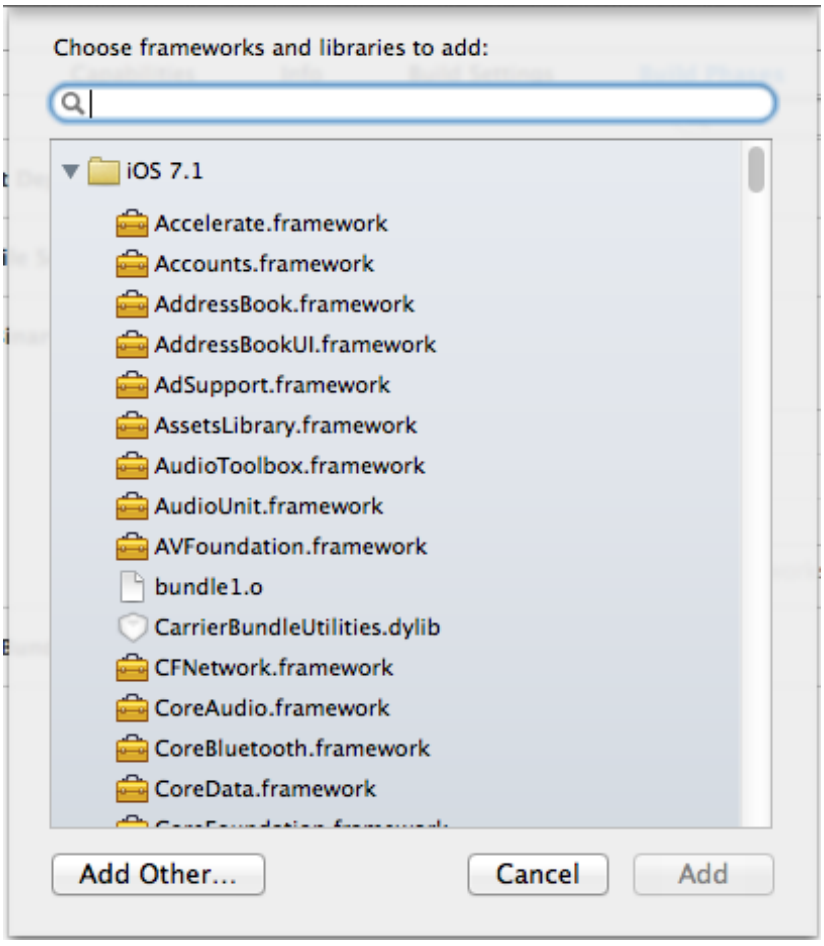
Also, import the following resource bundle:

- IDTech.bundle

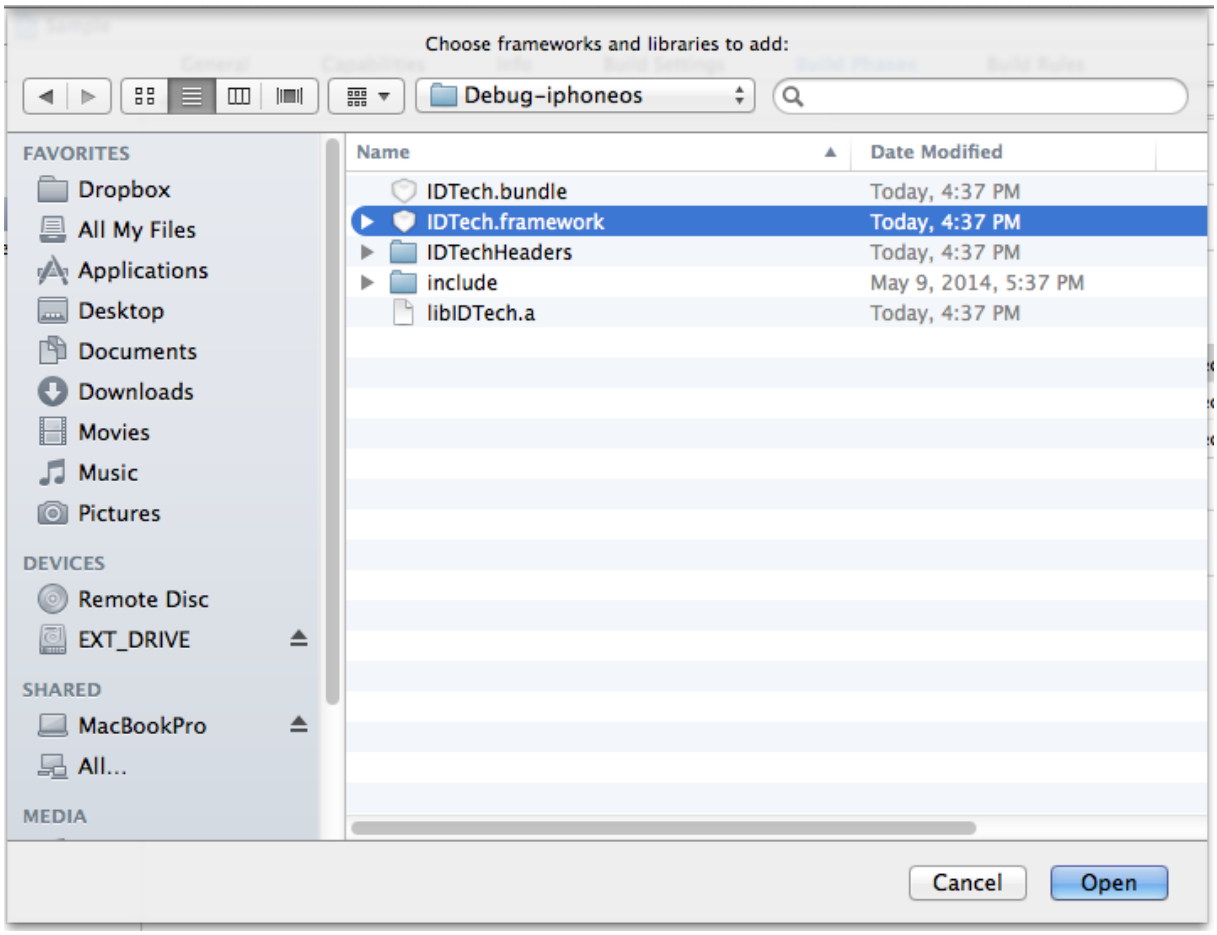
Under Build Phases, select Link Binary With Libraries and click the Add (+) button



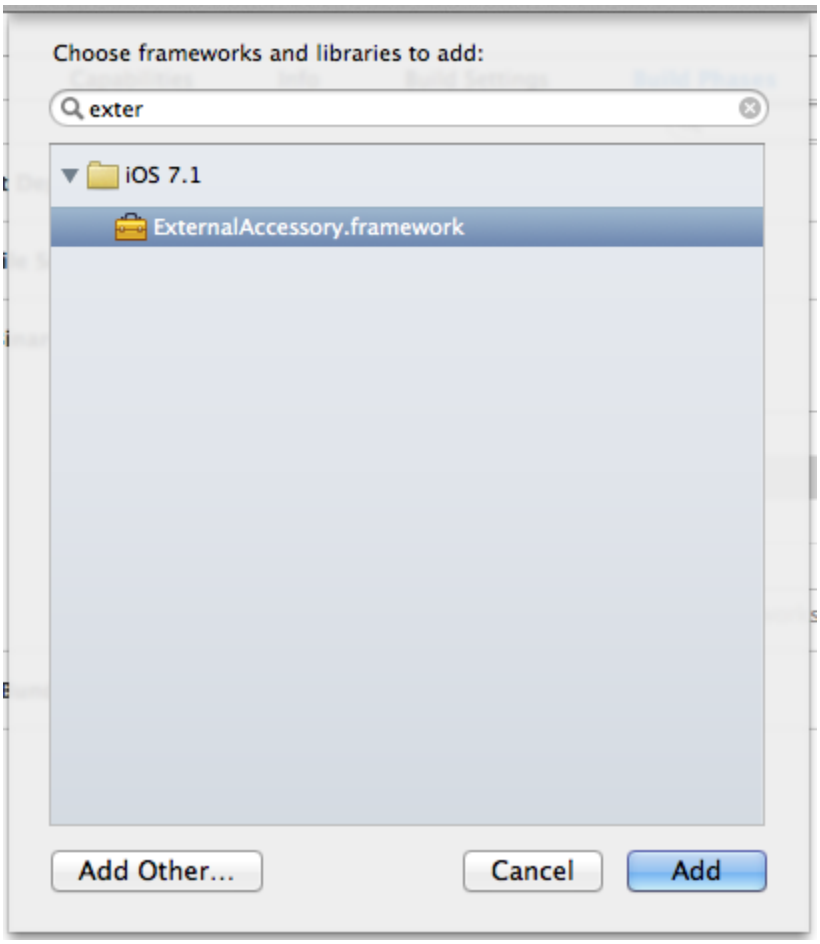
On the Choose Frameworks screen, click "Add Other" in the lower left



Navigate to the IDTech.framework folder, and click "Open"

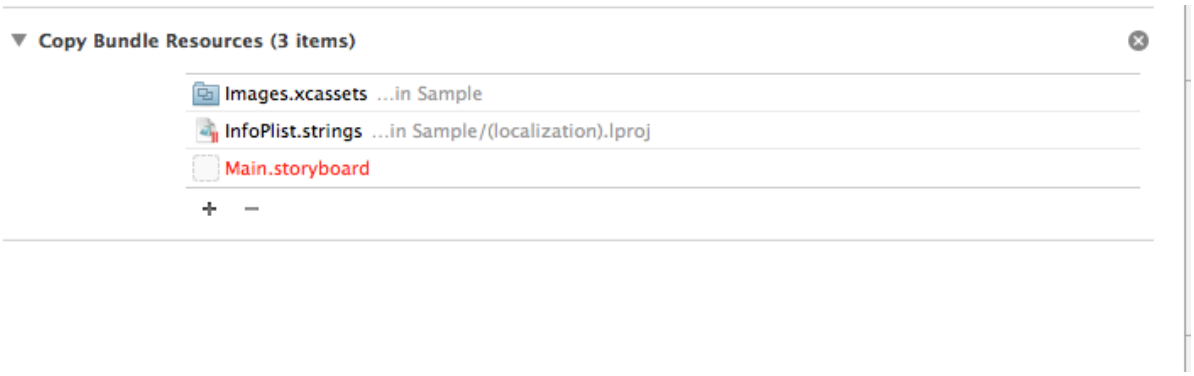


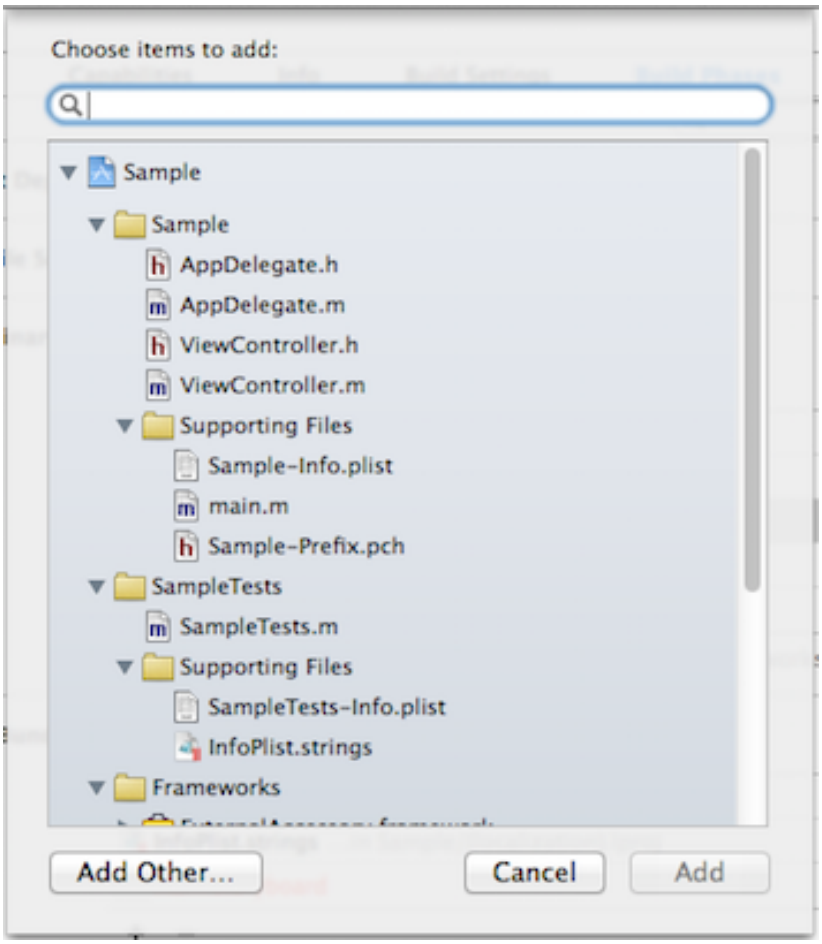
Link the ExternalAccessory framework. On the Choose Frameworks screen, type "exter" into the search bar, select ExternalAccesssory.framework and click "Open"

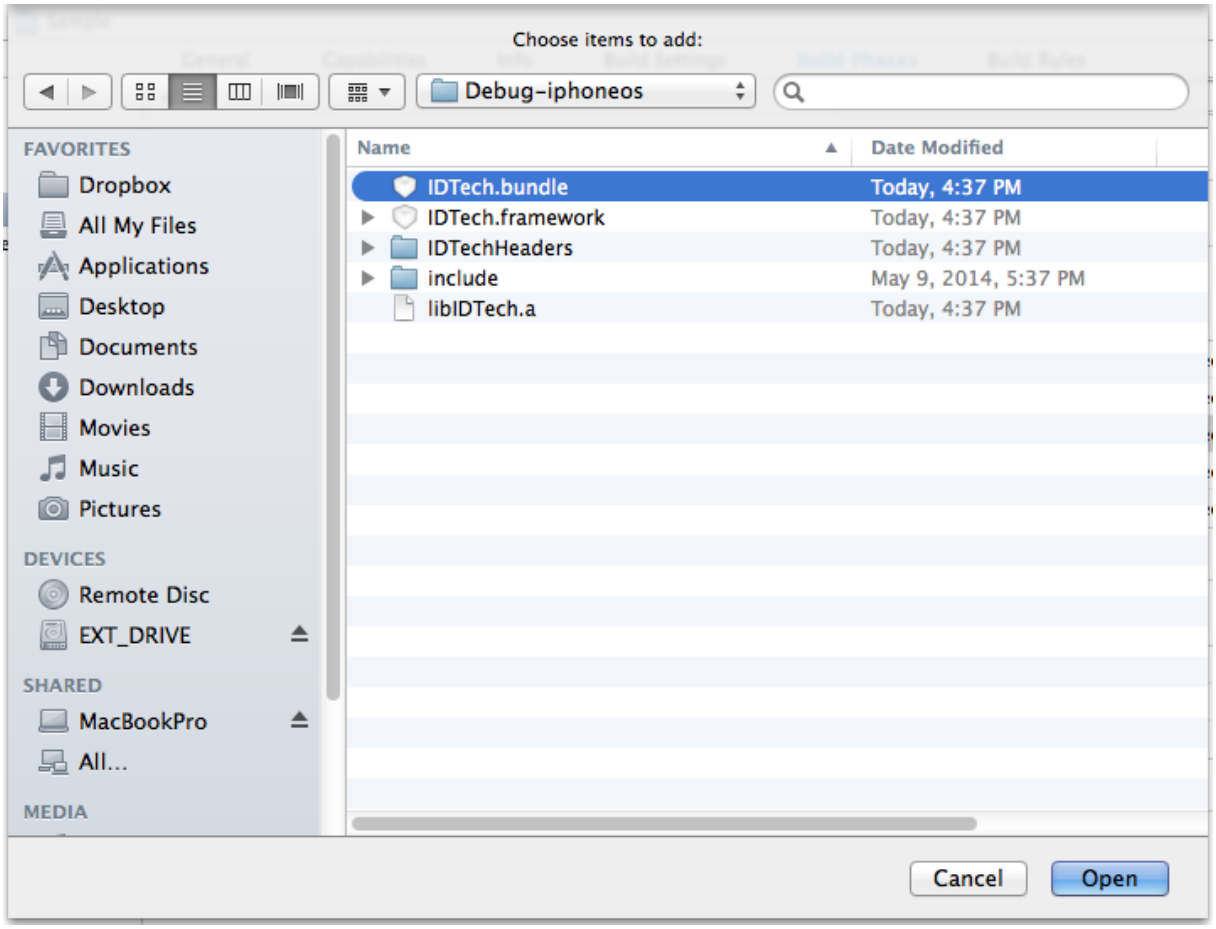


Repeat process for MediaPlayer.framework, AVFoundation.framework, AudioToolbox.framework, and CFNetwork.↩ framework

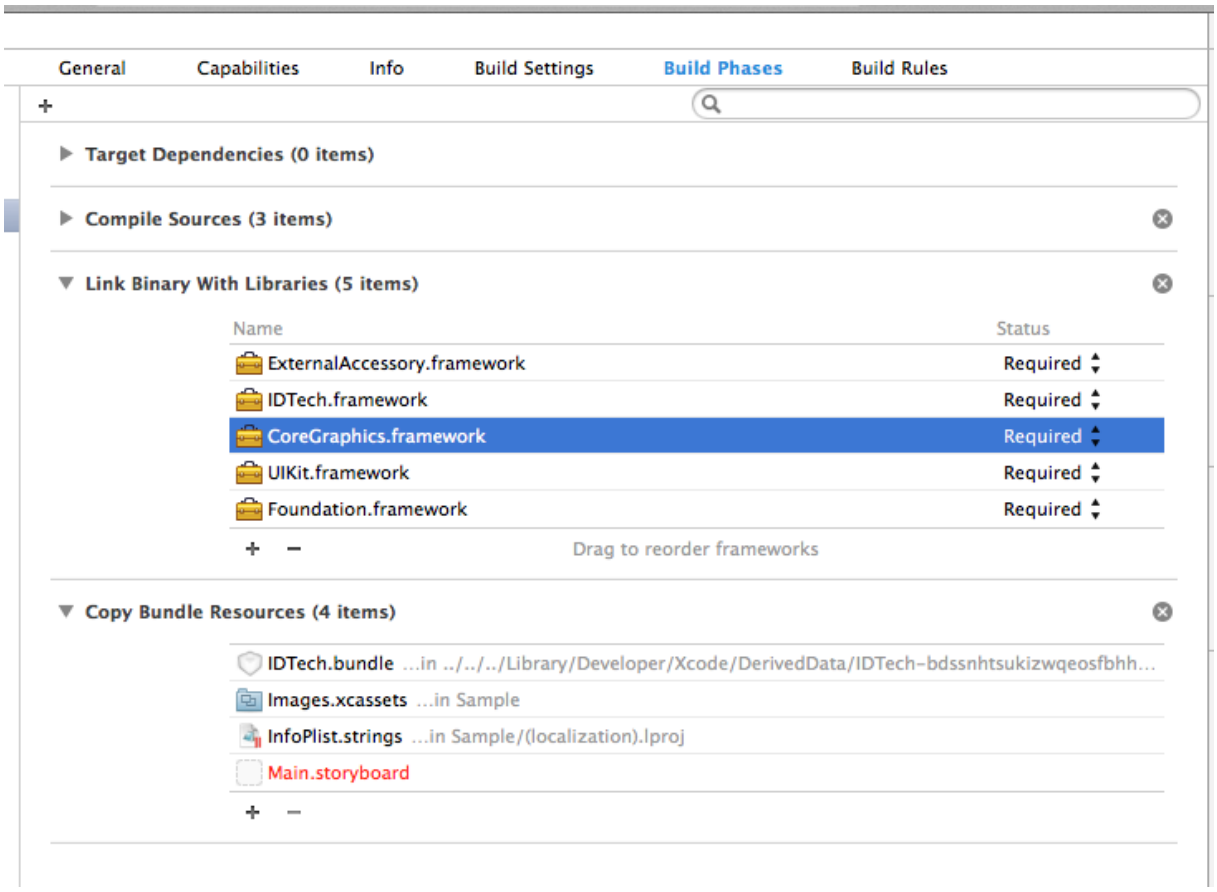
Link another library. Under Copy Bundle, click the Add (+) button, click "Add Other", navigate to and select the IDTech.bundle file and click "Open"




Choose items to add:
Sample
Sample
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Supporting Files
Sample-Info.plist
main.m
Sample-Prefix.pch
SampleTests
SampleTests.m
Supporting Files
SampleTests-Info.plist
InfoPlist.strings
Frameworks
Add Other...
Cancel
Add



The Build Phases should now include the required frameworks/libraries for the NEO2



8.3 Add Import statements to utilize frameworks

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

```
#import <IDTech/IDTech.h>
```

8.4 Amend the view controller interface to include the framework delegate classes:

In the header files of the classes that will be a delegate of IDTech.framework, include the reference to the framework delegate class name:

```
@interface ViewController : UIViewController <IDT_NEO2_Delegate>
```

8.5 Implement any/all of the optional delegate protocols used to receive data from IDT_NEO2_Delegate:

```
- (void) deviceConnected;
- (void) deviceDisconnected;
- (void) dataInOutMonitor:(NSData*)data  incoming:(BOOL)isIncoming;
- (void) swipeMSRData:(IDTMSRData*)cardData;
```

```
- (void) deviceMessage:(NSString*)message;

- (void) lcdDisplay:(int)mode  lines:(NSArray*)lines;

- (void) emvTransactionData:(IDTEMVData*)emvData errorCode:(int)error;

- (void) pinpadData:(NSData*)value keySN:(NSData*)KSN event:(EVENT_PINPAD_Types)event;
```

8.6 Call the Singleton instance of the IDT_NEO2 framework object:

A Singleton instance has been established in the IDT_NEO2 class. To utilize the delegate protocols, best practices would be initialize the connection by setting the delegate with the singleton instance.

To enable scanning for a NEO2 with Bluetooth Low Energy capabilities, you need to execute device_enableBLE↩ DeviceSearch: (IDT_NEO2) identifier:().

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [[IDT_NEO2 sharedController] setDelegate:self];
    [[IDT_NEO2 sharedController] device_enableBLEDeviceSearch:nil];
}
```

8.7 Sample Project Tutorial

Using Xcode 6.4, we will create a sample project that will interface with the NEO2 and will perform the following activities:

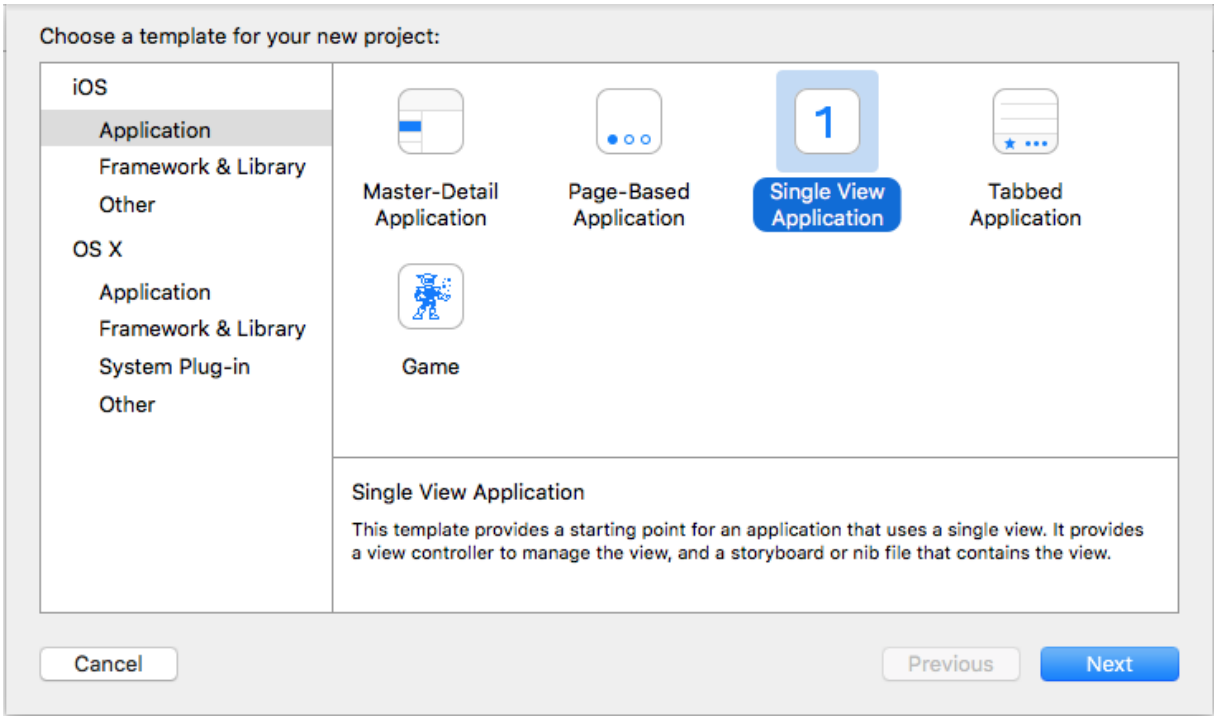
- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for CTLS/MSR/ICC (tap/swipe/insert)
- Enable Bluetooth Scanning
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

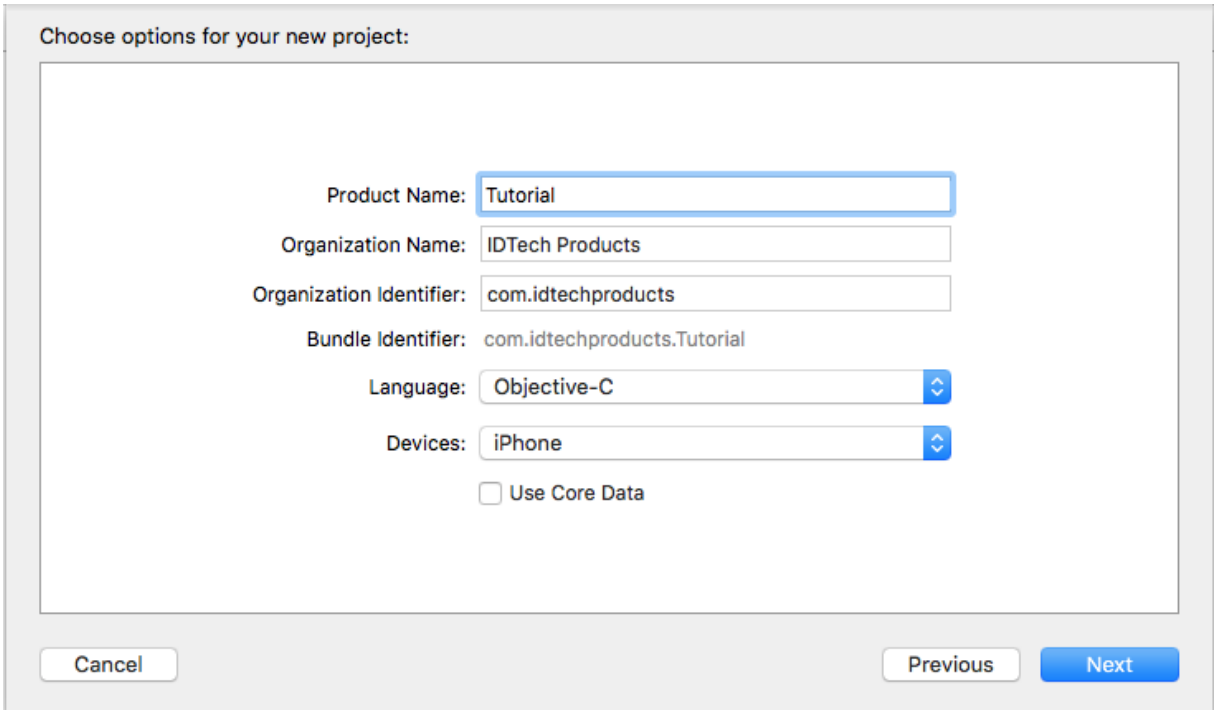
Protocol Delegates:

- Delegate to receive card swipes
- Delegate to detect headphone plug changes
- Delegate to detect device connected
- Delegate to detect device disconnected
- Delegate to receive EMV/CTLS tag data

8.7.1 Step 1: Create New Project

Create a new Single View Application in Xcode





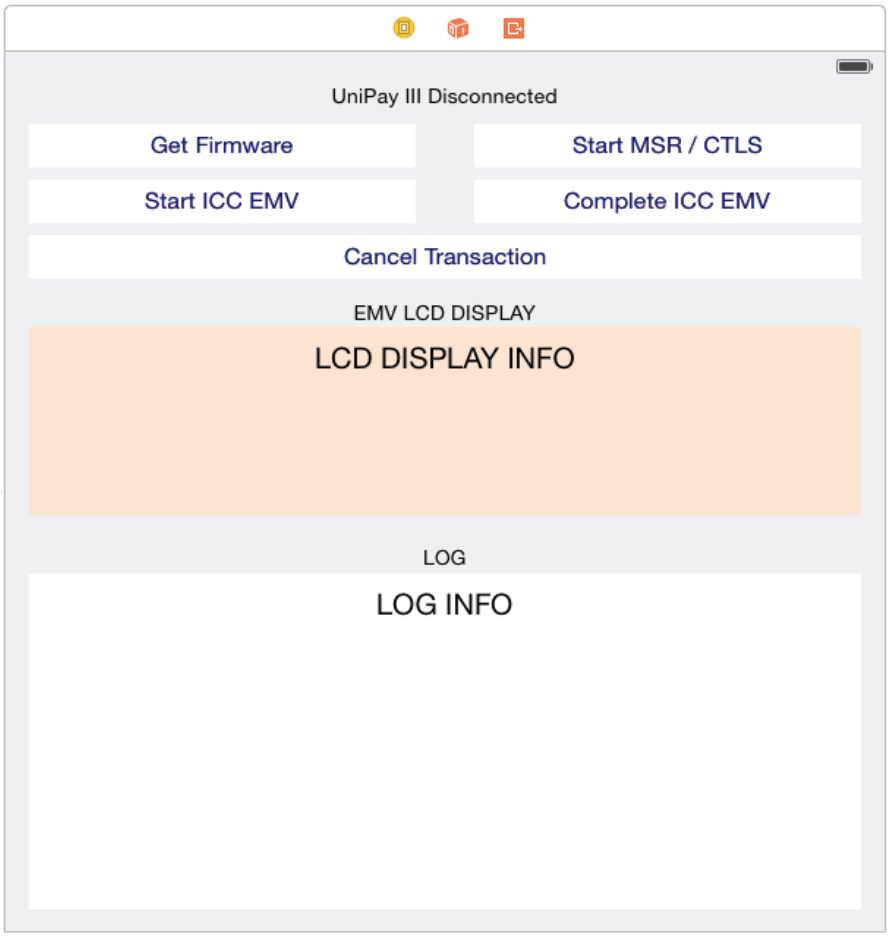
8.7.2 Step 2: Import Frameworks

Import the necessary framework/libraries

8.7.3 Step 3: Design Interface

Design the User Interface by editing the iPhone storyboard file
Open your storyboard and add items to so it contains the following buttons/fields:

- Add a label to the top that will signify connection/disconnection status.
- Add text views to communicate data from the NEO2 and for EMV LCD display information. Remove the Editable behavior if you don't want the keyboard to pop up if you accidentally select it.
- Add buttons to execute the following functions:
  - Get Firmware
  - Scan for BLE Device
  - Start Transaction
  - Complete ICC EMV
  - Cancel Transaction (add constraints accordingly so layout maps to intended screen size)



8.7.4 Step 4: Configure Header File

In the header file, perform the following:

- Add Import statements to utilize frameworks
- Amend the view controller interface
- Create an IBOutlet for the two UITextView and link it as a Referencing Outlet to the UITextView on the storyboard
- Create an IBOutlet for the UILabel and link it as a Referencing Outlet to the UILabel on the storyboard

- Create the 5 IBAction for the buttons, and link them to the "Touch Up Inside" event on the storyboard buttons

```
#import <UIKit/UIKit.h>
#import <IDTech/IDTech.h>

@interface ViewController : UIViewController<IDT_NEO2_Delegate>
{
    IBOutlet UITextView *lcdTextView;
    IBOutlet UITextView *logTextView;
    IBOutlet UILabel *connectedLabel;
}

-(IBAction) getFirmware:(id)sender;
-(IBAction) startBLE:(id)sender;
-(IBAction) startTransaction:(id)sender;
-(IBAction) completeEMV:(id)sender;
-(IBAction) cancelTransaction:(id)sender;


@property(nonatomic, strong) UITextView *lcdTextView;
@property(nonatomic, strong) UITextView *logTextView;
@property(nonatomic, strong) UILabel *connectedLabel;

@end
```

Storyboard Source Code

```
<?xml version="1.0" encoding="UTF-8"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="12121"
      systemVersion="16F73" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES"
      useTraitCollections="YES" colorMatched="YES" initialViewController="vXZ-lx-hvc">
    <device id="retina5_5" orientation="portrait">
        <adaptation id="fullscreen"/>
    </device>
    <dependencies>
        <deployment identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="12089"/>
        <capability name="Constraints to layout margins" minToolsVersion="6.0"/>
        <capability name="documents saved in the Xcode 8 format" minToolsVersion="8.0"/>
    </dependencies>
    <scenes>
        <!--View Controller-->
        <scene sceneID="ufC-wZ-h7g">
            <objects>
                <viewController id="vXZ-lx-hvc" customClass="ViewController" sceneMemberID="viewController"
      >
                    <layoutGuides>
                        <viewControllerLayoutGuide type="top" id="jyV-Pf-zRb"/>
                        <viewControllerLayoutGuide type="bottom" id="2fi-mo-0CV"/>
                    </layoutGuides>
                    <view key="view" contentMode="scaleToFill" id="kh9-bI-dsS">
                        <rect key="frame" x="0.0" y="0.0" width="414" height="736"/>
                        <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="YES"/>
                        <subviews>
                            <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
      horizontalHuggingPriority="251" verticalHuggingPriority="251" text="NEO2 Disconnected" textAlignment="center"
      lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
      translatesAutoresizingMaskIntoConstraints="NO" id="eDI-cL-7k6">
                                <rect key="frame" x="16" y="20" width="378" height="21"/>
                                <constraints>
                                    <constraint firstAttribute="height" constant="21" id="GlQ-61-SfD"/>
                                </constraints>
                                <fontDescription key="fontDescription" type="system" pointSize="14"/>
                                <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
      colorSpace="custom" customColorSpace="sRGB"/>
                                <nil key="highlightedColor"/>
                            </label>
                            <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
      horizontalHuggingPriority="251" verticalHuggingPriority="251" text="EMV LCD DISPLAY" textAlignment="center" lineBreakMo
      ="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
      translatesAutoresizingMaskIntoConstraints="NO" id="oIl-91-J9y">
                                <rect key="frame" x="16" y="227" width="378" height="21"/>
                                <constraints>
                                    <constraint firstAttribute="height" constant="21" id="cli-EH-4sy"/>
                                </constraints>
                                <fontDescription key="fontDescription" type="system" pointSize="14"/>
                                <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
      colorSpace="custom" customColorSpace="sRGB"/>
                                <nil key="highlightedColor"/>
                            </label>
                            <textView clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="
      scaleToFill" editable="NO" text="LCD DISPLAY INFO" textAlignment="center" translatesAutoresizingMaskIntoConstraints="
      NO" id="XfN-t5-Cxv">
                                <rect key="frame" x="16" y="256" width="378" height="128"/>
```

```
                            <color key="backgroundColor" red="0.98823529480000005" green="
0.89411765340000005" blue="0.8156862855" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="128" id="cbD-9q-LWB"/>
                            </constraints>
                            <fontDescription key="fontDescription" name="HelveticaNeue" family="
Helvetica Neue" pointSize="20"/>
                            <textInputTraits key="textInputTraits" autocapitalizationType="sentences"/>
                        </textView>
                        <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" ambiguous="YES" misplaced="YES" text="" lineBreakMode="
tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="BNt-fB-6Ni">
                            <rect key="frame" x="288" y="53" width="24" height="21"/>
                            <fontDescription key="fontDescription" type="system" pointSize="17"/>
                            <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <nil key="highlightedColor"/>
                        </label>
                        <button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="
center" contentVerticalAlignment="center" lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="8zA-5c-Igm">
                            <rect key="frame" x="20" y="53" width="374" height="26"/>
                            <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="26" id="qU1-ou-A1l"/>
                            </constraints>
                            <fontDescription key="fontDescription" type="system" pointSize="14"/>
                            <state key="normal" title="Get Firmware">
                                <color key="titleColor" red="0.068003949080000001" green="
0.072250786509999998" blue="0.44422978940000002" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                                <color key="titleShadowColor" red="0.5" green="0.5" blue="0.5" alpha="1
" colorSpace="custom" customColorSpace="sRGB"/>
                            </state>
                            <connections>
                                <action selector="getFirmware:" destination="vXZ-lx-hvc" eventType="
touchUpInside" id="ONf-1N-HDd"/>
                            </connections>
                        </button>
                        <button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="
center" contentVerticalAlignment="center" lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="QvK-Hw-R0Z">
                            <rect key="frame" x="20" y="87" width="374" height="26"/>
                            <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="26" id="TZO-wC-gU1"/>
                            </constraints>
                            <fontDescription key="fontDescription" type="system" pointSize="14"/>
                            <state key="normal" title="Scan for NEO2 BLE Device">
                                <color key="titleColor" red="0.068003949080000001" green="
0.072250786509999998" blue="0.44422978940000002" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                                <color key="titleShadowColor" red="0.5" green="0.5" blue="0.5" alpha="1
" colorSpace="custom" customColorSpace="sRGB"/>
                            </state>
                            <connections>
                                <action selector="startBLE:" destination="vXZ-lx-hvc" eventType="
touchUpInside" id="gZ8-hp-pgk"/>
                            </connections>
                        </button>
                        <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" fixedFrame="YES" text="" lineBreakMode="tailTruncation"
baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO" id
="c2A-gb-32n">
                            <rect key="frame" x="288" y="91" width="24" height="21"/>
                            <autoresizingMask key="autoresizingMask" flexibleMaxX="YES" flexibleMaxY="
YES"/>
                            <fontDescription key="fontDescription" type="system" pointSize="17"/>
                            <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <nil key="highlightedColor"/>
                        </label>
                        <button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="
center" contentVerticalAlignment="center" lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="KML-Tf-QdE">
                            <rect key="frame" x="20" y="121" width="374" height="26"/>
                            <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="26" id="vhA-Zy-E5Q"/>
                            </constraints>
                            <fontDescription key="fontDescription" type="system" pointSize="14"/>
                            <state key="normal" title="Start Transaction">
                                <color key="titleColor" red="0.068003949080000001" green="
0.072250786509999998" blue="0.44422978940000002" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                                <color key="titleShadowColor" red="0.5" green="0.5" blue="0.5" alpha="1
```

```
" colorSpace="custom" customColorSpace="sRGB"/>
                            </state>
                            <connections>
                                <action selector="startTransaction:" destination="vXZ-lx-hvc" eventType
="touchUpInside" id="btb-Iy-8No"/>
                            </connections>
                        </button>
                        <button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="
center" contentVerticalAlignment="center" lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="U0Y-vt-9An">
                            <rect key="frame" x="20" y="189" width="374" height="26"/>
                            <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="26" id="UoI-vD-ShH"/>
                            </constraints>
                            <fontDescription key="fontDescription" type="system" pointSize="16"/>
                            <state key="normal" title="Cancel Transaction">
                                <color key="titleColor" red="0.068003949080000001" green="
0.072250786509999998" blue="0.44422978940000002" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                                <color key="titleShadowColor" red="0.5" green="0.5" blue="0.5" alpha="1
" colorSpace="custom" customColorSpace="sRGB"/>
                            </state>
                            <connections>
                                <action selector="cancelTransaction:" destination="vXZ-lx-hvc"
eventType="touchUpInside" id="rOr-8n-gJz"/>
                            </connections>
                        </button>
                        <textView clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="
scaleToFill" editable="NO" text="LOG INFO" translatesAutoresizingMaskIntoConstraints="NO" id="Ybs-ph-PwR">
                            <rect key="frame" x="16" y="421" width="378" height="295"/>
                            <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <fontDescription key="fontDescription" name="HelveticaNeue" family="
Helvetica Neue" pointSize="14"/>
                            <textInputTraits key="textInputTraits" autocapitalizationType="sentences"/>
                        </textView>
                        <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" text="LOG" textAlignment="center" lineBreakMode="
tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="ngg-AN-FNd">
                            <rect key="frame" x="16" y="392" width="378" height="21"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="21" id="cH1-gW-ZeD"/>
                            </constraints>
                            <fontDescription key="fontDescription" type="system" pointSize="14"/>
                            <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <nil key="highlightedColor"/>
                        </label>
                        <button opaque="NO" contentMode="scaleToFill" contentHorizontalAlignment="
center" contentVerticalAlignment="center" lineBreakMode="middleTruncation"
translatesAutoresizingMaskIntoConstraints="NO" id="IhF-UM-clE">
                            <rect key="frame" x="20" y="155" width="374" height="26"/>
                            <color key="backgroundColor" red="1" green="1" blue="1" alpha="1"
colorSpace="custom" customColorSpace="sRGB"/>
                            <constraints>
                                <constraint firstAttribute="height" constant="26" id="cio-IH-GQr"/>
                            </constraints>
                            <fontDescription key="fontDescription" type="system" pointSize="14"/>
                            <state key="normal" title="Complete ICC EMV">
                                <color key="titleColor" red="0.068003949080000001" green="
0.072250786509999998" blue="0.44422978940000002" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                                <color key="titleShadowColor" red="0.5" green="0.5" blue="0.5" alpha="1
" colorSpace="custom" customColorSpace="sRGB"/>
                            </state>
                            <connections>
                                <action selector="completeEMV:" destination="vXZ-lx-hvc" eventType="
touchUpInside" id="sBg-6w-7jS"/>
                            </connections>
                        </button>
                    </subviews>
                    <color key="backgroundColor" red="0.93725490199999995" green="0.93725490199999995"
blue="0.95686274510000002" alpha="1" colorSpace="custom" customColorSpace="sRGB"/>
                    <constraints>
                        <constraint firstItem="QvK-Hw-R0Z" firstAttribute="trailing" secondItem="
KML-Tf-QdE" secondAttribute="trailing" id="0cQ-wx-zHB"/>
                        <constraint firstItem="XfN-t5-Cxv" firstAttribute="trailing" secondItem="
ngg-AN-FNd" secondAttribute="trailing" id="1bC-O7-S1P"/>
                        <constraint firstItem="IhF-UM-clE" firstAttribute="trailing" secondItem="
U0Y-vt-9An" secondAttribute="trailing" id="3xN-DP-18G"/>
                        <constraint firstItem="KML-Tf-QdE" firstAttribute="top" secondItem="QvK-Hw-R0Z"
 secondAttribute="bottom" constant="8" symbolic="YES" id="7nF-2H-I4g"/>
                        <constraint firstItem="IhF-UM-clE" firstAttribute="top" secondItem="KML-Tf-QdE"
 secondAttribute="bottom" constant="8" symbolic="YES" id="ATt-74-bel"/>
                        <constraint firstItem="KML-Tf-QdE" firstAttribute="leading" secondItem="
IhF-UM-clE" secondAttribute="leading" id="BWn-nl-dRE"/>
```

```
                        <constraint firstItem="eDI-cL-7k6" firstAttribute="leading" secondItem="
      kh9-bI-dsS" secondAttribute="leadingMargin" constant="-4" id="F6G-60-w5z"/>
                        <constraint firstItem="eDI-cL-7k6" firstAttribute="trailing" secondItem="
      8zA-5c-Igm" secondAttribute="trailing" id="HnZ-Fj-o8y"/>
                        <constraint firstItem="eDI-cL-7k6" firstAttribute="top" secondItem="jyV-Pf-zRb"
       secondAttribute="bottom" id="KzH-zX-FA1"/>
                        <constraint firstItem="8zA-5c-Igm" firstAttribute="top" secondItem="eDI-cL-7k6"
       secondAttribute="bottom" constant="12" id="LHm-Dm-f2p"/>
                        <constraint firstItem="IhF-UM-clE" firstAttribute="leading" secondItem="
      U0Y-vt-9An" secondAttribute="leading" id="MkL-j0-QAp"/>
                        <constraint firstItem="oIl-91-J9y" firstAttribute="trailing" secondItem="
      XfN-t5-Cxv" secondAttribute="trailing" id="Rgp-90-Rn4"/>
                        <constraint firstItem="8zA-5c-Igm" firstAttribute="leading" secondItem="
      QvK-Hw-R0Z" secondAttribute="leading" id="Rvy-qq-Fxo"/>
                        <constraint firstItem="XfN-t5-Cxv" firstAttribute="leading" secondItem="
      ngg-AN-FNd" secondAttribute="leading" id="Yll-yb-55G"/>
                        <constraint firstItem="U0Y-vt-9An" firstAttribute="trailing" secondItem="
      oIl-91-J9y" secondAttribute="trailing" id="bAe-rc-myD"/>
                        <constraint firstItem="8zA-5c-Igm" firstAttribute="trailing" secondItem="
      QvK-Hw-R0Z" secondAttribute="trailing" id="bSV-Fd-Hzd"/>
                        <constraint firstItem="eDI-cL-7k6" firstAttribute="leading" secondItem="
      oIl-91-J9y" secondAttribute="leading" id="bu0-Jx-Q7i"/>
                        <constraint firstItem="8zA-5c-Igm" firstAttribute="top" secondItem="BNt-fB-6Ni"
       secondAttribute="top" id="cvR-tl-cV9"/>
                        <constraint firstItem="oIl-91-J9y" firstAttribute="leading" secondItem="
      XfN-t5-Cxv" secondAttribute="leading" id="d83-ov-vMy"/>
                        <constraint firstItem="Ybs-ph-PwR" firstAttribute="top" secondItem="ngg-AN-FNd"
       secondAttribute="bottom" constant="8" symbolic="YES" id="f9l-5E-xVh"/>
                        <constraint firstItem="U0Y-vt-9An" firstAttribute="top" secondItem="IhF-UM-clE"
       secondAttribute="bottom" constant="8" symbolic="YES" id="gne-Hd-cG8"/>
                        <constraint firstItem="Ybs-ph-PwR" firstAttribute="leading" secondItem="
      ngg-AN-FNd" secondAttribute="leading" id="hIx-P5-n22"/>
                        <constraint firstItem="XfN-t5-Cxv" firstAttribute="top" secondItem="oIl-91-J9y"
       secondAttribute="bottom" constant="8" symbolic="YES" id="jrl-iP-BpW"/>
                        <constraint firstItem="QvK-Hw-R0Z" firstAttribute="top" secondItem="8zA-5c-Igm"
       secondAttribute="bottom" constant="8" symbolic="YES" id="ktf-S5-As7"/>
                        <constraint firstItem="QvK-Hw-R0Z" firstAttribute="leading" secondItem="
      KML-Tf-QdE" secondAttribute="leading" id="lMh-z7-Khl"/>
                        <constraint firstItem="Ybs-ph-PwR" firstAttribute="trailing" secondItem="
      ngg-AN-FNd" secondAttribute="trailing" id="o1j-yC-m9I"/>
                        <constraint firstItem="oIl-91-J9y" firstAttribute="top" secondItem="U0Y-vt-9An"
       secondAttribute="bottom" constant="12" id="oI7-ZT-AiO"/>
                        <constraint firstAttribute="bottom" secondItem="Ybs-ph-PwR" secondAttribute="
      bottom" constant="20" symbolic="YES" id="ptU-ZW-gnV"/>
                        <constraint firstItem="ngg-AN-FNd" firstAttribute="top" secondItem="XfN-t5-Cxv"
       secondAttribute="bottom" constant="8" symbolic="YES" id="qx1-ql-cTJ"/>
                        <constraint firstItem="KML-Tf-QdE" firstAttribute="trailing" secondItem="
      IhF-UM-clE" secondAttribute="trailing" id="rSe-0N-OFU"/>
                        <constraint firstItem="8zA-5c-Igm" firstAttribute="leading" secondItem="
      kh9-bI-dsS" secondAttribute="leadingMargin" id="wwH-eG-SK5"/>
                        <constraint firstItem="eDI-cL-7k6" firstAttribute="trailing" secondItem="
      kh9-bI-dsS" secondAttribute="trailingMargin" id="xVD-uB-XlY"/>
                    </constraints>
                </view>
                <connections>
                    <outlet property="connectedLabel" destination="eDI-cL-7k6" id="EYe-RB-KZq"/>
                    <outlet property="lcdTextView" destination="XfN-t5-Cxv" id="aN5-cG-h6Y"/>
                    <outlet property="logTextView" destination="Ybs-ph-PwR" id="EOn-F5-kzy"/>
                </connections>
            </viewController>
            <placeholder placeholderIdentifier="IBFirstResponder" id="x5A-6p-PRh" sceneMemberID="
      firstResponder"/>
        </objects>
        <point key="canvasLocation" x="24.637681159420293" y="36.684782608695656"/>
      </scene>
    </scenes>
</document>
```

8.7.5 Step 5: Configure Method File

In the header file, perform the following:

- set delegate and initialize IDT_NEO2 singleton object in the `viewDidLoad` method.

  Reference: Call the Singleton instance of the IDT_NEO2 framework object

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [[IDT_NEO2 sharedController] setDelegate:self];

}
```

- Implement protocol delegate IDT_NEO2Delegate::deviceDisconnected() and IDT_NEO2Delegate::deviceConnected() to monitor connect/disconnect events and modify our connection label upon change. Reference: Implement optional delegate protocols

```
-(void) deviceConnected{
    [connectedLabel setText:@"NEO2 CONNECTED"];
}
-(void) deviceDisconnected{
    [connectedLabel setText:@"NEO2 DISCONNECTED"];
}
```

- Implement protocol delegate swipeMSRData:() to receive card swipe data. Reference: Implement optional delegate protocols

```
-(void) appendMessageToResults:(NSString*) message{
    [logTextView setText:[NSString stringWithFormat:@"%@\n====================\n%@",message, self.
      logTextView.text]];

}

- (void) swipeMSRData:(IDTMSRData*)cardData{
    NSLog(@"--MSR event Received, Type: %d, data: %@", cardData.event, cardData.encTrack1);
    switch (cardData.event) {
        case EVENT_MSR_CARD_DATA:
        {
            switch (cardData.captureEncodeType) {
                case CAPTURE_ENCODE_TYPE_ISOABA:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"ISO/ABA"]
      ];
                    break;
                case CAPTURE_ENCODE_TYPE_AAMVA:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"AA/MVA"]]
      ;
                    break;

                case CAPTURE_ENCODE_TYPE_Other:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"Other"]];
                    break;

                case CAPTURE_ENCODE_TYPE_Raw:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"Raw"]];
                    break;

                case CAPTURE_ENCODE_TYPE_JIS_I:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"
      CAPTURE_ENCODE_TYPE_JIS_I"]];
                    break;

                case CAPTURE_ENCODE_TYPE_JIS_II:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"
      CAPTURE_ENCODE_TYPE_JIS_II"]];
                    break;

                default:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %@", @"UNKNOWN"]
      ];

                    break;
            }
            switch (cardData.captureEncryptType) {
                case CAPTURE_ENCRYPT_TYPE_AES:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %@", @"AES"]];
                    break;
                case CAPTURE_ENCRYPT_TYPE_TDES:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %@", @"TDES"]];
                    break;
                case CAPTURE_ENCRYPT_TYPE_NO_ENCRYPTION:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %@", @"NONE"]];
                    break;


                default:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %@", @"UNKNOWN"
      ]];

                    break;
            }

            [self appendMessageToResults:[NSString stringWithFormat:@"Full card data: %@", cardData.
      cardData]];
            [self appendMessageToResults:[NSString stringWithFormat:@"Track 1: %@", cardData.track1]];
            [self appendMessageToResults:[NSString stringWithFormat:@"Track 2: %@", cardData.track2]];
            [self appendMessageToResults:[NSString stringWithFormat:@"Track 3: %@", cardData.track3]];
```

```
      [self appendMessageToResults:[NSString stringWithFormat:@"Length Track 1: %i", cardData.
track1Length]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Length Track 2: %i", cardData.
track2Length]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Length Track 3: %i", cardData.
track3Length]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Encoded Track 1: %@", cardData.
encTrack1.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Encoded Track 2: %@", cardData.
encTrack2.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Encoded Track 3: %@", cardData.
encTrack3.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Hash Track 1: %@", cardData.
hashTrack1.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Hash Track 2: %@", cardData.
hashTrack2.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"Hash Track 3: %@", cardData.
hashTrack3.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"KSN: %@", cardData.KSN.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"\nSessionID: %@",  cardData.sessionID
.description]];
      [self appendMessageToResults:[NSString stringWithFormat:@"\nReader Serial Number: %@",
cardData.RSN]];
      [self appendMessageToResults:[NSString stringWithFormat:@"\nRead Status: %2X",  cardData.
readStatus]];
      if (cardData.unencryptedTags != nil) [self appendMessageToResults:[NSString stringWithFormat:@"
Unencrytped Tags: %@", cardData.unencryptedTags.description]];
      if (cardData.encryptedTags != nil) [self appendMessageToResults:[NSString stringWithFormat:@"
Encrypted Tags: %@", cardData.encryptedTags.description]];
      if (cardData.maskedTags != nil) [self appendMessageToResults:[NSString stringWithFormat:@"
Masked Tags: %@", cardData.maskedTags.description]];

      NSLog(@"Track 1: %@", cardData.track1);
      NSLog(@"Track 2: %@", cardData.track2);
      NSLog(@"Track 3: %@", cardData.track3);
      NSLog(@"Encoded Track 1: %@", cardData.encTrack1.description);
      NSLog(@"Encoded Track 2: %@", cardData.encTrack2.description);
      NSLog(@"Encoded Track 3: %@", cardData.encTrack3.description);
      NSLog(@"Hash Track 1: %@", cardData.hashTrack1.description);
      NSLog(@"Hash Track 2: %@", cardData.hashTrack2.description);
      NSLog(@"Hash Track 3: %@", cardData.hashTrack3.description);
      NSLog(@"SessionID: %@", cardData.sessionID.description);
      NSLog(@"nReader Serial Number: %@", cardData.RSN);
      NSLog(@"Read Status: %2X", cardData.readStatus);
      NSLog(@"KSN: %@", cardData.KSN.description);

      return;
  }
      break;

  case EVENT_MSR_CANCEL_KEY:
  {
      [self appendMessageToResults:[NSString stringWithFormat:@"(Event) MSR Cancel Key received: %@",
 cardData.encTrack1]];
      return;
  }
      break;

  case EVENT_MSR_BACKSPACE_KEY:
  {
      [self appendMessageToResults:[NSString stringWithFormat:@"(Event) MSR Backspace Key received:
 %@", cardData.encTrack1]];
      return;
  }
      break;

  case EVENT_MSR_ENTER_KEY:
  {
      [self appendMessageToResults:[NSString stringWithFormat:@"(Event) MSR Enter Key received: %@",
cardData.encTrack1]];
      return;
  }
      break;

  case EVENT_MSR_UNKNOWN:
  {
      [self appendMessageToResults:[NSString stringWithFormat:@"(Event) MSR unknown event, data: %@",
 cardData.encTrack1]];
      return;
  }
      break;
  case EVENT_MSR_TIMEOUT:
  {
      [self appendMessageToResults:@"(Event) MSR TIMEOUT"];
      return;
  }
  default:
```

```
            break;
    }
}
```

- Implement protocol delegate emvTransactionData:() to report EMV transaction results

```
- (void) emvTransactionData:(IDTEMVData*)emvData errorCode:(int)error{
    NSLog(@"EMV_RESULT_CODE_V2_response = %2X",error);

    [self appendMessageToResults:[NSString stringWithFormat:@"EMV_RESULT_CODE_V2_response = %2X",error]];
    if (emvData == nil) {
        [self appendMessageToResults:[NSString stringWithFormat:@"EMV TRANSACTION ERROR. Refer to
       EMV_RESULT_CODE_V2_response = 0x%2X",error]];
        return;
    }

    if (emvData.resultCodeV2 == EMV_RESULT_CODE_V2_GO_ONLINE) {
        [self appendMessageToResults:@"ONLINE REQUEST"];
    }
    if (emvData.resultCodeV2 == EMV_RESULT_CODE_V2_APPROVED || emvData.resultCodeV2 ==
      EMV_RESULT_CODE_V2_APPROVED_OFFLINE ) {
        [self appendMessageToResults:@"APPROVED"];
    }
    if (emvData.resultCodeV2 == EMV_RESULT_CODE_V2_MSR_SUCCESS) {
        [self appendMessageToResults:@"MSR Data Captured"];
    }

    if (emvData.cardType == 0) {
        [self appendMessageToResults:@"CONTACT"];
    }
    if (emvData.cardType == 1) {
        [self appendMessageToResults:@"CONTACTLESS"];
    }

    if (emvData.unencryptedTags != nil) [self appendMessageToResults:[NSString stringWithFormat:@"
      Unencrytped Tags: %@", emvData.unencryptedTags.description]];
    if (emvData.encryptedTags != nil) [self appendMessageToResults:[NSString stringWithFormat:@"Encrypted
       Tags: %@", emvData.encryptedTags.description]];
    if (emvData.maskedTags != nil) [self appendMessageToResults:[NSString stringWithFormat:@"Masked Tags:
       %@", emvData.maskedTags.description]];
}
```

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

```
- (void) lcdDisplay:(int)mode  lines:(NSArray*)lines{
    NSMutableString* str = [NSMutableString new];
    if (lines != nil) {
        for (NSString* s in lines) {
            [str appendString:s];
            [str appendString:@"\n"];
        }
    }

    switch (mode) {
        case 0x10:
            //clear screen
            lcdTextView.text = @"";
            break;
        case 0x03:
            lcdTextView.text = str;
            break;
        case 0x01:
        case 0x02:
        case 0x08:{
            [[IDT_NEO2 sharedController] emv_callbackResponseLCD:mode selection:(unsigned char)1];;
        }

            break;
        default:
            break;
    }
}
```

- Implement the button press methods

```
-(IBAction) getFirmware:(id)sender{
    NSString *result;
    logTextView.text = @"";
    RETURN_CODE rt = [[IDT_NEO2 sharedController]  device_getFirmwareVersion:&result];
```

```
    if (RETURN_CODE_DO_SUCCESS == rt)
    {
        [self appendMessageToResults: [NSString stringWithFormat:@"Get FM info:  %@", result]];
    }
}
-(IBAction) startBLE:(id)sender{
    [self appendMessageToResults: @"Turning on BLE scanning"];
    [[IDT_NEO2 sharedController] device_enableBLEDeviceSearch:nil];
}
-(IBAction) startTransaction:(id)sender{
    logTextView.text = @"";
    RETURN_CODE rt = [[IDT_NEO2 sharedController] device_startTransaction:1.00 amtOther:0 type:0
      timeout:60 tags:nil forceOnline:false  fallback:true];
    if (RETURN_CODE_DO_SUCCESS == rt)
    {
        [self appendMessageToResults: @"Start Transaction Command Accepted. Please Swipe/Insert/Tap"];
    }
}
-(IBAction) completeEMV:(id)sender{
    logTextView.text = @"";
    RETURN_CODE rt = [[IDT_NEO2 sharedController]  emv_completeOnlineEMVTransaction:true
      hostResponseTags:[IDTUtility hexToData:@"8A023030"] returnTags:nil];
    if (RETURN_CODE_DO_SUCCESS == rt)
    {
        [self appendMessageToResults: @"Complete Transaction Command Accepted"];
    }
}
-(IBAction) cancelTransaction:(id)sender{
    logTextView.text = @"";
    RETURN_CODE rt = [[IDT_NEO2 sharedController] ctls_cancelTransaction];
    if(RETURN_CODE_DO_SUCCESS == rt){
        [self appendMessageToResults:@"CancelMSR/CTLS: OK."];
    }
}
```

Chapter 9

Core Implementation NEO2: Swift

IDTech Framework includes class libraries to interface with the NEO2. This guide assume a fair understanding of Xcode 7.3+ and general Apple iOS programming knowledge.

9.1 Integrating with IDTech framework

- Import the Necessary Framework/Libraries
- Create a Bridging Header File
- Add Import Statement to the Bridging Header File
- Amend the View Controller Interface
- Implement Optional Delegate Protocols
- Allocate/Initialize IDT_NEO2 Object
- Sample Project Tutorial

9.2 Import the Necessary Framework/Libraries

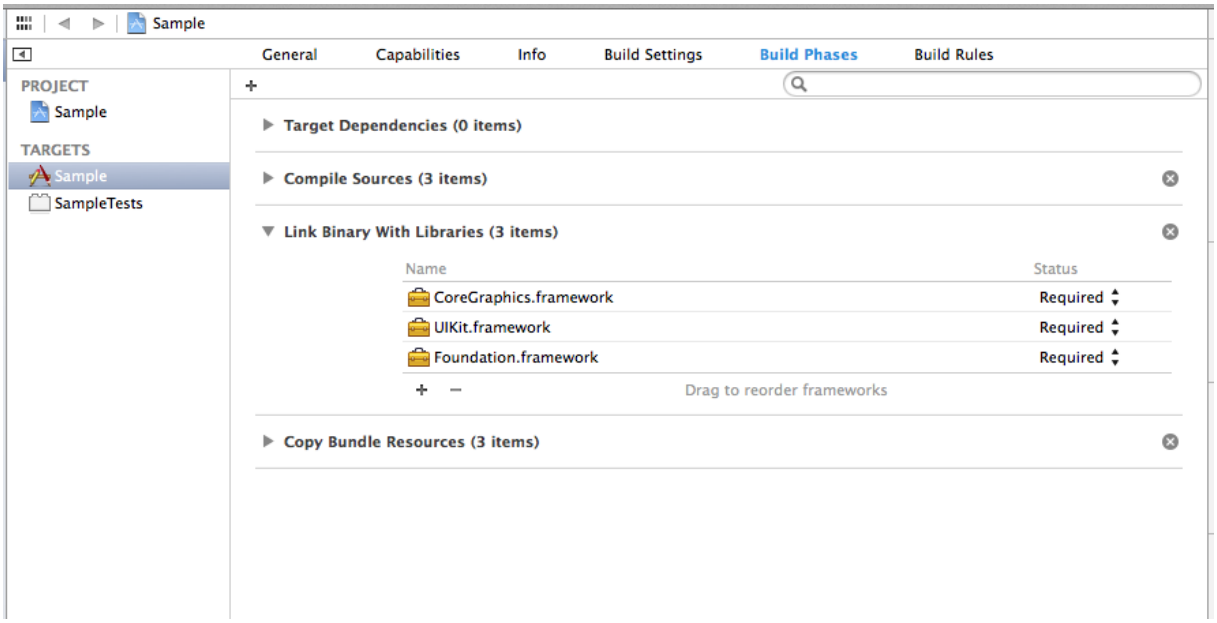
Communicating with IDTech Devices requires the following framework/libraries to be imported into the project:

- IDTech.framework
- ExternalAccessory.framework
- MediaPlayer.framework
- AVFoundation.framework
- AudioToolbox.framework
- CFNetwork.framework

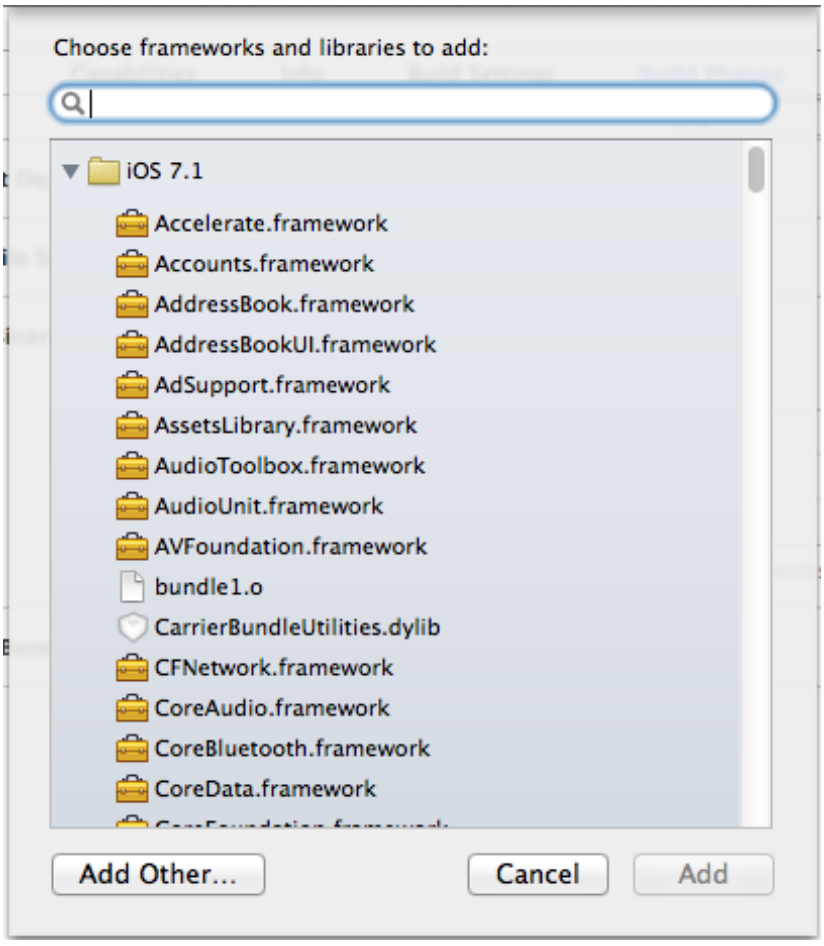
Also, import the following resource bundle:

- IDTech.bundle

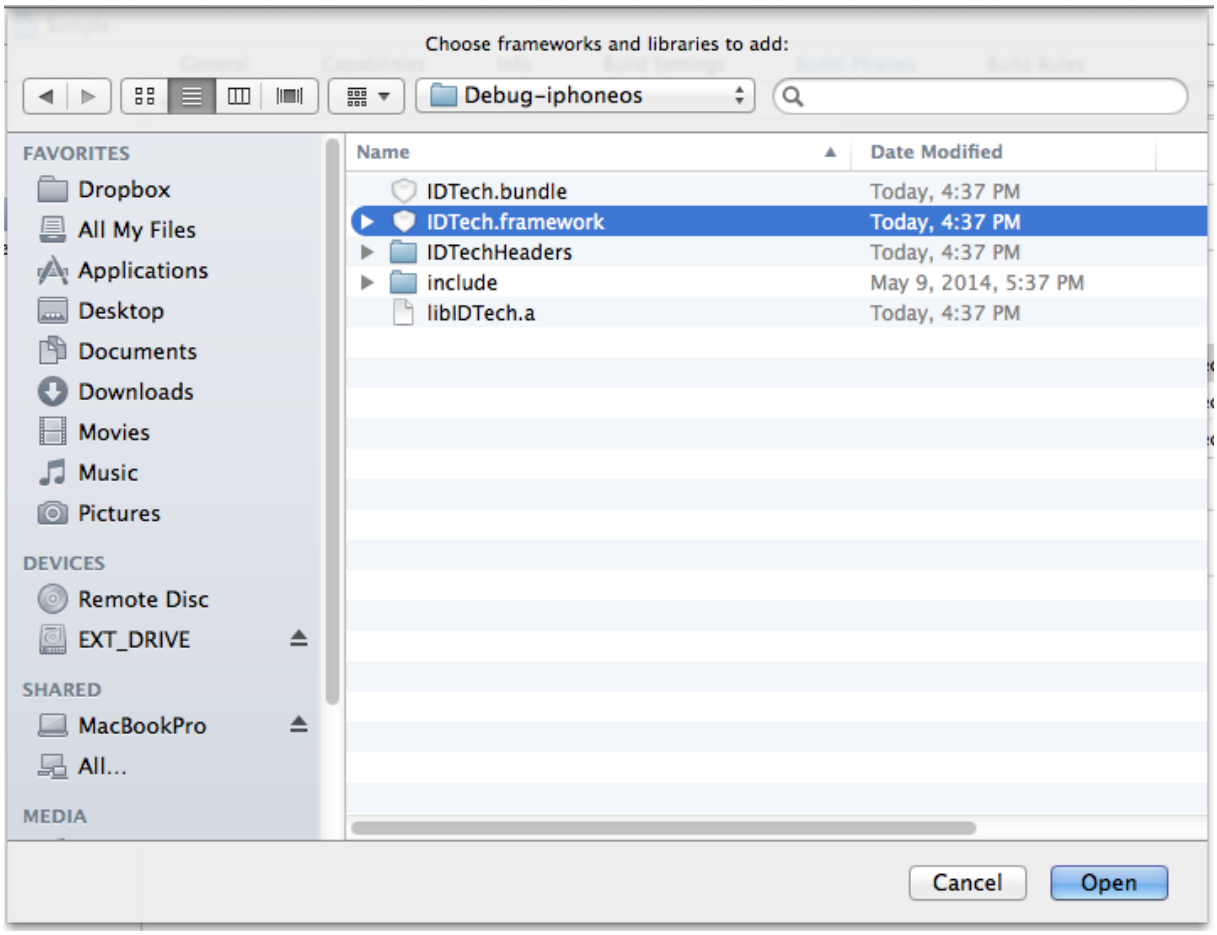
Under Build Phases, select Link Binary With Libraries and click the Add (+) button



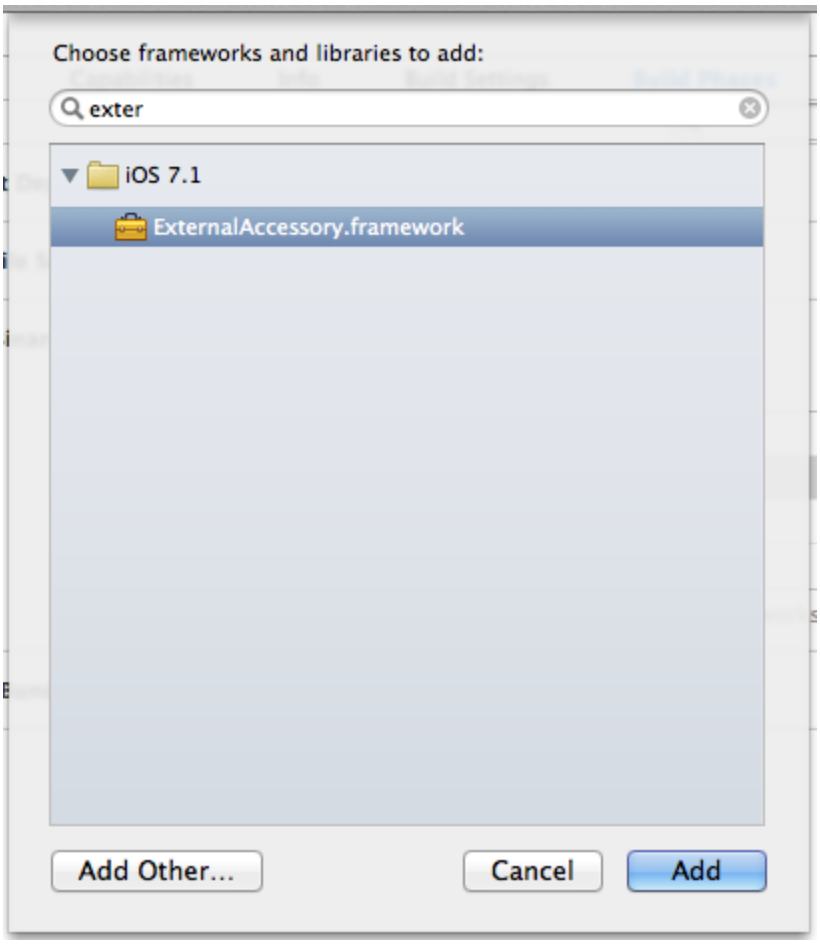
On the Choose Frameworks screen, click "Add Other" in the lower left



Navigate to the IDTech.framework folder, and click "Open"

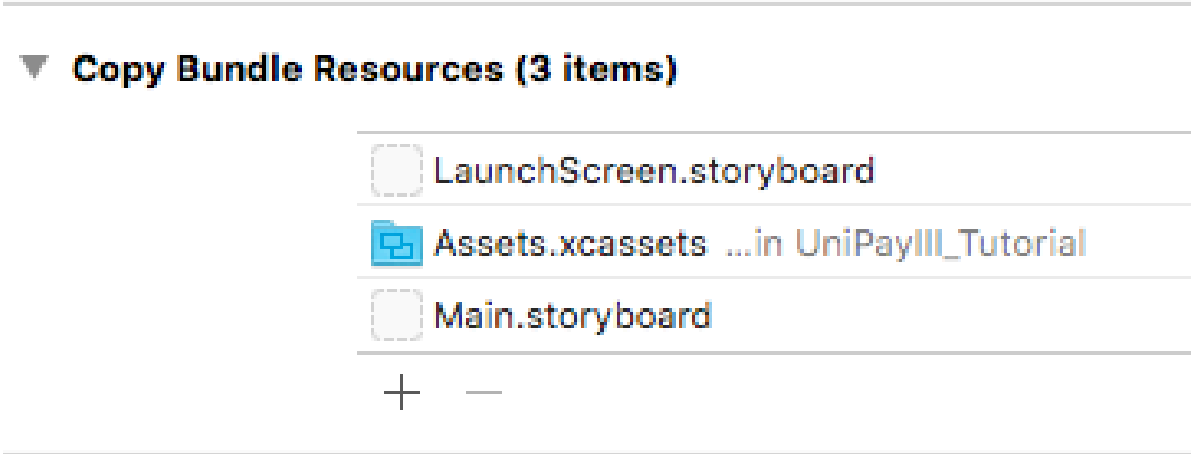


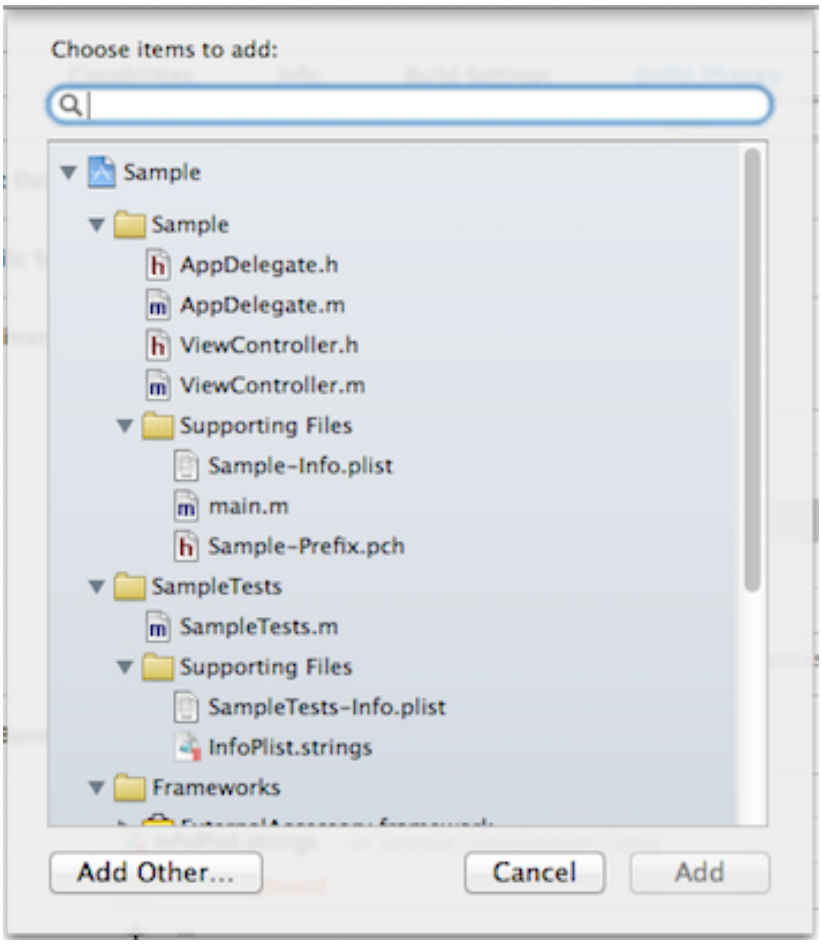
Link the ExternalAccessory framework. On the Choose Frameworks screen, type "exter" into the search bar, select ExternalAccesssory.framework and click "Add"

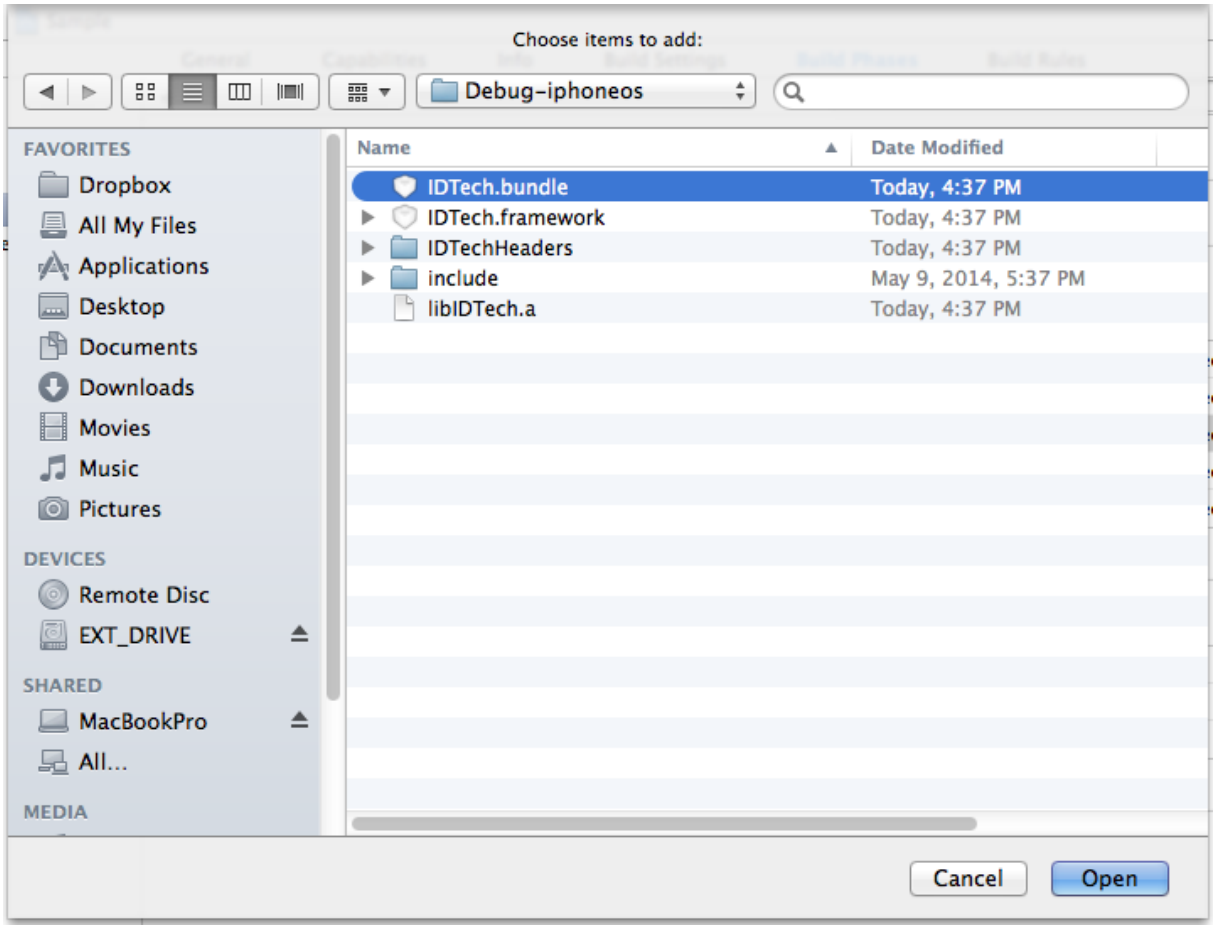


Repeat process for MediaPlayer.framework, AVFoundation.framework, AudioToolbox.framework, and CFNetwork.↩framework

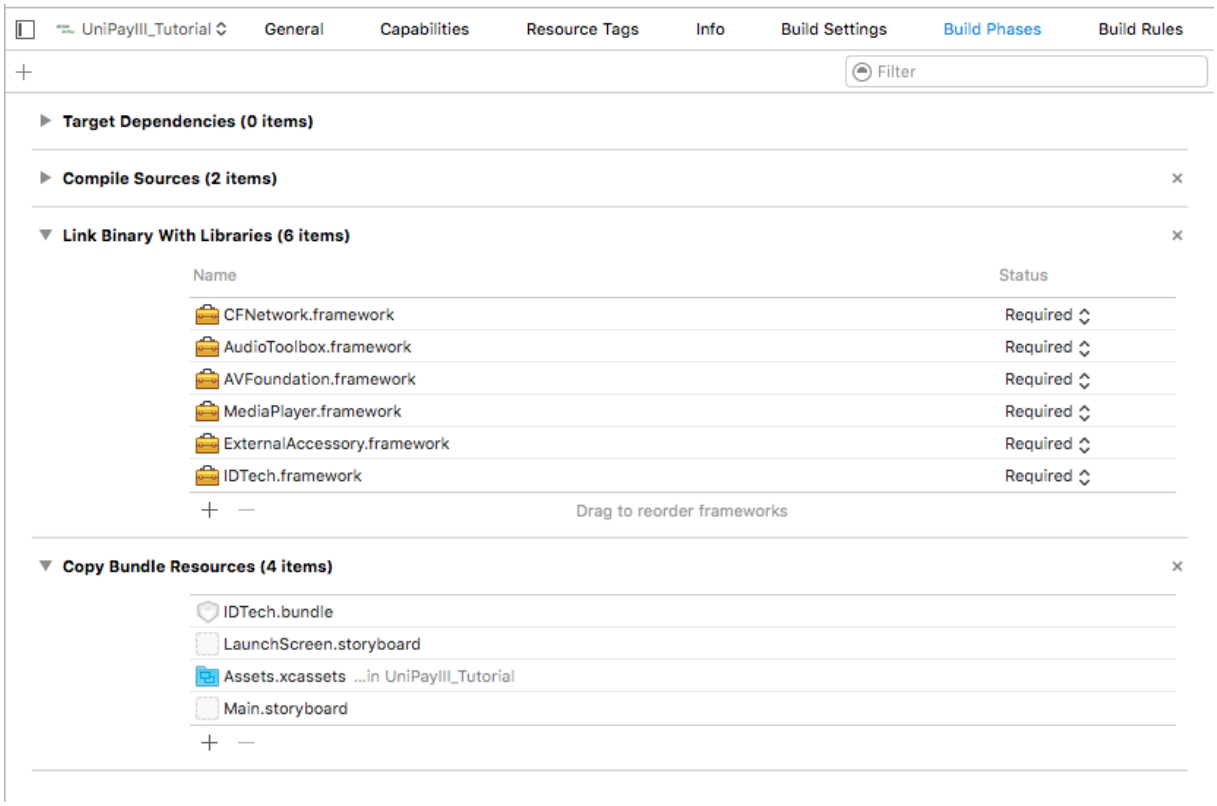
Link another library. Under Copy Bundle, click the Add (+) button, click "Add Other", navigate to and select the IDTech.bundle file and click "Open"




Choose items to add:
Sample
Sample
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Supporting Files
Sample-Info.plist
main.m
Sample-Prefix.pch
SampleTests
SampleTests.m
Supporting Files
SampleTests-Info.plist
InfoPlist.strings
Frameworks
Add Other...
Cancel
Add



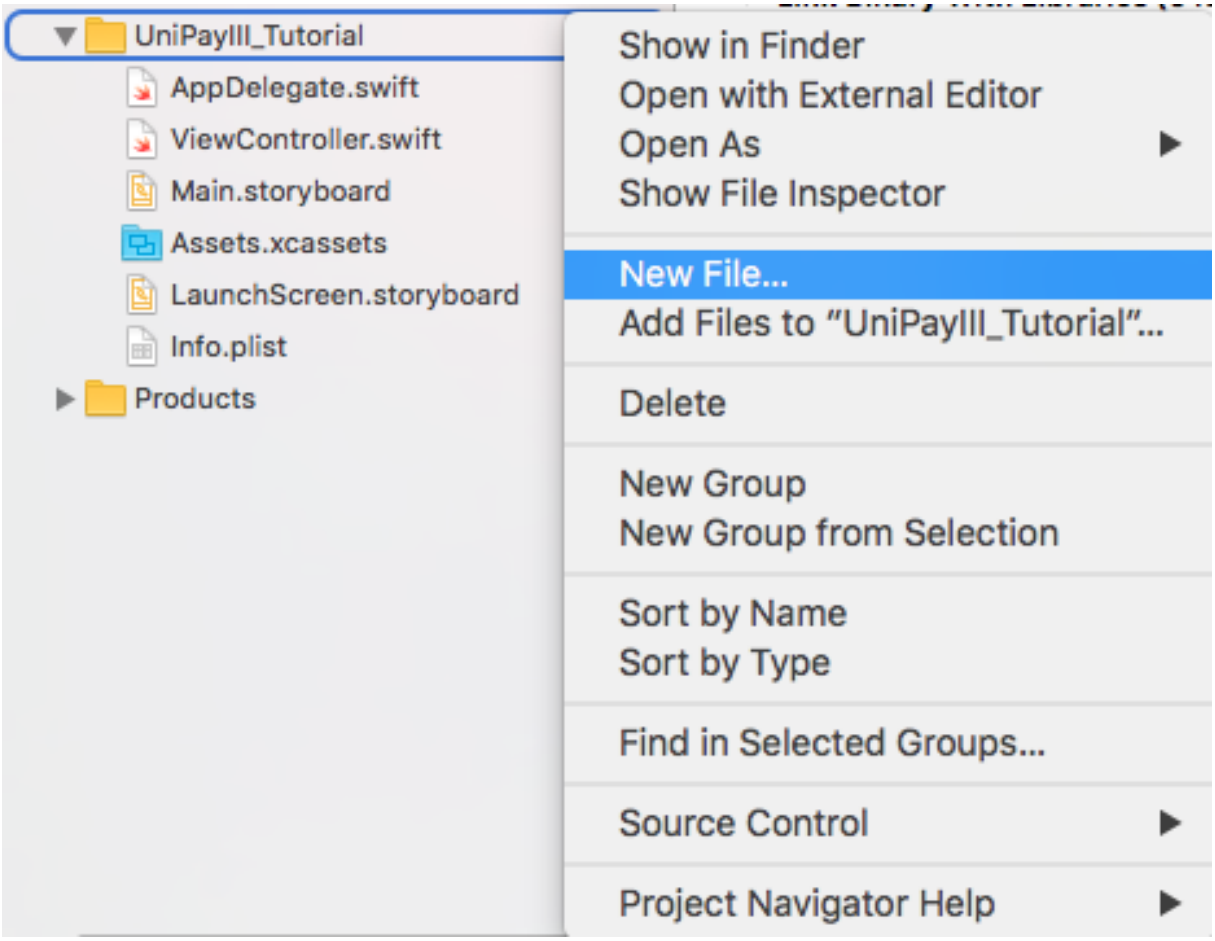
The Build Phases should now include the required frameworks/libraries for the NEO2



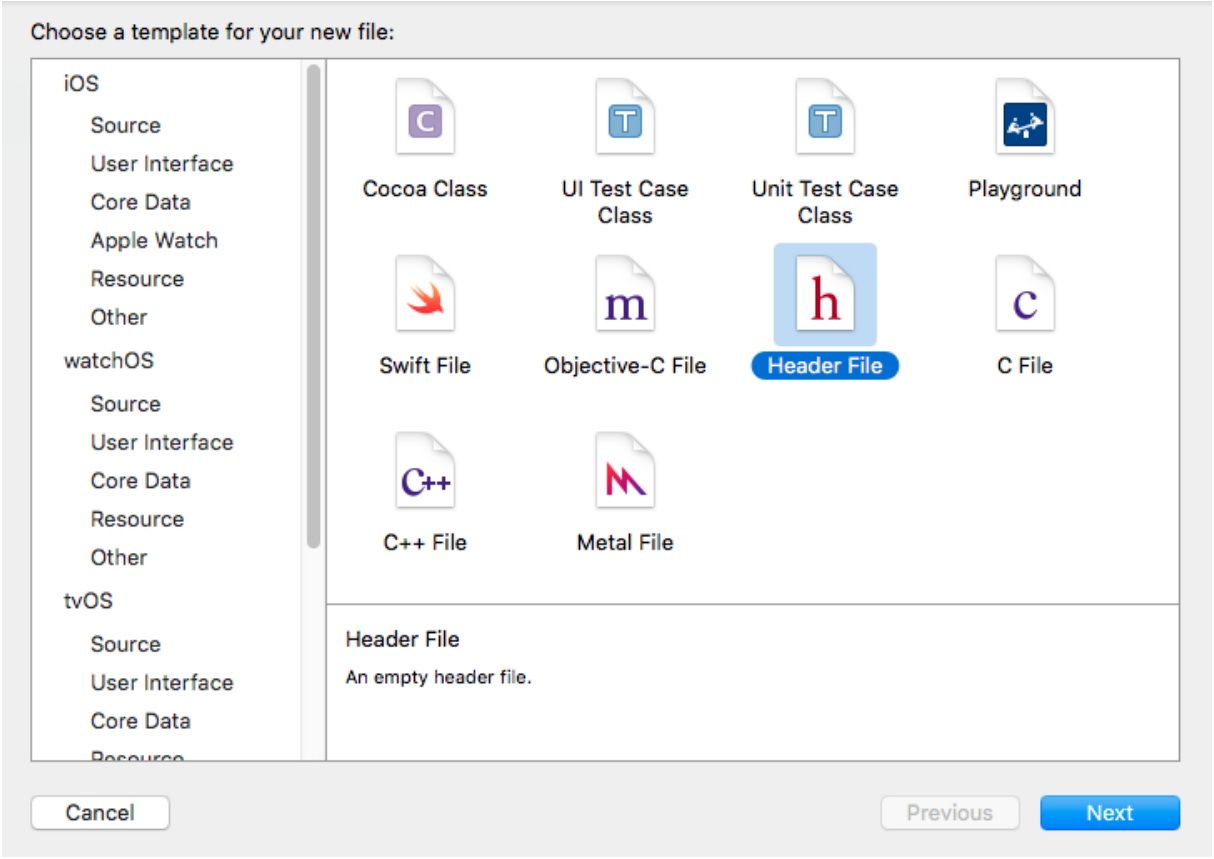
9.3 Create a Bridging Header File

Using the NEO2 SDK with Swift requires a bridging header to allow the application to have a mixed-language codebase

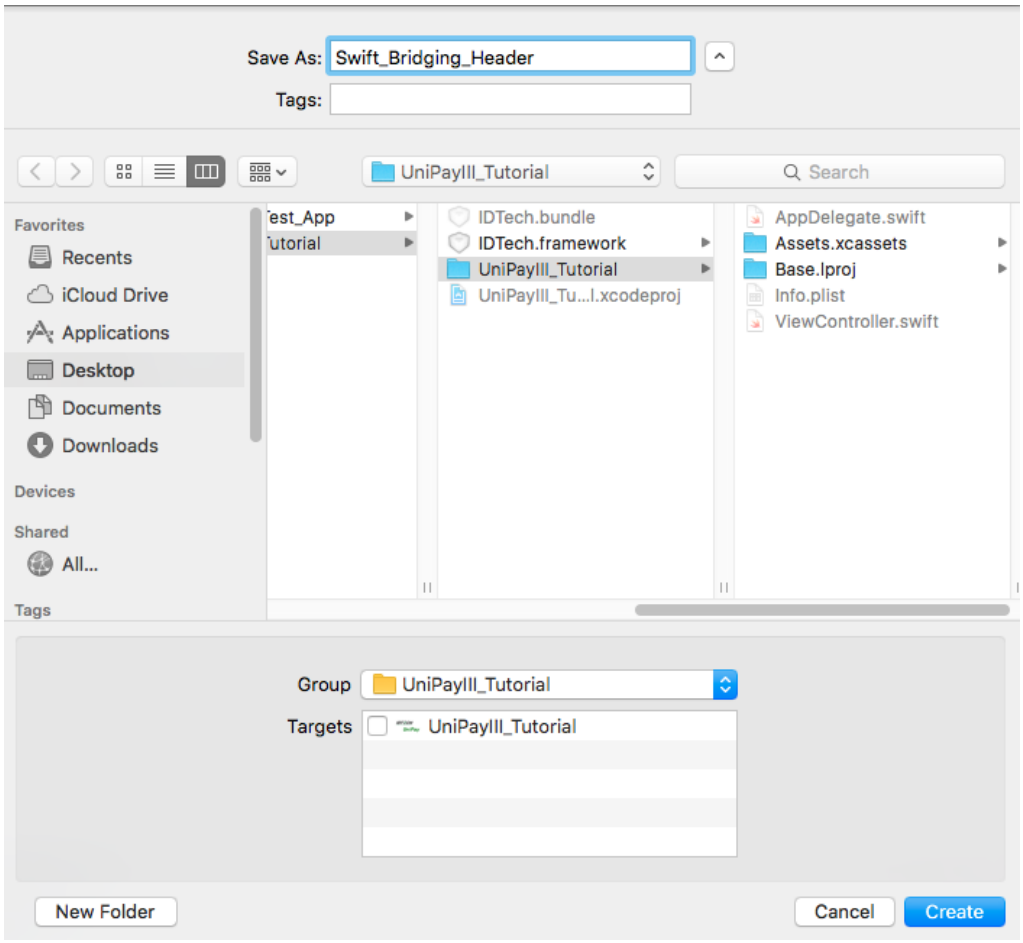
First, you will need to create a header file. Right click on your project directory and click "New File...".



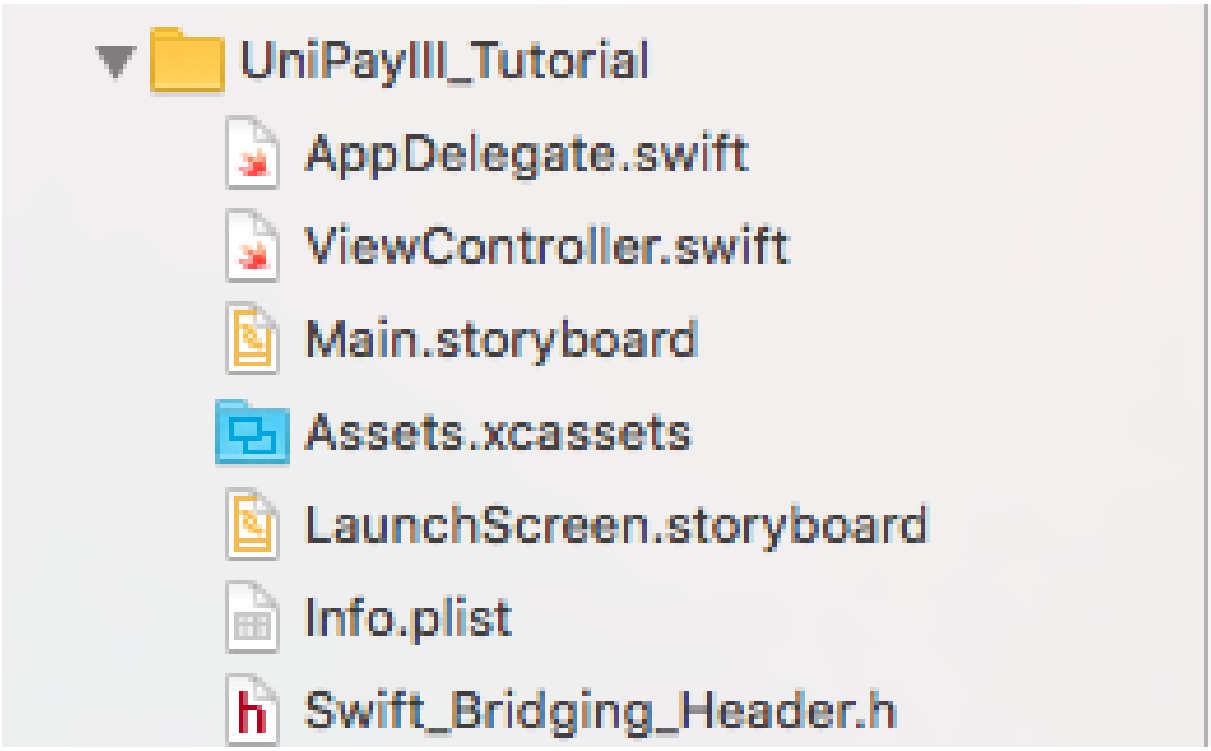
On the file creation screen, select Header File and click "Next"



On the following screen, choose a name for your header file and click "Create"



The bridging header file is now created

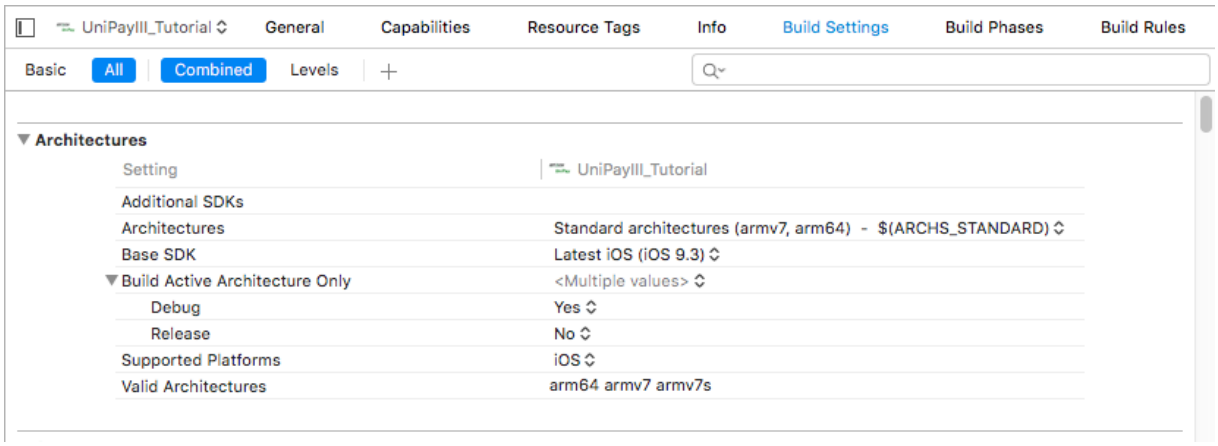


The header file should look similar to the following:

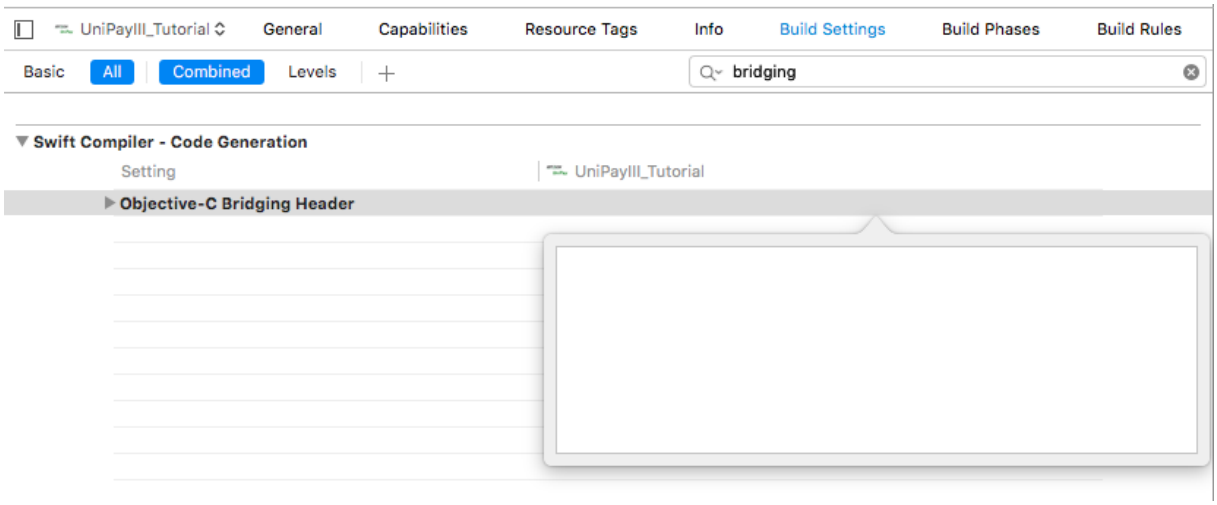
```
#ifndef Swift_Bridging_Header_h
#define Swift_Bridging_Header_h

#endif
```

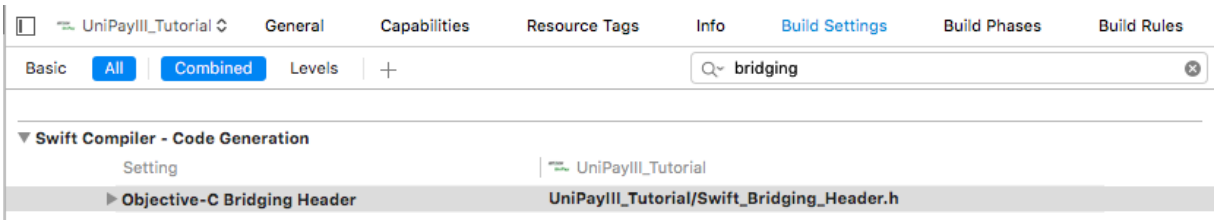
Next, you will need to link your header file to the bridging header setting. Navigate to Build Settings within Xcode



Type "bridging" into the search bar and double click the Objective-C Bridging Header setting to get an input box



Type the relative path to your project directory and the name of the bridging header file and then click enter to save the setting



9.4 Add Import Statement to the Bridging Header File

Inside of the newly created bridging header file, add the following import statement:

```
#import <IDTech/IDTech.h>
```

The file should now look similar to the following:

```
#ifndef Swift_Bridging_Header_h
#define Swift_Bridging_Header_h

#import <IDTech/IDTech.h>

#endif
```

9.5 Amend the View Controller Interface

In the view controller classes that will be a delegate of IDTech.framework, include the reference to the framework delegate class name:

```
class ViewController: UIViewController, IDT_NEO2_Delegate
```

9.6 Implement Optional Delegate Protocols

Implement any/all of the optional delegate protocols used to receive data from IDT_NEO2_Delegate:

```
func deviceConnected()

func deviceDisconnected()

func dataInOutMonitor(data: NSData!, incoming isIncoming: Bool)

func swipeMSRData(cardData: IDTMSRData!)

func deviceMessage(message: NSString!)
```

```
func lcdDisplay(mode: Int32, lines: [AnyObject]!)

func pinpadData(_ value: Data, keySN KSN: Data, event: EVENT_PINPAD_Types)

func emvTransactionData(emvData: IDTEMVData!, errorCode error: Int32)
```

9.7 Allocate/Initialize IDT_NEO2 Object

A Singleton instance has been established in the IDT_NEO2 class. To utilize the delegate protocols, best practices would be to initialize the connection by setting the delegate with the singleton instance.

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.

    IDT_NEO2.sharedController().delegate = self
}
```

9.8 Sample Project Tutorial

Using Xcode 7.3.1, we will create a sample project that will interface with the NEO2 and will perform the following activities:

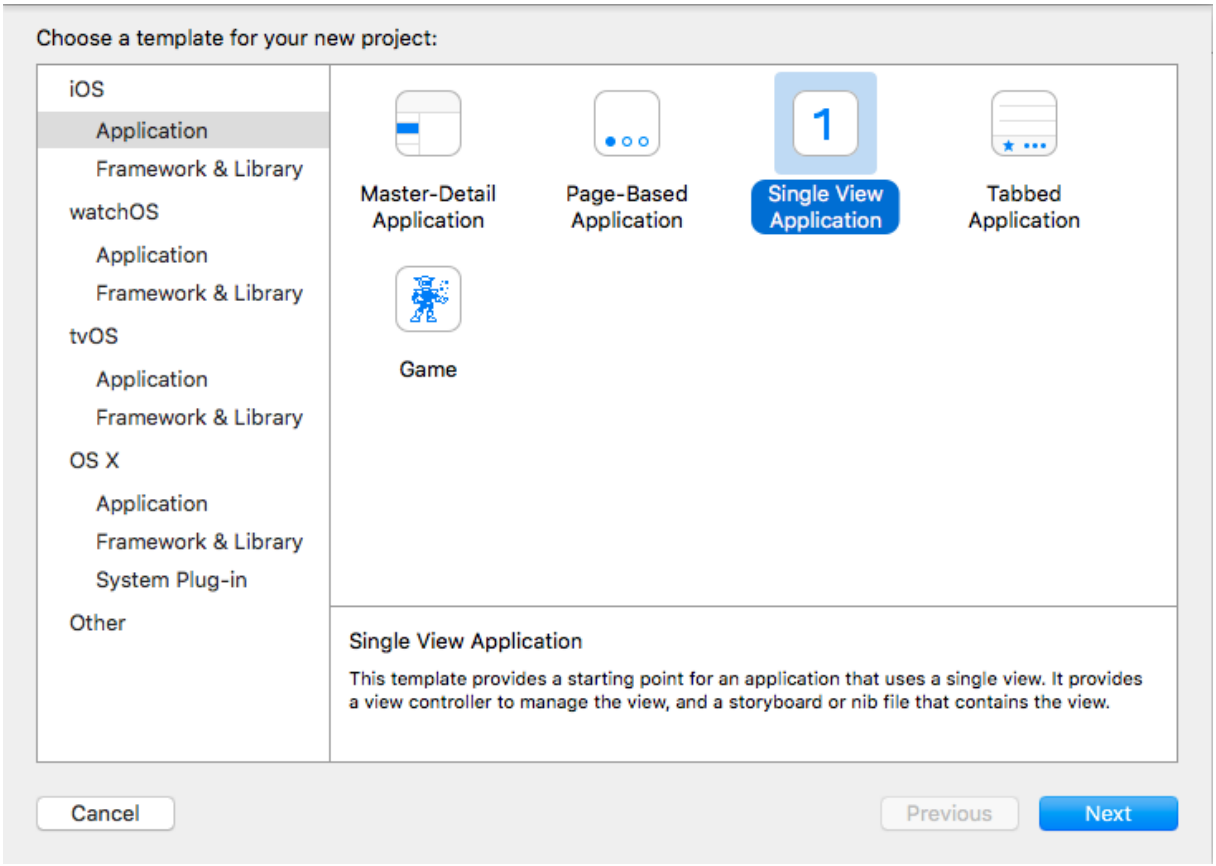
- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for CTLS/MSR (tap/swipe)
- Start/Complete/Cancel EMV Transaction
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

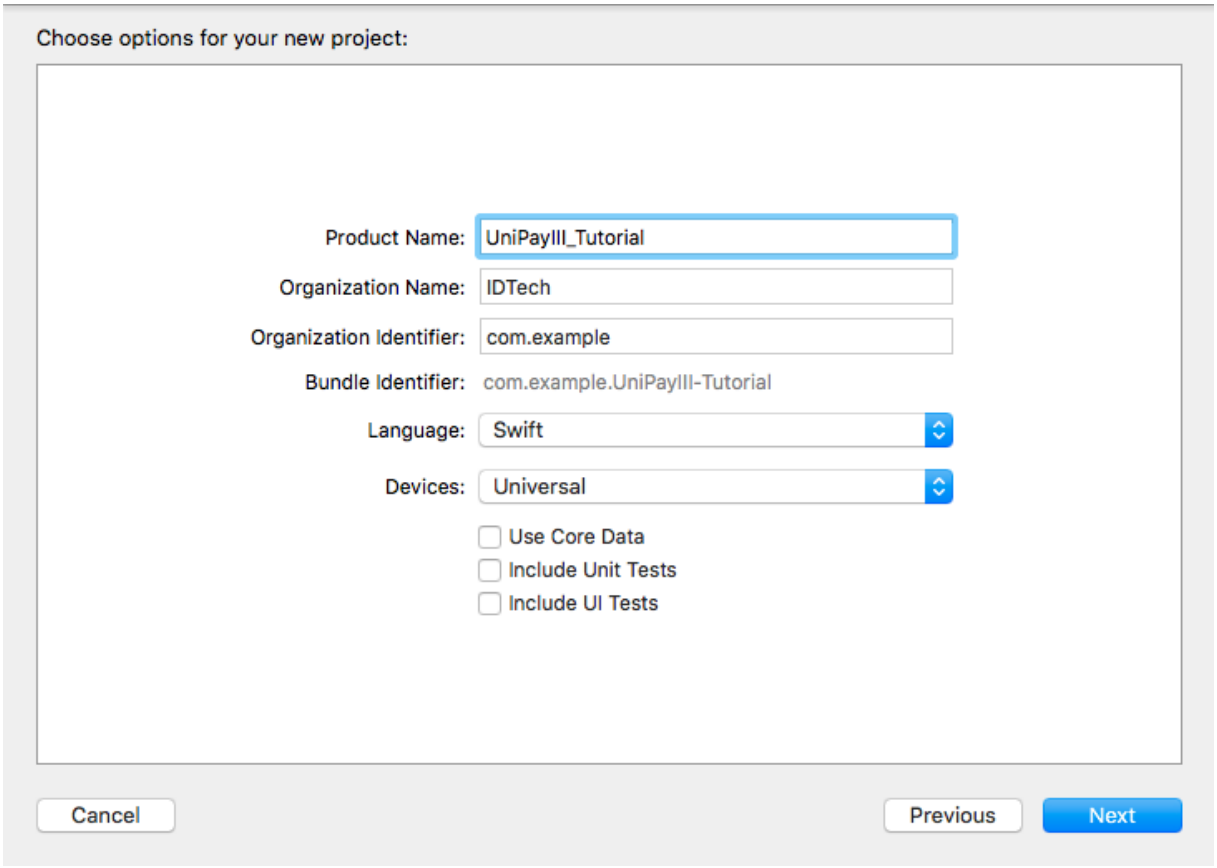
Protocol Delegates:

- Delegate to receive card swipes
- Delegate to detect headphone plug changes
- Delegate to detect device connected
- Delegate to detect device disconnected
- Delegate to receive EMV/CTLS tag data

9.8.1 Step 1: Create New Project

Create a new Single View Application in Xcode



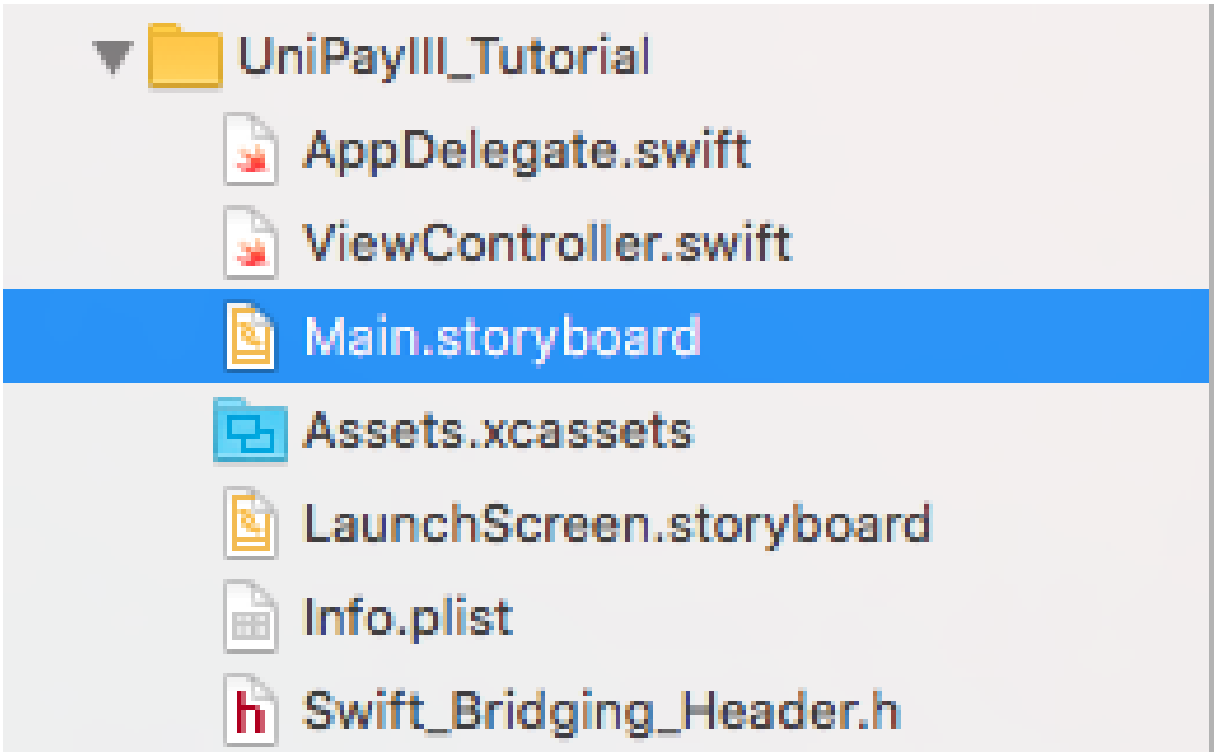


9.8.2 Step 2: Import Frameworks

Import the Necessary Framework/Libraries

9.8.3 Step 3: Design Interface

Design the User Interface by editing the Main.storyboard file



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

- Add a navigation bar at the top to display the application's name
- Add a label to the top that will signify connection/disconnection status.
- Add text views to communicate data from the NEO2 and for EMV LCD display information. Remove the Editable and Selectable behaviors if you don't want the keyboard to pop up if you accidentally select it.
- Add buttons to execute the following functions:
  - Get Firmware
  - Start MSR / CTLS
  - Start ICC EMV
  - Complete ICC EMV
  - Cancel Transaction
    (Add constraints accordingly so layout maps to intended screen size)

9.8.4 Step 4: Configure the Bridging Header and View Controller Files

Create the bridging header file by performing the following:

- Create a Bridging Header File

In the bridging header file, perform the following:

- Add Import Statement to the Bridging Header File

In the view controller file, perform the following:

- Amend the View Controller Interface
- Create an IBOutlet for the two UITextView and link it as a Referencing Outlet to the UITextView on the storyboard
- Create an IBOutlet for the UILabel and link it as a Referencing Outlet to the UILabel on the storyboard
- Create the 5 IBAction for the buttons, and link them to the "Touch Up Inside" event on the storyboard buttons

```
import UIKit

class ViewController: UIViewController, IDT_NEO2_Delegate {
    @IBOutlet weak var connectionStatus: UILabel!
    @IBOutlet weak var lcdTextView: UITextView!
    @IBOutlet weak var logTextView: UITextView!

    @IBAction func getFirmware(sender: UIButton) {}
    @IBAction func startMSR_CTLS(sender: UIButton) {}
    @IBAction func startICCEMV(sender: UIButton) {}
    @IBAction func completeICCEMV(sender: UIButton) {}
    @IBAction func cancelTransaction(sender: UIButton) {}
}
```

Storyboard Source Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0" toolsVersion="10117"
      systemVersion="15F34" targetRuntime="iOS.CocoaTouch" propertyAccessControl="none" useAutolayout="YES"
      useTraitCollections="YES" initialViewController="BYZ-38-t0r">
    <dependencies>
        <deployment identifier="iOS"/>
        <plugIn identifier="com.apple.InterfaceBuilder.IBCocoaTouchPlugin" version="10085"/>
        <capability name="Constraints to layout margins" minToolsVersion="6.0"/>
    </dependencies>
    <scenes>
        <!--View Controller-->
        <scene sceneID="tne-QT-ifu">
            <objects>
                <viewController id="BYZ-38-t0r" customClass="ViewController" customModule="NEO2_Tutorial"
      customModuleProvider="target" sceneMemberID="viewController">
                    <layoutGuides>
                        <viewControllerLayoutGuide type="top" id="y3c-jy-aDJ"/>
                        <viewControllerLayoutGuide type="bottom" id="wfy-db-euE"/>
                    </layoutGuides>
                    <view key="view" contentMode="scaleToFill" id="8bC-Xf-vdC">
                        <rect key="frame" x="0.0" y="0.0" width="600" height="600"/>
                        <autoresizingMask key="autoresizingMask" widthSizable="YES" heightSizable="YES"/>
                        <subviews>
                            <navigationBar contentMode="scaleToFill"
      translatesAutoresizingMaskIntoConstraints="NO" id="etO-Zq-HWr">
                                <rect key="frame" x="0.0" y="20" width="600" height="44"/>
                                <items>
                                    <navigationItem title="NEO2 Tutorial" id="gtg-5k-9F0"/>
                                </items>
                            </navigationBar>
                            <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
      horizontalHuggingPriority="251" verticalHuggingPriority="251" text="Disconnected" textAlignment="center" lineBreak
      tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
      translatesAutoresizingMaskIntoConstraints="NO" id="rZO-0f-qR3">
                                <rect key="frame" x="0.0" y="64" width="600" height="21"/>
                                <color key="backgroundColor" red="1" green="0.0" blue="0.0" alpha="1"
      colorSpace="calibratedRGB"/>
                                <fontDescription key="fontDescription" type="system" pointSize="17"/>
                                <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
      colorSpace="calibratedRGB"/>
                                <nil key="highlightedColor"/>
                            </label>
                            <view contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO"
      id="ZFI-2h-07z">
                                <rect key="frame" x="0.0" y="84" width="600" height="114"/>
                                <subviews>
                                    <button opaque="NO" contentMode="scaleToFill"
      contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="mid
      translatesAutoresizingMaskIntoConstraints="NO" id="tU5-uw-pz7">
                                        <rect key="frame" x="8" y="8" width="112" height="30"/>
                                        <constraints>
                                            <constraint firstAttribute="width" constant="112" id="
      ewS-6U-IbJ"/>
```

```
                                        <constraint firstAttribute="height" constant="30" id="
wJE-mi-v7f"/>
                                    </constraints>
                                    <state key="normal" title="Get Firmware"/>
                                    <connections>
                                        <action selector="getFirmware:" destination="BYZ-38-t0r"
eventType="touchUpInside" id="jeG-VR-ZKR"/>
                                    </connections>
                                </button>
                                <button opaque="NO" contentMode="scaleToFill"
contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="midd
translatesAutoresizingMaskIntoConstraints="NO" id="TGp-e1-4bO">
                                    <rect key="frame" x="8" y="46" width="118" height="30"/>
                                    <constraints>
                                        <constraint firstAttribute="width" constant="118" id="
8Yd-Gz-QRA"/>
                                        <constraint firstAttribute="height" constant="30" id="
k2q-kI-Y0P"/>
                                    </constraints>
                                    <state key="normal" title="Start ICC EMV"/>
                                    <connections>
                                        <action selector="startICCEMV:" destination="BYZ-38-t0r"
eventType="touchUpInside" id="9m1-BE-9sH"/>
                                    </connections>
                                </button>
                                <button opaque="NO" contentMode="scaleToFill"
contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="midd
translatesAutoresizingMaskIntoConstraints="NO" id="wSh-X8-jQA">
                                    <rect key="frame" x="452" y="8" width="140" height="30"/>
                                    <constraints>
                                        <constraint firstAttribute="width" constant="140" id="
g8v-qe-2wp"/>
                                        <constraint firstAttribute="height" constant="30" id="
rDD-ii-GJV"/>
                                    </constraints>
                                    <state key="normal" title="Start MSR / CTLS"/>
                                    <connections>
                                        <action selector="startMSR_CTLS:" destination="BYZ-38-t0r"
eventType="touchUpInside" id="cDe-en-kHV"/>
                                    </connections>
                                </button>
                                <button opaque="NO" contentMode="scaleToFill"
contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="midd
translatesAutoresizingMaskIntoConstraints="NO" id="HHZ-Uc-sT9">
                                    <rect key="frame" x="234" y="84" width="132" height="30"/>
                                    <state key="normal" title="Cancel Transaction"/>
                                    <connections>
                                        <action selector="cancelTransaction:" destination="BYZ-38-t0r"
eventType="touchUpInside" id="GXx-f7-Rgx"/>
                                    </connections>
                                </button>
                                <button opaque="NO" contentMode="scaleToFill"
contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="midd
translatesAutoresizingMaskIntoConstraints="NO" id="hio-sJ-hC4">
                                    <rect key="frame" x="441" y="46" width="151" height="30"/>
                                    <constraints>
                                        <constraint firstAttribute="height" constant="30" id="
UDz-HE-wk5"/>
                                        <constraint firstAttribute="width" constant="151" id="
h6B-ou-ihN"/>
                                    </constraints>
                                    <state key="normal" title="Complete ICC EMV"/>
                                    <connections>
                                        <action selector="completeICCEMV:" destination="BYZ-38-t0r"
eventType="touchUpInside" id="qCC-FA-5Ny"/>
                                    </connections>
                                </button>
                            </subviews>
                            <color key="backgroundColor" white="1" alpha="1" colorSpace="
calibratedWhite"/>
                            <constraints>
                                <constraint firstItem="tU5-uw-pz7" firstAttribute="leading" secondItem=
"ZFI-2h-07z" secondAttribute="leading" constant="8" id="2Dm-Lo-nLb"/>
                                <constraint firstItem="wSh-X8-jQA" firstAttribute="top" secondItem="
ZFI-2h-07z" secondAttribute="top" constant="8" id="52h-Gk-Z0b"/>
                                <constraint firstItem="HHZ-Uc-sT9" firstAttribute="top" secondItem="
ZFI-2h-07z" secondAttribute="top" constant="84" id="8LN-Tw-JGa"/>
                                <constraint firstItem="hio-sJ-hC4" firstAttribute="top" secondItem="
wSh-X8-jQA" secondAttribute="bottom" constant="8" id="BYD-32-uzT"/>
                                <constraint firstAttribute="trailing" secondItem="wSh-X8-jQA"
secondAttribute="trailing" constant="8" id="F76-HQ-FNU"/>
                                <constraint firstAttribute="trailing" secondItem="hio-sJ-hC4"
secondAttribute="trailing" constant="8" id="Fdc-pe-qvL"/>
                                <constraint firstItem="HHZ-Uc-sT9" firstAttribute="centerX" secondItem=
"ZFI-2h-07z" secondAttribute="centerX" id="SzC-4b-OJj"/>
                                <constraint firstItem="TGp-e1-4bO" firstAttribute="top" secondItem="
tU5-uw-pz7" secondAttribute="bottom" constant="8" id="f3n-e6-UUG"/>
```

```
                                <constraint firstAttribute="height" constant="114" id="iHH-b7-rpI"/>
                                <constraint firstItem="TGp-e1-4bO" firstAttribute="leading" secondItem=
"ZFI-2h-07z" secondAttribute="leading" constant="8" id="xwM-oH-jQU"/>
                                <constraint firstItem="tU5-uw-pz7" firstAttribute="top" secondItem="
ZFI-2h-07z" secondAttribute="top" constant="8" id="yjz-Nm-fce"/>
                            </constraints>
                        </view>
                        <view contentMode="scaleToFill" translatesAutoresizingMaskIntoConstraints="NO"
id="Y4B-gd-hVo">
                            <rect key="frame" x="0.0" y="206" width="600" height="394"/>
                            <subviews>
                                <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" text="EMV LCD Display" textAlignment="natural"
lineBreakMode="tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="8IP-lE-FQE">
                                    <rect key="frame" x="234" y="0.0" width="133" height="21"/>
                                    <constraints>
                                        <constraint firstAttribute="width" constant="133" id="
H9G-5s-1ya"/>
                                        <constraint firstAttribute="height" constant="21" id="
OcU-YA-QYt"/>
                                    </constraints>
                                    <fontDescription key="fontDescription" type="system" pointSize="17"
/>
                                    <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
colorSpace="calibratedRGB"/>
                                    <nil key="highlightedColor"/>
                                </label>
                                <textView clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="
scaleToFill" editable="NO" textAlignment="natural" selectable="NO" translatesAutoresizingMaskIntoConstraints=
"NO" id="MFI-aU-Awe">
                                    <rect key="frame" x="8" y="29" width="584" height="160"/>
                                    <color key="backgroundColor" white="0.66666666669999997" alpha="
0.14835831930000001" colorSpace="calibratedWhite"/>
                                    <fontDescription key="fontDescription" type="system" pointSize="14"
/>
                                    <textInputTraits key="textInputTraits" autocapitalizationType="
sentences"/>
                                </textView>
                                <textView clipsSubviews="YES" multipleTouchEnabled="YES" contentMode="
scaleToFill" editable="NO" textAlignment="natural" selectable="NO" translatesAutoresizingMaskIntoConstraints=
"NO" id="RTX-Hq-mMT">
                                    <rect key="frame" x="8" y="226" width="584" height="160"/>
                                    <color key="backgroundColor" white="0.66666666669999997" alpha="
0.14835831930000001" colorSpace="calibratedWhite"/>
                                    <fontDescription key="fontDescription" type="system" pointSize="14"
/>
                                    <textInputTraits key="textInputTraits" autocapitalizationType="
sentences"/>
                                </textView>
                                <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" text="Log" textAlignment="natural" lineBreakMode="
tailTruncation" baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
translatesAutoresizingMaskIntoConstraints="NO" id="G5K-IS-8cP">
                                    <rect key="frame" x="286" y="197" width="29" height="21"/>
                                    <constraints>
                                        <constraint firstAttribute="width" constant="29" id="RTx-9r-dXr
"/>
                                        <constraint firstAttribute="height" constant="21" id="
tub-2e-uEi"/>
                                    </constraints>
                                    <fontDescription key="fontDescription" type="system" pointSize="17"
/>
                                    <color key="textColor" red="0.0" green="0.0" blue="0.0" alpha="1"
colorSpace="calibratedRGB"/>
                                    <nil key="highlightedColor"/>
                                </label>
                            </subviews>
                            <color key="backgroundColor" white="1" alpha="1" colorSpace="
calibratedWhite"/>
                            <constraints>
                                <constraint firstAttribute="trailing" secondItem="MFI-aU-Awe"
secondAttribute="trailing" constant="8" id="0M8-q5-74Y"/>
                                <constraint firstItem="G5K-IS-8cP" firstAttribute="top" secondItem="
MFI-aU-Awe" secondAttribute="bottom" constant="8" id="3nN-pq-c7u"/>
                                <constraint firstAttribute="bottom" secondItem="RTX-Hq-mMT"
secondAttribute="bottom" constant="8" id="4k8-dp-CKK"/>
                                <constraint firstItem="8IP-lE-FQE" firstAttribute="centerX" secondItem=
"Y4B-gd-hVo" secondAttribute="centerX" id="A1H-Xc-KUF"/>
                                <constraint firstItem="RTX-Hq-mMT" firstAttribute="top" secondItem="
G5K-IS-8cP" secondAttribute="bottom" constant="8" id="CB4-g5-bKw"/>
                                <constraint firstItem="G5K-IS-8cP" firstAttribute="centerX" secondItem=
"Y4B-gd-hVo" secondAttribute="centerX" id="DUq-Oc-RKU"/>
                                <constraint firstItem="MFI-aU-Awe" firstAttribute="top" secondItem="
8IP-lE-FQE" secondAttribute="bottom" constant="8" id="EIn-ER-enh"/>
                                <constraint firstAttribute="trailing" secondItem="RTX-Hq-mMT"
secondAttribute="trailing" constant="8" id="H04-dI-fGz"/>
```

```
                                <constraint firstItem="MFI-aU-Awe" firstAttribute="leading" secondItem=
    "Y4B-gd-hVo" secondAttribute="leading" constant="8" id="JT5-MI-88R"/>
                                <constraint firstItem="8IP-lE-FQE" firstAttribute="top" secondItem="
    Y4B-gd-hVo" secondAttribute="top" id="QDs-Pg-soH"/>
                                <constraint firstItem="RTX-Hq-mMT" firstAttribute="leading" secondItem=
    "Y4B-gd-hVo" secondAttribute="leading" constant="8" id="X2o-dk-ffd"/>
                                <constraint firstItem="RTX-Hq-mMT" firstAttribute="height" secondItem="
    MFI-aU-Awe" secondAttribute="height" id="aOL-Ir-y48"/>
                            </constraints>
                        </view>
                    </subviews>
                    <color key="backgroundColor" white="1" alpha="1" colorSpace="custom"
    customColorSpace="calibratedWhite"/>
                    <constraints>
                        <constraint firstItem="etO-Zq-HWr" firstAttribute="leading" secondItem="
    8bC-Xf-vdC" secondAttribute="leadingMargin" constant="-20" id="569-PZ-U8Q"/>
                        <constraint firstAttribute="trailingMargin" secondItem="rZO-0f-qR3"
    secondAttribute="trailing" constant="-20" id="7Hr-mA-xXY"/>
                        <constraint firstItem="rZO-0f-qR3" firstAttribute="leading" secondItem="
    8bC-Xf-vdC" secondAttribute="leadingMargin" constant="-20" id="CDb-2X-s6Q"/>
                        <constraint firstAttribute="trailingMargin" secondItem="Y4B-gd-hVo"
    secondAttribute="trailing" constant="-20" id="FoY-PQ-8gC"/>
                        <constraint firstAttribute="trailingMargin" secondItem="ZFI-2h-07z"
    secondAttribute="trailing" constant="-20" id="HYU-Az-cXm"/>
                        <constraint firstItem="Y4B-gd-hVo" firstAttribute="top" secondItem="ZFI-2h-07z"
     secondAttribute="bottom" constant="8" id="LZS-v7-y1M"/>
                        <constraint firstItem="ZFI-2h-07z" firstAttribute="top" secondItem="etO-Zq-HWr"
     secondAttribute="bottom" constant="20" id="Pcm-zb-zXS"/>
                        <constraint firstAttribute="trailingMargin" secondItem="etO-Zq-HWr"
    secondAttribute="trailing" constant="-20" id="S1d-up-ufn"/>
                        <constraint firstItem="Y4B-gd-hVo" firstAttribute="leading" secondItem="
    8bC-Xf-vdC" secondAttribute="leadingMargin" constant="-20" id="fxv-DL-Q18"/>
                        <constraint firstItem="wfy-db-euE" firstAttribute="top" secondItem="Y4B-gd-hVo"
     secondAttribute="bottom" id="l4D-dJ-DrY"/>
                        <constraint firstItem="etO-Zq-HWr" firstAttribute="top" secondItem="y3c-jy-aDJ"
     secondAttribute="bottom" id="pfJ-Fb-EWV"/>
                        <constraint firstItem="rZO-0f-qR3" firstAttribute="top" secondItem="etO-Zq-HWr"
     secondAttribute="bottom" id="vXY-dJ-IM4"/>
                        <constraint firstItem="ZFI-2h-07z" firstAttribute="leading" secondItem="
    8bC-Xf-vdC" secondAttribute="leadingMargin" constant="-20" id="y5F-nP-7CB"/>
                    </constraints>
                </view>
                <connections>
                    <outlet property="connectionStatus" destination="rZO-0f-qR3" id="8PR-rE-16S"/>
                    <outlet property="lcdTextView" destination="MFI-aU-Awe" id="Hzq-Kn-xTJ"/>
                    <outlet property="logTextView" destination="RTX-Hq-mMT" id="NRT-pZ-ik7"/>
                </connections>
            </viewController>
            <placeholder placeholderIdentifier="IBFirstResponder" id="dkx-z0-nzr" sceneMemberID="
    firstResponder"/>
        </objects>
        <point key="canvasLocation" x="698" y="449"/>
    </scene>
</scenes>
</document>
```

9.8.5 Step 5: Finalize the View Controller File

In the view controller file, perform the following:

- Set delegate and initialize IDT_NEO2 singleton object in the `viewDidLoad` method. To automatically connect to BLE, enter the friendly name of the device Reference: Allocate/Initialize IDT_NEO2 Object

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.

    IDT_NEO2.sharedController().delegate = self
    IDT_NEO2.sharedController().device_setBLEFriendlyName("VP-2722"); //set to
       device ble name
    IDT_NEO2.sharedController().device_enableBLEDeviceSearch(nil);
}
```

- Implement protocol delegate IDT_NEO2Delegate::deviceDisconnected() and IDT_NEO2Delegate::deviceConnected() to monitor connect/disconnect events and modify our connection label upon change. Reference: Implement Optional Delegate Protocols

```
func setConnectionStatus(_ status: String, backgroundColor: UIColor) {
    connectionStatus.text = status;
    connectionStatus.backgroundColor = backgroundColor
}

func appendMessageToLog(_ message: String) {
    logTextView.text = "\n" + (logTextView.text) + "\n" + (message)
    logTextView.scrollRangeToVisible(NSRange(location: 0, length: 0))
}

func displayReturnError(_ operation: String, rt: RETURN_CODE) {
    let message = operation+" ERROR: ID-" + String(describing: rt) + "Message: " + (
      IDT_NEO2.sharedController().device_getResponseCodeString(Int32(rt.rawValue)))

    appendMessageToLog(message)
}

func deviceConnected() {
    setConnectionStatus("Connected", backgroundColor: UIColor.green)
    appendMessageToLog("NEO2 Connected\nFramework Version:" + IDT_Device.sdk_version())
}

func deviceDisconnected() {
    setConnectionStatus("Disconnected", backgroundColor: UIColor.red)
}
```

- Implement protocol delegate swipeMSRData:() to receive card swipe data. Reference: Implement Optional Delegate Protocols

```
func swipeMSRData(_ cardData: IDTMSRData!) {
    NSLog("--MSR event received, type: " + String(describing: cardData.event) + " data: " + cardData.
      encTrack1.hexEncodedString())
    switch cardData.event {
    case EVENT_MSR_CARD_DATA:
        switch cardData.captureEncodeType {
            case CAPTURE_ENCODE_TYPE_ISOABA:
                appendMessageToLog("Encode Type: ISO/ABA")
            case CAPTURE_ENCODE_TYPE_AAMVA:
                appendMessageToLog("Encode Type: AA/MVA")
            case CAPTURE_ENCODE_TYPE_Other:
                appendMessageToLog("Encode Type: Other")
            case CAPTURE_ENCODE_TYPE_Raw:
                appendMessageToLog("Encode Type: Raw")
            case CAPTURE_ENCODE_TYPE_JIS_I:
                appendMessageToLog("Encode Type: CAPTURE_ENCODE_TYPE_JIS_I")
            case CAPTURE_ENCODE_TYPE_JIS_II:
                appendMessageToLog("Encode Type: CAPTURE_ENCODE_TYPE_JIS_II")
            default:
                appendMessageToLog("Encode Type: UNKWOWN")
        }

        switch cardData.captureEncryptType {
            case CAPTURE_ENCRYPT_TYPE_AES:
                appendMessageToLog("Encrypt Type: AES")
            case CAPTURE_ENCRYPT_TYPE_TDES:
                appendMessageToLog("Encrypt Type: TDES")
            case CAPTURE_ENCRYPT_TYPE_NO_ENCRYPTION:
                appendMessageToLog("Encrypt Type: NONE")
            default:
                appendMessageToLog("Encrypt Type: UNKNOWN")
        }

        appendMessageToLog("Full card data: " + (cardData.cardData == nil ? "N/A" : cardData.cardData.
      hexEncodedString()))
        appendMessageToLog("Track 1: " + (cardData.track1 == nil ? "N/A" : cardData.track1))
        appendMessageToLog("Track 2: " +  (cardData.track2 == nil ? "N/A" : cardData.track2))
        appendMessageToLog("Track 3: " + (cardData.track3 == nil ? "N/A" : cardData.track3))
        appendMessageToLog("Length Track 1: " + cardData.track1Length.description)
        appendMessageToLog("Length Track 2: " + cardData.track2Length.description)
        appendMessageToLog("Length Track 3: " + cardData.track3Length.description)
        appendMessageToLog("Encoded Track 1: " + (cardData.encTrack1 == nil ? "N/A" : cardData.encTrack1.
      hexEncodedString()))
```

```
        appendMessageToLog("Encoded Track 2: " + (cardData.encTrack2 == nil ? "N/A" : cardData.encTrack2.
    hexEncodedString()))
        appendMessageToLog("Encoded Track 3: " + (cardData.encTrack3 == nil ? "N/A" : cardData.encTrack3.
    hexEncodedString()))
        appendMessageToLog("Hash Track 1: " + (cardData.hashTrack1 == nil ? "N/A" : cardData.hashTrack1.
    hexEncodedString()))
        appendMessageToLog("Hash Track 2: " + (cardData.hashTrack2 == nil ? "N/A" : cardData.hashTrack2.
    hexEncodedString()))
        appendMessageToLog("Hash Track 3: " + (cardData.hashTrack3 == nil ? "N/A" : cardData.hashTrack3.
    hexEncodedString()))
        appendMessageToLog("KSN: " +  (cardData.ksn == nil ? "N/A" : cardData.ksn.hexEncodedString()))
        appendMessageToLog("\nSessionID: " +   (cardData.sessionID == nil ? "N/A" : cardData.sessionID.
    hexEncodedString()))
        appendMessageToLog("\nReader Serial Number: " +   (cardData.rsn == nil ? "N/A" : cardData.rsn))
        appendMessageToLog("\nRead Status: " + String(describing: cardData.readStatus))

        if cardData.unencryptedTags != nil {
            appendMessageToLog("Unencrypted Tags: " + String(describing: cardData.unencryptedTags))
        }

        if cardData.encryptedTags != nil {
            appendMessageToLog("Encrypted Tags: " + String(describing: cardData.encryptedTags))
        }

        if cardData.maskedTags != nil {
            appendMessageToLog("Masked Tags: " + String(describing: cardData.maskedTags))
        }

        NSLog("Track 1: " + (cardData.track1 == nil ? "N/A" : cardData.track1))
        NSLog("Track 2: " + (cardData.track2 == nil ? "N/A" : cardData.track2))
        NSLog("Track 3: " + (cardData.track3 == nil ? "N/A" : cardData.track3))
        NSLog("Encoded Track 1: " + (cardData.encTrack1 == nil ? "N/A" : cardData.encTrack1.
    hexEncodedString()))
        NSLog("Encoded Track 2: " + (cardData.encTrack2 == nil ? "N/A" : cardData.encTrack2.
    hexEncodedString()))
        NSLog("Encoded Track 3: " + (cardData.encTrack3 == nil ? "N/A" : cardData.encTrack3.
    hexEncodedString()))
        NSLog("Hash Track 1: " + (cardData.hashTrack1 == nil ? "N/A" : cardData.hashTrack1.hexEncodedString
    ()))
        NSLog("Hash Track 2: " + (cardData.hashTrack2 == nil ? "N/A" : cardData.hashTrack2.hexEncodedString
    ()))
        NSLog("Hash Track 3: " + (cardData.hashTrack3 == nil ? "N/A" : cardData.hashTrack3.hexEncodedString
    ()))
        NSLog("SessionID: " + (cardData.sessionID == nil ? "N/A" : cardData.sessionID.hexEncodedString()))
        NSLog("nReader Serial Number: " + (cardData.rsn == nil ? "N/A" : cardData.rsn))
        NSLog("Read Status: " + String(describing: cardData.readStatus))
        NSLog("KSN: " + (cardData.ksn == nil ? "N/A" : cardData.ksn.hexEncodedString()))

    case EVENT_MSR_CANCEL_KEY:
        appendMessageToLog("(Event) MSR Cancel Key received: " + cardData.encTrack1.hexEncodedString())

    case EVENT_MSR_BACKSPACE_KEY:
        appendMessageToLog("(Event) MSR Backspack Key received: " + cardData.encTrack1.hexEncodedString())

    case EVENT_MSR_ENTER_KEY:
        appendMessageToLog("(Event) MSR Enter Key received: " + cardData.encTrack1.hexEncodedString())

    case EVENT_MSR_UNKNOWN:
        appendMessageToLog("(Event) MSR unknown event, data: " + cardData.encTrack1.hexEncodedString())
    case EVENT_MSR_TIMEOUT:
        appendMessageToLog("MSR Timeout")

    default:
        break

    }
}
```

- Implement protocol delegate emvTransactionData:() to report EMV transaction results. Reference: Implement Optional Delegate Protocols

```
func emvTransactionData(_ emvData: IDTEMVData!, errorCode error: Int32) {

    NSLog("EMV_RESULT_CODE_V2_response = " + String(describing: error))

    appendMessageToLog("EMV transaction data response: " + IDT_NEO2.
      sharedController().device_getResponseCodeString(error))

    if emvData == nil {
        appendMessageToLog("EMV TRANSACTION ERROR. Refer to EMV_RESULT_CODE_V2_response = " + error.
```

```
      description)
        return;
    }

    if emvData.resultCodeV2 != EMV_RESULT_CODE_V2_NO_RESPONSE {
        appendMessageToLog("EMV_RESULT_CODE_V2_RESPONSE: " + String(describing: emvData.resultCodeV2))
    }

    if emvData.resultCodeV2 == EMV_RESULT_CODE_V2_GO_ONLINE {
        appendMessageToLog("ONLINE REQUEST")
    }

    if emvData.resultCodeV2 == EMV_RESULT_CODE_V2_START_TRANS_SUCCESS {
        appendMessageToLog("Start success: authentication required")
    }

    if emvData.resultCodeV2 == EMV_RESULT_CODE_V2_APPROVED || emvData.resultCodeV2 ==
      EMV_RESULT_CODE_V2_APPROVED_OFFLINE {
        appendMessageToLog("APPROVED");
    }

    if emvData.resultCodeV2 == EMV_RESULT_CODE_V2_MSR_SUCCESS {
        appendMessageToLog("MSR Data Captured")
    }

    if emvData.cardType == 0 {
        appendMessageToLog("CONTACT")
    }

    if emvData.cardType == 1 {
        appendMessageToLog("CONTACTLESS")
    }

    if emvData.unencryptedTags != nil {
        appendMessageToLog("Unencrypted Tags: " + String(describing: emvData.unencryptedTags))
    }

    if emvData.encryptedTags != nil {
        appendMessageToLog("Encrypted Tags: " + String(describing: emvData.encryptedTags))
    }

    if emvData.maskedTags != nil {
        appendMessageToLog("Masked Tags: " + String(describing: emvData.maskedTags))
    }

    if emvData.hasAdvise {
        appendMessageToLog("Response has advise request")
    }

    if emvData.hasReversal {
        appendMessageToLog("Response has reversal request")
    }
}
```

- Implement protocol delegate lcdDisplay:() to receive LCD messages, and automatically select 1st menu item/language when presented with choices. Normal operation would require a choice be made by card holder. Reference: Implement Optional Delegate Protocols

```
func lcdDisplay(_ mode: Int32, lines: [AnyObject]!) {
    var str = ""

    if lines != nil {
        for s in lines {
            str += s as! String
            str += "\n"
        }
    }

    switch mode {
        case 0x10:
            lcdTextView.text = ""
        case 0x03:
            lcdTextView.text = str
        case 0x01, 0x02, 0x08:
            IDT_NEO2.sharedController().emv_callbackResponseLCD(mode, selection: 1)
        default:
            break
    }
}
```

- Implement the button press methods

```
@IBAction func getFirmware(_ sender: UIButton) {
    var result: NSString?
    let rt = IDT_NEO2.sharedController().device_getFirmwareVersion(&result)

    logTextView.text = ""

    if RETURN_CODE_DO_SUCCESS == rt {
        appendMessageToLog("Get firmware: " + (result! as String))
    } else {
        displayReturnError("Get firmware", rt: rt)
    }
}

@IBAction func startMSR_CTLS(_ sender: UIButton) {
    let rt = IDT_NEO2.sharedController().ctls_startTransaction(1.00, type: 00,
      timeout: 30, tags: nil)

    logTextView.text = ""

    if RETURN_CODE_DO_SUCCESS == rt {
        appendMessageToLog("Enabled MSR / CTLS")
    } else {
        displayReturnError("Start MSR / CTLS", rt: rt)
    }
}

@IBAction func startICCEMV(_ sender: UIButton) {
    let rt = IDT_NEO2.sharedController().emv_startTransaction(1.00, amtOther: 0,
      type: 0, timeout: 60, tags: nil, forceOnline: false, fallback: true)

    logTextView.text = ""

    if RETURN_CODE_DO_SUCCESS == rt {
        appendMessageToLog("Start Transaction Command Accepted")
    } else {
        displayReturnError("Start ICC EMV", rt: rt)
    }
}

@IBAction func completeICCEMV(_ sender: UIButton) {
    let rt = IDT_NEO2.sharedController().emv_completeOnlineEMVTransaction(true,
      hostResponseTags: IDTUtility.hex(toData: "8A023030"))

    logTextView.text = ""

    if RETURN_CODE_DO_SUCCESS == rt {
        appendMessageToLog("Complete Transaction Command Accepted")
    } else {
        displayReturnError("Complete ICC EMV", rt: rt)
    }
}

@IBAction func cancelTransaction(_ sender: UIButton) {
    let rt = IDT_NEO2.sharedController().
      ctls_cancelTransaction()

    logTextView.text = ""

    if RETURN_CODE_DO_SUCCESS == rt {
        appendMessageToLog("Canceled transaction")
    } else {
        displayReturnError("Cancel transaction", rt: rt)
    }
}
```

Chapter 10

NEO2 Error Code Reference

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

```
EE0E    User Interface Event
EE11    Communication type not supported, VT-1, burst, etc.
EE12    Secure interface is not functional or is in an intermediate state.
EE13    Data field is not mod 8
EE14    Pad 0x80 not found where expected
EE15    Specified key type is invalid
EE16    Could not retrieve key from the SAM (InitSecureComm)
EE17    Hash code problem
EE18    Could not store the key into the SAM (InstallKey)
EE19    Frame is too large
EE1A    Unit powered up in authentication state but POS must resend the InitSecureComm command
EE1B    The EEPROM may not be initialized because SecCommInterface does not make sense
EE1C    Problem encoding APDU
EE20    Unsupported Index (ILM) SAM Transceiver error - problem communicating with the SAM (Key Mgr)
EE21    Unexpected Sequence Counter in multiple frames for single bitmap (ILM) Length error in data
      returned from the SAM (Key Mgr)
EE22    Improper bit map (ILM)
EE23    Request Online Authorization
EE24    ViVOCard3 raw data read successful
EE25    Message index not available (ILM) ViVOcomm activate transaction card type (ViVOcomm)
EE26    Version Information Mismatch (ILM)
EE27    Not sending commands in correct index message index (ILM)
EE28    Time out or next expected message not received (ILM)
EE29    ILM languages not available for viewing (ILM)
EE2A    Other language not supported (ILM)
EE50    Auto-Switch OK
EE51    Auto-Switch failed
EE60    Data not exist
EE61    Data Full
EE62    Write Flash Error
EE63    Ok and Have Next Command
EE80    Cannot start Contact EMV transaction
EE81    CTLS/MSR cancelled due to card insertion
EE90    Account DUKPT Key not exist
EE91    Account DUKPT Key KSN exhausted
```

Chapter 11

Enumeration Reference

IDTMSRData

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

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

IDTCommon

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

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

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

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

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

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

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

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

    RETURN_CODE_EMV_AUTHORIZATION_ACCEPTED = 0x0E00,
    RETURN_CODE_EMV_AUTHORIZATION_UNABLE_TO_GO_ONLINE = 0x0E01,
```

```
    RETURN_CODE_EMV_AUTHORIZATION_TECHNICAL_ISSUE = 0x0E02,
    RETURN_CODE_EMV_AUTHORIZATION_DECLINED = 0x0E03,
    RETURN_CODE_EMV_AUTHORIZATION_ISSUER_REFERRAL = 0x0E04,

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



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


} RETURN_CODE;




typedef enum{
    EMV_RESULT_CODE_V2_NO_RESPONSE = -1,
    EMV_RESULT_CODE_V2_APPROVED_OFFLINE = 0x0000,
    EMV_RESULT_CODE_V2_DECLINED_OFFLINE = 0x0001,
    EMV_RESULT_CODE_V2_APPROVED = 0x0002,
    EMV_RESULT_CODE_V2_DECLINED = 0x0003,
    EMV_RESULT_CODE_V2_GO_ONLINE = 0x0004,
    EMV_RESULT_CODE_V2_CALL_YOUR_BANK = 0x0005,
    EMV_RESULT_CODE_V2_NOT_ACCEPTED = 0x0006,
    EMV_RESULT_CODE_V2_USE_MAGSTRIPE = 0x0007,
    EMV_RESULT_CODE_V2_TIME_OUT = 0x0008,
    EMV_RESULT_CODE_V2_START_TRANS_SUCCESS = 0x0010,
    EMV_RESULT_CODE_V2_SWIPE_NON_ICC = 0x0011,
    EMV_RESULT_CODE_V2_TRANSACTION_CANCELLED = 0x0012,
    EMV_RESULT_CODE_V2_GO_ONLINE_CTLS = 0x0023,
    EMV_RESULT_CODE_CTLS_TWO_CARDS = 0x7A,
    EMV_RESULT_CODE_CTLS_TERMINATE = 0x7E,
    EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER = 0x7D,
    EMV_RESULT_CODE_MSR_SWIPE_CAPTURED = 0x80,
    EMV_RESULT_CODE_REQUEST_ONLINE_PIN = 0x81,
    EMV_RESULT_CODE_REQUEST_SIGNATURE = 0x82,
    EMV_RESULT_CODE_FALLBACK_TO_CONTACT = 0x83,
    EMV_RESULT_CODE_FALLBACK_TO_OTHER = 0x84,
    EMV_RESULT_CODE_REVERSAL_REQUIRED = 0x85,
    EMV_RESULT_CODE_ADVISE_REQUIRED = 0x86,
    EMV_RESULT_CODE_ADVISE_REVERSAL_REQUIRED = 0x87,
    EMV_RESULT_CODE_NO_ADVISE_REVERSAL_REQUIRED = 0x88,
    EMV_RESULT_CODE_UNABLE_TO_REACH_HOST = 0xFF,
    EMV_RESULT_CODE_V2_FILE_ARG_INVALID = 0x1001,
    EMV_RESULT_CODE_V2_FILE_OPEN_FAILED = 0x1002,
    EMV_RESULT_CODE_V2_FILE_OPERATION_FAILED = 0X1003,
    EMV_RESULT_CODE_V2_MEMORY_NOT_ENOUGH = 0x2001,
    EMV_RESULT_CODE_V2_SMARTCARD_FAIL = 0x3001,
    EMV_RESULT_CODE_V2_SMARTCARD_INIT_FAILED = 0x3003,
    EMV_RESULT_CODE_V2_FALLBACK_SITUATION = 0x3004,
    EMV_RESULT_CODE_V2_SMARTCARD_ABSENT = 0x3005,
    EMV_RESULT_CODE_V2_SMARTCARD_TIMEOUT = 0x3006,
    EMV_RESULT_CODE_V2_MSR_CARD_ERROR = 0x3007,
    EMV_RESULT_CODE_V2_MSR_CARD_READ_ERROR = 0x3012,
    EMV_RESULT_CODE_V2_TIMEOUT_INSERT_OR_FALLBACK = 0x3013,
    EMV_RESULT_CODE_V2_PARSING_TAGS_FAILED= 0X5001,
    EMV_RESULT_CODE_V2_CARD_DATA_ELEMENT_DUPLICATE = 0X5002,
    EMV_RESULT_CODE_V2_DATA_FORMAT_INCORRECT = 0X5003,
    EMV_RESULT_CODE_V2_APP_NO_TERM = 0X5004,
    EMV_RESULT_CODE_V2_APP_NO_MATCHING = 0X5005,
    EMV_RESULT_CODE_V2_AMANDATORY_OBJECT_MISSING = 0X5006,
    EMV_RESULT_CODE_V2_APP_SELECTION_RETRY = 0X5007,
    EMV_RESULT_CODE_V2_AMOUNT_ERROR_GET = 0X5008,
    EMV_RESULT_CODE_V2_CARD_REJECTED = 0X5009,
    EMV_RESULT_CODE_V2_AIP_NOT_RECEIVED = 0X5010,
    EMV_RESULT_CODE_V2_AFL_NOT_RECEIVEDE = 0X5011,
    EMV_RESULT_CODE_V2_AFL_LEN_OUT_OF_RANGE = 0X5012,
    EMV_RESULT_CODE_V2_SFI_OUT_OF_RANGE = 0X5013,
```

```
    EMV_RESULT_CODE_V2_AFL_INCORRECT = 0X5014,
    EMV_RESULT_CODE_V2_EXP_DATE_INCORRECT = 0X5015,
    EMV_RESULT_CODE_V2_EFF_DATE_INCORRECT = 0X5016,
    EMV_RESULT_CODE_V2_ISS_COD_TBL_OUT_OF_RANGE = 0X5017,
    EMV_RESULT_CODE_V2_CRYPTOGRAM_TYPE_INCORRECT = 0X5018,
    EMV_RESULT_CODE_V2_PSE_BY_CARD_NOT_SUPPORTED = 0X5019,
    EMV_RESULT_CODE_V2_USER_LANGUAGE_SELECTED = 0X5020,
    EMV_RESULT_CODE_V2_SERVICE_NOT_ALLOWED = 0X5021,
    EMV_RESULT_CODE_V2_NO_TAG_FOUND = 0X5022,
    EMV_RESULT_CODE_V2_CARD_BLOCKED = 0X5023,
    EMV_RESULT_CODE_V2_LEN_INCORRECT = 0X5024,
    EMV_RESULT_CODE_V2_CARD_COM_ERROR = 0X5025,
    EMV_RESULT_CODE_V2_TSC_NOT_INCREASED = 0X5026,
    EMV_RESULT_CODE_V2_HASH_INCORRECT = 0X5027,
    EMV_RESULT_CODE_V2_ARC_NOT_PRESENCED = 0X5028,
    EMV_RESULT_CODE_V2_ARC_INVALID = 0X5029,
    EMV_RESULT_CODE_V2_COMM_NO_ONLINE = 0X5030,
    EMV_RESULT_CODE_V2_TRAN_TYPE_INCORRECT = 0X5031,
    EMV_RESULT_CODE_V2_APP_NO_SUPPORT = 0X5032,
    EMV_RESULT_CODE_V2_APP_NOT_SELECT = 0X5033,
    EMV_RESULT_CODE_V2_LANG_NOT_SELECT = 0X5034,
    EMV_RESULT_CODE_V2_TERM_DATA_NOT_PRESENCED = 0X5035,
    EMV_RESULT_CODE_V2_CVM_TYPE_UNKNOWN = 0X6001,
    EMV_RESULT_CODE_V2_CVM_AIP_NOT_SUPPORTED = 0X6002,
    EMV_RESULT_CODE_V2_CVM_TAG_8E_MISSING = 0X6003,
    EMV_RESULT_CODE_V2_CVM_TAG_8E_FORMAT_ERROR = 0X6004,
    EMV_RESULT_CODE_V2_CVM_CODE_IS_NOT_SUPPORTED = 0X6005,
    EMV_RESULT_CODE_V2_CVM_COND_CODE_IS_NOT_SUPPORTED = 0X6006,
    EMV_RESULT_CODE_V2_CVM_NO_MORE = 0X6007,
    EMV_RESULT_CODE_V2_PIN_BYPASSED_BEFORE = 0X6008,
    EMV_RESULT_CODE_V2_GO_ONLINE_CL = 0xEE23
} EMV_RESULT_CODE_V2_Types;



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

Chapter 12

EMV Tag Reference

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

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

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

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

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

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

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

Tag	Description
DFEF21	BIN
DFEF22	AID
DFEF23	HMAC
DFEF24	HMAC KSN
DFEF25	Output Data Format Select
DFEF26	MSR fallback
DFEF27	Online capability
DFEF28	Disable Encrypt ON
DFEF2C	Terminal AID List
DFEF2E	Terminal Transaction Log
DFEF2F	CUP configuration
DFEF30	White List
DFEF31	Black List
DFEF32	Auto-Switch
DFEF34	Antenna Detection Switch
DFEF35	Communications Watchdog Period
DFEF36	Media Control & Status Tracking
DFEF37	Interface Select
DFEF38	Timeout for Next Command
DFEF39	Network Indicate
DFEF3A	Reader Behavior Mode
DFEF3B	Autopoll Transaction Separation Interval
DFEF40	Ascii-code encryption Tag57 TLV
DFEF41	MAC Verification Data for SRED
DFEF42	MAC VerificationKSN for SRED
DFEF43	Local TZ/DST information.
DFEF44	CombinationOptions
DFEF45	Removal Timeout
DFEF46	ACT Pass Response DOL
DFEF47	CDA Hash Input
DFEF48	Indicate - retrieve transaction resultagain due to Output RAM isNotenough.
DFEF49	Outcome Parameter Set
DFE↩ F4A	User Interface Request Data
DFEF4B	MSR Equivalent Data Option
DFEF4C	MSR Equivalent Data Track Lengths
DFEF4D	MSR Equivalent Data
DFEF4E	ACT MSD Response DOL
DFEF4F	ACT Decline Response DOL
DFEF50	Terminal Interchange Profile (JCB)
DFEF51	Bypass EMV Completion Output
DFEF52	Re-FallBack times
DFEF53	Dynamic Reader Limits
DFEF54	SmartTap AID Index
DFEF55	Kernel Specific Features
DFEF56	Retry Limit
DFEF57	PPSE Terminate Flags
DFEF59	Terminal Data Setting - Default Amount
DFEF5A	Terminal Data Setting - Tags to Return
DFE↩ F5B	Mask for Tag5A

Tag	Description
DFEF5C	Mask for Tag56
DFEF5D	Mask for Tag57
DFEF5E	Mask for Tag9F6B
DFEF5F	Mask forTagFFEE13
DFEF60	Mask forTagFFEE14
DFEF61	Error Code
DFEF62	Allow MSR Swipe data from ICC Card
DFEF63	Tags To Read Yet
DFEF64	Referral Timeout
DFEF6E	USB-KB Output Data Postfix
DFEF6F	Inter-character Delay for USB-KB Interface
DFEF70	PISCES dual interface interference prevention mechanism fine-tune parameters.
DFEF71	Waiting ICC insert time
DFEF72	Pre-poll card mechanism control in ACT cmd & config setting
DFEF73	Transaction Message Type
DFEF74	Reference amplitude value
DFEF75	Reference delta value
DFEF76	Transaction Interface Type to activate
DFEF77	Timeout for waiting next command
DFEF78	EMV contact L2 display messages option
DFEF79	PIN block format (when TDES)
DFEF7A	Enable Apple Pay Check
DFEF7B	Apple Pay Status
DFEF7C	Track Bit Encoding
DFEF7D	Re-power on times
DFEF7E	Fallback response code list
FF69	ViVOpay Proprietary Tag List
FF70	Serial Finite State Machine Version
FF71	Transaction Finite State Machine Version
FF72	System Information Suite
FF73	Serial Protocol Version
FF74	Serial Protocol Suite
FF75	L1 Paypass Version
FF76	L1 LCR Version
FF77	L2 Card App Version
FF78	L2 Card App Suite
FF79	GMEDs Data
FF79	User Experience Version
FF7A	User Experience Suite
FF7B	ViVOtech Proprietary Suite
FF7C	VIUDS Scheme IDs Supported
FF7D	VIUDS Scheme ID Selection Criteria
FFE0	Registered Application Provider Identifier (RID)
FFE1	Partial Selection Allowed
FFE2	Application Flow
FFE3	Selection Features - GR 1.2.10
FFE4	Group Number / Fallback Group
FFE5	Max AID Length
FFE6	AID Disabled

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

9F33 Terminal Capabilities

```
Byte 1
```

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

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

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

9F40 Additional Terminal Capabilities

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

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

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

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

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

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

9F35 Terminal Type

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

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

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

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

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

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

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

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


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

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

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

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

Chapter 13

Hierarchical Index

13.1 Class Hierarchy

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

Chapter 14

Class Index

14.1 Class List

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

Chapter 15

Class Documentation

15.1 ADF_Info Struct Reference

Public Attributes

- int **type**
- int **address**
- int **size**

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

- Source_iOS/IDTCommon.h

15.2 AIDEntry Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- unsigned char aid [16]

  AID value as per payment networks.
- unsigned char aidLen

  AID's length.

15.2.1 Detailed Description

AID Entry - Used to populate array in IDT_BTPay::emv_retrieveAIDList:() IDT_UniPay::emv_retrieveAIDList:().

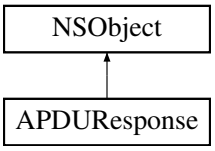
The documentation for this struct was generated from the following file:

- Source_iOS/IDTCommon.h

15.3 APDUResponse Class Reference

`#import <APDUResponse.h>`

Inheritance diagram for APDUResponse:



Instance Methods

- (void) - clear

Class Methods

- (APDUResponse ∗) + sharedController

Properties

- unsigned char SW1

  Status Word Byte 1.
- unsigned char SW2

  Status Word Byte 2.
- BOOL hasKSN

  KSN data read.
- BOOL hasEncryption

  APDU response is encrypted.
- int apduLength

  Length of valid R-APDU.
- NSData ∗ response

  APDU Response excluding SW1 and SW2.
- NSData ∗ ksn

  Key Seral Number.

15.3.1 Detailed Description

Used in IDT_BTpay::icc_exchangeAPDU:encrypted:ksn:response:() IDT_UniPay::icc_exchangeAPDU:encrypted↩ :response:()

15.3.2 Method Documentation

15.3.2.1 - (void) clear

clears all APDUResponse properties

15.3.2.2 + (APDUResponse ∗) sharedController

Singleton instance of APDUResponse

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

- Source_iOS/APDUResponse.h

15.4 ApplicationID Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- unsigned char acquirerIdentifier [6]

  Indicates which acquirer/processor processes the corresponding AID. Tag 9F01.
- unsigned char aid [16]

  AID value as per payment networks. Tag 9F06.
- unsigned char aidLen

  AID's length.
- unsigned char applicationSelectionIndicator

  Standard parameter.
- unsigned char applicationVersionNumber [2]

  EMV application version number. Tag 9F09.
- unsigned char XAmount [3]

  Not used by Agnos Framework.
- unsigned char YAmount [3]

  Not used by Agnos Framework.
- unsigned char skipTACIACDefault

  Indicates whether or not terminal uses default values for risk management.
- unsigned char tac

  Indicates whether or not terminal uses Terminal Action Code. 0x00 or 0x01.
- unsigned char floorlLimitChecking

  Indicates whether or not terminal uses Floor Limit Checking. 0x00 or 0x01.
- unsigned char randomTransactionSelection

  Indicates whether or not terminal uses Random Transaction Selection. 0x00 or 0x01.
- unsigned char velocitiyChecking

  Indicates whether or not terminal uses Velocity Checking. 0x00 or 0x01.
- unsigned char tACDenial [5]

  Terminal Action Code Denial.
- unsigned char tACOnline [5]

  Terminal Action Code Online.
- unsigned char tACDefault [5]

  Terminal Action Code Default.
- unsigned char terminalFloorLimit [3]

  Standard parameter. Tag 9F1B.
- unsigned char targetPercentage

  EMV offline risk management parameter.
- unsigned char thresholdValue [3]

  EMV offline risk management parameter.
- unsigned char maxTargetPercentage

  EMV offline risk management parameter.
- unsigned char defaultTDOL

  Standard parameter.
- unsigned char tdolValue [252]

  Transaction Data Object List value.
- unsigned char tdolLen

  Transaction Data Object List length.

- unsigned char defaultDDOL

  Standard parameter.. Tag.
- unsigned char ddolValue [252]

  Dynamic Data Object List value.
- unsigned char ddolLen

  Dynamic Data Object List length.
- unsigned char transactionCurrencyCode [2]

  AID's currency. Example: For Canada, {0x01,0x24}. Tag 5F2A.
- unsigned char transactionCurrencyExponent

  Transaction Currency Exponent. Example: Amount 4.53$ is managed as 453. Tag 5F36.

15.4.1 Detailed Description

device AID File - 571 bytes

Used as parameter in IDT_BTPay::emv_setApplicationData:()

Used as return value of aidResponse in IDT_BTPay::emv_retrieveApplicationData:response:()

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

- Source_iOS/IDTCommon.h

15.5 CAKey Struct Reference

```
#include <IDTCommon.h>
```

Public Attributes

- unsigned char hashAlgorithm

  Hash Algorithm 0x01 = SHA-1.
- unsigned char encryptionAlgorithm

  Encryption Algorithm 0x01 = RSA.
- unsigned char rid [5]

  As per payment networks definition.
- unsigned char index

  As per payment networks definition.
- unsigned char exponentLength

  Length of exponent. 0x01 or 0x03 as per EMV specs.
- unsigned char keyLength

  Length of key. max 248 bytes as per EMV specs.
- unsigned char exponent [3]

  CA Public Key Exponent.
- unsigned char key [248]

  CA Public Key.

15.5.1 Detailed Description

Certificate Authority Public Key

Used as parameter in IDT_BTPay::emv_retrieveCAPK:response:(), IDT_BTPay::emv_removeCAPK:(), IDT_BTPay::emv_setCAPK:(), IDT_UniPay::emv_retrieveCAPK:response:(), IDT_UniPay::emv_removeCAPK:(), IDT_UniPay::emv_setCAPK:()

Used as return value in IDT_BTPay::emv_retrieveCAPK:response:() IDT_UniPay::emv_retrieveCAPK:response:()

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

- Source_iOS/IDTCommon.h

15.6 CRLEntry Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- unsigned char rid [5]

  As per payment networks definition.
- unsigned char index

  As per payment networks definition.
- unsigned char serialNumber [3]

  As per payment networks definition.

15.6.1 Detailed Description

Certificate Revocation List Entry - 9 bytes

Used as parameter in IDT_BTPay::emv_retrieveCRLForRID:response:(), IDT_BTPay::emv_removeCRL:(), IDT_BTPay::emv_removeCRLUnit:(), IDT_BTPay::emv_setCRL:() IDT_UniPay::emv_retrieveCRLForRID:response:(), IDT_UniPay::emv_removeCRL:(), IDT_UniPay::emv_removeCRLUnit:(), IDT_UniPay::emv_setCRL:()

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

- Source_iOS/IDTCommon.h

15.7 ICCReaderStatus Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- bool iccPower

  Determines if ICC has been powered up.
- bool cardSeated

  Determines if card is inserted.
- bool latchClosed

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

- bool magneticDataPresent

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

15.7.1 Detailed Description

Structure used to return response from IDT_BTPay::icc_getICCReaderStatus() and IDT_UniPay::icc_getICC↩ ReaderStatus()

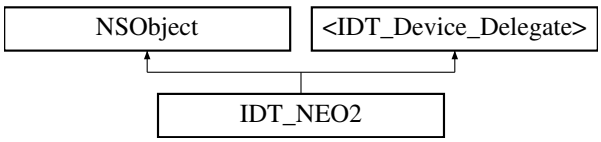
The documentation for this struct was generated from the following file:

- Source_iOS/IDTCommon.h

15.8 IDT_NEO2 Class Reference

`#import <IDT_NEO2.h>`

Inheritance diagram for IDT_NEO2:



Instance Methods

- (void) - close
- (RETURN_CODE) - ctls_cancelTransaction
- (RETURN_CODE) - emv_cancelTransaction
- (RETURN_CODE) - device_cancelTransaction
- (RETURN_CODE) - device_logClear
- (RETURN_CODE) - device_logRead:
- (RETURN_CODE) - device_logEnable:
- (RETURN_CODE) - device_getRT1050FirmwareVersion:
- (RETURN_CODE) - device_getBootloaderVersion:
- (RETURN_CODE) - device_rebootDevice
- (RETURN_CODE) - device_buzzerOnOff
- (RETURN_CODE) - ctls_getConfigurationGroup:response:
- (RETURN_CODE) - device_setMerchantRecord:enabled:merchantID:merchantURL:
- (RETURN_CODE) - device_getMerchantRecord:record:
- (RETURN_CODE) - device_getTransactionResults:
- (RETURN_CODE) - device_getDeviceTreeVersion:is1050:
- (RETURN_CODE) - device_get1050FuseStatus:
- (RETURN_CODE) - device_get1050BootloaderVersion:
- (RETURN_CODE) - device_get1050DeviceTreeVersion:
- (RETURN_CODE) - device_resetNVM
- (RETURN_CODE) - config_checkDUKPTKey:value:
- (RETURN_CODE) - config_getDEKVariantType:
- (RETURN_CODE) - config_setDEKVariantType:
- (RETURN_CODE) - config_getDUKPT_KSN:
- (RETURN_CODE) - config_getSalt_KCV:

- (RETURN_CODE) - config_setRKLKeys:tr31:nonce:hmac:kv:nonceDevice:hmacDevice:
- (RETURN_CODE) - config_setKeyslot_PEK_DEK:keyslot:
- (RETURN_CODE) - config_getKeyslot_PEK_DEK:keyslotDEK:
- (RETURN_CODE) - config_getDUKPT_DEK_Attribution:mode:outputModeWorkingKey:variantKeyUsage:
- (RETURN_CODE) - ctls_removeAllCAPK
- (RETURN_CODE) - ctls_removeApplicationData:
- (RETURN_CODE) - ctls_removeCAPK:
- (RETURN_CODE) - ctls_updateBalance:authCode:date:time:
- (RETURN_CODE) - ctls_getAllConfigGroups:
- (RETURN_CODE) - ctls_getAllConfigurationGroups:
- (RETURN_CODE) - ctls_removeConfigurationGroup:
- (RETURN_CODE) - ctls_retrieveAIDList:
- (RETURN_CODE) - ctls_retrieveApplicationData:response:
- (RETURN_CODE) - ctls_retrieveCAPK:key:
- (RETURN_CODE) - ctls_retrieveCAPKList:
- (RETURN_CODE) - ctls_retrieveTerminalData:
- (RETURN_CODE) - ctls_setApplicationData:
- (RETURN_CODE) - emv_setCAPK:
- (RETURN_CODE) - ctls_setCAPK:
- (RETURN_CODE) - ctls_setConfigurationGroup:
- (RETURN_CODE) - ctls_setTerminalData:
- (RETURN_CODE) - ctls_startTransaction:type:timeout:tags:
- (RETURN_CODE) - ctls_resetConfigurationGroup:
- (RETURN_CODE) - icc_getKeyFormatForICCDUKPT:
- (RETURN_CODE) - icc_setKeyFormatForICCDUKPT:
- (RETURN_CODE) - updateFirmwareNeo2:data:
- (RETURN_CODE) - device_getFirmwareVersion:
- (RETURN_CODE) - emv_getBatteryVoltage:
- (RETURN_CODE) - emv_getBatteryPercentage:
- (bool) - device_enableBLEDeviceSearch:
- (NSString ∗) - device_getBLEFriendlyName
- (void) - device_setBLEFriendlyName:
- (bool) - device_disableBLEDeviceSearch
- (NSUUID ∗) - device_connectedBLEDevice
- (void) - device_disconnectBLE
- (RETURN_CODE) - device_getAutoPollTransactionResults:
- (RETURN_CODE) - device_extendedErrorCondition:
- (NSString ∗) - device_getResponseCodeString:
- (bool) - device_isConnected:
- (RETURN_CODE) - device_sendIDGCommand:subCommand:data:response:
- (RETURN_CODE) - device_sendIDGCommandV3:subCommand:data:response:
- (void) - setServiceScanFilter:
- (RETURN_CODE) - device_setPassThrough:
- (RETURN_CODE) - device_pollForToken:card:serialNumber:
- (RETURN_CODE) - device_antennaControl:
- (RETURN_CODE) - device_exchangeContactlessData:receiveData:
- (RETURN_CODE) - device_setSpecialFunctionOrFeature:addRequirement:
- (RETURN_CODE) - device_getSpecialFunctionOrFeature:addRequirement:
- (RETURN_CODE) - device_setTerminalData:
- (RETURN_CODE) - device_retrieveTerminalData:
- (RETURN_CODE) - device_queryFile:filename:isSD:exists:timestamp:fileSize:
- (RETURN_CODE) - device_readFileFromSD:filename:fileData:
- (RETURN_CODE) - device_addTLVToTerminalData:
- (RETURN_CODE) - device_setBurstMode:
- (RETURN_CODE) - device_setPollMode:

- (RETURN_CODE) - emv_authenticateTransaction:
- (RETURN_CODE) - emv_callbackResponseLCD:selection:
- (RETURN_CODE) - emv_callbackResponsePIN:KSN:PIN:
- (RETURN_CODE) - emv_completeOnlineEMVTransaction:hostResponseTags:
- (void) - emv_disableAutoAuthenticateTransaction:
- (RETURN_CODE) - emv_getEMVL2Version:
- (RETURN_CODE) - emv_removeApplicationData:
- (RETURN_CODE) - emv_removeCAPK:index:
- (RETURN_CODE) - device_getPollMode:
- (RETURN_CODE) - device_getTransArmorID:
- (RETURN_CODE) - emv_getEMVKernelCheckValue:
- (RETURN_CODE) - emv_getEMVKernelVersion:
- (RETURN_CODE) - emv_callbackResponsePIN_ETC:ksn:pin:
- (RETURN_CODE) - emv_callbackResponseKSN:
- (RETURN_CODE) - emv_verifyDUKPTLoaded:
- (RETURN_CODE) - emv_getEMVKernelVersionExt:
- (RETURN_CODE) - emv_removeAllApplicationData
- (RETURN_CODE) - device_enableL80PassThrough:
- (RETURN_CODE) - device_getProductType:
- (RETURN_CODE) - device_getProcessorType:
- (RETURN_CODE) - device_getHardwareInfo:
- (RETURN_CODE) - device_getUIDofMCU:
- (RETURN_CODE) - device_pingDevice
- (RETURN_CODE) - device_enableL100PassThrough:
- (RETURN_CODE) - device_setTransArmorID:
- (RETURN_CODE) - device_listenForNotifications:
- (RETURN_CODE) - device_controlLED:control:
- (RETURN_CODE) - device_certificateType:
- (RETURN_CODE) - device_deleteFile:isSD:
- (RETURN_CODE) - device_deleteDirectory:
- (RETURN_CODE) - device_listDirectory:recursive:onSD:directory:
- (RETURN_CODE) - device_createDirectory:
- (RETURN_CODE) - device_lowPowerMode:wakeOnTrans:
- (RETURN_CODE) - device_controlUserInterface:
- (RETURN_CODE) - device_loadCertCA:CertData:
- (RETURN_CODE) - device_rrcConnect
- (RETURN_CODE) - device_rrcDisconnect
- (RETURN_CODE) - device_rrcRunApp:
- (RETURN_CODE) - device_rrcInstallApp:
- (RETURN_CODE) - device_rrcUninstallApp:
- (RETURN_CODE) - device_rrcDownloadApp:appData:
- (RETURN_CODE) - device_getMsrSecurePara:b1:b2:b3:tlv:
- (RETURN_CODE) - device_getModuleVer:
- (RETURN_CODE) - device_disBlueLED
- (RETURN_CODE) - device_enaBlueLED:
- (RETURN_CODE) - device_onYellowLED
- (RETURN_CODE) - device_offYellowLED
- (RETURN_CODE) - device_enterStandbyMode
- (RETURN_CODE) - device_getLightSensorVal:
- (RETURN_CODE) - device_setTransArmorEncryption:
- (RETURN_CODE) - adf_getModuleBytes:adfInfo:
- (RETURN_CODE) - adf_getModuleInfo:adfInfo:
- (RETURN_CODE) - adf_eraseFlash:
- (RETURN_CODE) - adf_setJTAG:
- (RETURN_CODE) - adf_setADFMode:

- (RETURN_CODE) - adf_getADFMode:
- (RETURN_CODE) - msr_getMSRTrack:
- (RETURN_CODE) - msr_getConfiguration:
- (RETURN_CODE) - msr_setConfiguration:
- (RETURN_CODE) - msr_retrieveWhiteList:
- (RETURN_CODE) - msr_setMSRTrack:
- (RETURN_CODE) - emv_setTerminalMajorConfiguration:
- (RETURN_CODE) - emv_getTerminalMajorConfiguration:
- (RETURN_CODE) - emv_removeCRLList
- (RETURN_CODE) - emv_removeTerminalData
- (RETURN_CODE) - emv_retrieveAIDList:
- (RETURN_CODE) - emv_retrieveApplicationData:response:
- (RETURN_CODE) - emv_retrieveCAPKFile:index:response:
- (RETURN_CODE) - emv_retrieveCAPKList:
- (RETURN_CODE) - emv_retrieveCAPK:index:response:
- (RETURN_CODE) - emv_retrieveCRLList:
- (RETURN_CODE) - emv_retrieveTerminalData:
- (RETURN_CODE) - emv_retrieveTransactionResult:retrievedTags:
- (RETURN_CODE) - emv_setApplicationData:configData:
- (RETURN_CODE) - emv_setCAPKFile:
- (RETURN_CODE) - emv_setCRLEntries:
- (RETURN_CODE) - emv_setTerminalData:
- (RETURN_CODE) - emv_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:
- (RETURN_CODE) - emv_exchangeCerts:nonce:signature:
- (RETURN_CODE) - emv_getEMVConfigurationCheckValue:
- (RETURN_CODE) - emv_callbackResponseMSR:
- (RETURN_CODE) - emv_generateDUKPT:signature:key:
- (RETURN_CODE) - config_getSerialNumber:
- (RETURN_CODE) - icc_exchangeAPDU:response:
- (RETURN_CODE) - icc_getICCReaderStatus:
- (RETURN_CODE) - icc_powerOnICC:
- (RETURN_CODE) - icc_powerOffICC:
- (RETURN_CODE) - msr_cancelMSRSwipe
- (RETURN_CODE) - msr_startMSRSwipe
- (bool) - isConnected
- (RETURN_CODE) - device_startTransaction:type:timeout:tags:
- (RETURN_CODE) - pin_captureFunctionKey
- (RETURN_CODE) - pin_cancelPin
- (RETURN_CODE) - pin_capturePin:PAN:minPIN:maxPIN:message:
- (RETURN_CODE) - config_setBluetoothParameters:oldPW:newPW:
- (RETURN_CODE) - felica_authentication:
- (RETURN_CODE) - felica_readWithMac:blockList:blocks:
- (RETURN_CODE) - felica_SendCommand:response:
- (RETURN_CODE) - felica_writeWithMac:data:
- (RETURN_CODE) - felica_read:numBlocks:blockList:blocks:
- (RETURN_CODE) - felica_write:blockCount:blockList:data:statusFlag:
- (RETURN_CODE) - ctls_nfcCommand:response:
- (void) - scanForBLEDevices:serviceUUIDs:options:
- (void) - scanForBLEDeviceNames:serviceUUIDs:options:
- (RETURN_CODE) - felica_requestService:response:
- (void) - setServiceUUID:
- (RETURN_CODE) - pin_captureAmountInput:maxPIN:message:signature:
- (RETURN_CODE) - pin_captureNumericInput:minPIN:maxPIN:message:signature:

Class Methods

- (NSString ∗) + SDK_version
- (IDT_NEO2 ∗) + sharedController
- (NSString ∗) + createFastEMVData:

Properties

- id< IDT_NEO2_Delegate > delegate

15.8.1 Detailed Description

Class to drive the IDT_NEO2 device

15.8.2 Method Documentation

15.8.2.1 - (RETURN_CODE) adf_eraseFlash: (ADF_TYPE) *type*

Erase ADF Flash

Erases the ADF flash memory

Parameters

type	The ADF type • ADF_TYPE_SDK = SDK • ADF_TYPE_APP = Application

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.2 - (RETURN_CODE) adf_getADFMode: (BOOL ∗) *enable*

Get ADF Mode

Retrieves the state of the ADF environment

Parameters

enable	True = ADF is enabled, False = ADF is disabled

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.3 - (RETURN_CODE) adf_getModuleBytes: (ADF_TYPE) *type* adfInfo:(NSArray< NSData ∗ > ∗∗) *adfInfo*

Get Module Bytes

Retrieves the first 64 bytes of the module information running in an ADF environment

Parameters

type	The ADF type • ADF_TYPE_SDK = SDK • ADF_TYPE_APP = Application
adfInfo	List of modules information

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.4 - (RETURN_CODE) adf_getModuleInfo: (ADF_TYPE) *type* adfInfo:(NSArray ∗∗) *adfInfo*

Get Module Info

Retrieves the module information when running in an ADF environment

Parameters

type	The ADF type • ADF_TYPE_SDK = SDK • ADF_TYPE_APP = Application
adfInfo	List of modules information

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.5 - (RETURN_CODE) adf_setADFMode: (BOOL) *enable*

Set ADF Mode

Enables/Disables the ADF environment

Parameters

enable	True = ADF enabled, False = ADF disabled

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.6 - (RETURN_CODE) adf_setJTAG: (BOOL) *enable*

Set JTAG

Enables/Disables the JTAG pin in an ADF environment

Parameters

enable	True = JTAG enabled, False = JTAG disabled

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.7 - (void) close

Close Device

15.8.2.8 - (RETURN_CODE) config_checkDUKPTKey: (Byte) *keyIndex* value:(NSData **) *val*

Check DUKPT Key

This command checks whether a valid DUKPT key is stored at the specified slot and if a valid key is found then some basic information related to the type of key is returned. The actual key data is never returned. This command can be used to check whether a key is already present before injecting a key in a KeyIndex to prevent overwriting an existing DUKPT key.

Parameters

keyIndex	Data Encryption key (usually 5)
val	-Byte 0 = Key State: 00h = Unused, 01h = Valid, 02h = End of Life, FFh = Not Available -Byte 1-2 = Key Usage (ASCII). "D0" = Used to encrypt data -Byte 3 = Algorithm (ASCII). "T" = Triple DES -Byte 4 = Mode of Use (ASCII). "E" = Encryption Only -Byte 5-6 = Key Version (ASCII). "00" = Key version not used

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.9 - (RETURN_CODE) config_getDEKVariantType: (Byte *) *type*

Get Data Encryption Key Variant Type

Returns the data encryption key type

Parameters

type	-0 = Data Variant -1 = Pin Variant

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.10 - (RETURN_CODE) config_getDUKPT_DEK_Attribution: (Byte) *keyslot* mode:(Byte *) *mode* outputModeWorkingKey:(Byte *) *outputMode_workingKey* variantKeyUsage:(Byte *) *variant_keyUsage*

Get DUKPT DEK Attribution based on KeySlot

Command settings can only be changed for each key one time

Parameters

keyslot	Key Slot
mode	0 = TDES, 1 = AES
outputMode_workingKey	0 = TDES / TDES, 1 = AES-128 / TDES3 2 = TransArmor / AES-128, Output Mode is when TDES, Working Key is when AES
variant_keyUsage	0 = Data Key / Encrypt/Decrypt, 1 = PIN Key / Encrypt, 2 = N/A / Decrypt, Variant is when TDES, Key Usage is when AES

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.11 - (RETURN_CODE) config_getDUKPT_KSN: (NSData ∗∗) *KSN*

Get DUKPT Key Serial Number (KSN)

Host can use this command to retrieve the KSN of the DUKPT key.

Parameters

KSN	Key Serial Number

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.12 - (RETURN_CODE) config_getKeyslot_PEK_DEK: (NSData ∗∗) *keyslotPEK* keyslotDEK:(NSData ∗∗) *keyslotDEK*

Get KeySlot for DUKPT PEK and DUKPT DEK

Returns the KeySlot for PEK and DEK (if they exist)

Parameters

keyslotPEK	PEK KeySlot (0-9). Value of 255 (0xff) = does not exist
keyslotDEK	DEK KeySlot (0-9). Value of 255 (0xff) = does not exist

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.13 - (RETURN_CODE) config_getSalt_KCV: (NSData ∗∗) *KCV*

Get Salt KCV

Host can use this command to retrieve the Salt KCV

Parameters

KCV	KCV

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.14 - (RETURN_CODE) config_getSerialNumber: (NSString **) *response*

Polls device for Serial Number

Parameters

response	Returns Serial Number

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.15 - (RETURN_CODE) config_setBluetoothParameters: (NSString *) *name* oldPW:(NSString *) *oldPW* newPW:(NSString *) *newPW*

Set BluetoothParameters

Sets the name and password for the BLE module.

Sending nil to all three parameters resets the default password to 123456

Parameters

name	Device name, 1-25 characters
oldPW	Old password, as a six character string, example "123456"
newPW	New password, as a six character string, example "654321"

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.16 - (RETURN_CODE) config_setDEKVariantType: (Byte) *type*

Set Data Encryption Key Variant Type

This command exists to specify the key variant type of Data encryption key (Slot = 0), and MUST be used before the initial loading of the Data encryption key into the device. The key variant type CANNOT be changed once the Data encryption key is present. It must remain either Data variant or PIN variant.

Parameters

type	-0 = Data Variant -1 = PIN Variant

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.17 - (RETURN_CODE) config_setKeyslot_PEK_DEK: (BOOL) *isPEK* keyslot:(Byte) *keySlot*

Set KeySlot for DUKPT PEK and DUKPT DEK

Selects one of the available PEKs for use in subsequent PIN encryption operations. Selects one of the available DEKs for use in subsequent Data encryption operations.

Parameters

isPEK	TRUE = set PEK, FALSE = set DEK
keySlot	The KeySlot number specifies the key this command will select

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.18 - (RETURN_CODE) config_setRKLKeys: (short) *keyNumber* tr31:(NSData ∗) *tr31* nonce:(NSData ∗) *nonce* hmac:(NSData ∗) *hmac* kv:(NSData ∗∗) *kv* nonceDevice:(NSData ∗∗) *nonce_device* hmacDevice:(NSData ∗∗) *hmac_device*

Set RKL Keys

Sets the RKL Keys

Parameters

keyNumber	Number of keys remaining to load
tr31	ASN.1 structure of encrypted key(s)
nonce	Nonce generated by RKMS to generate an HMAC used for auth
hmac	HMAC-SHA256 generated from RKMS
kv	ASN.1 Key Verification Structure returned from device
nonce_device	Nonce generated by the device to generate a MAC used for auth
hmac_device	HMAC-SHA256 generated from device terminal

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.19 + (NSString∗) createFastEMVData: (IDTEMVData ∗) *emvData*

Create Fast EMV Data

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

Parameters

emvData	The IDTEMVData object populated with card data.

Returns

Fast EMV String data

15.8.2.20 - (RETURN_CODE) ctls_cancelTransaction

Cancels Transaction request (all interfaces, CTLS/MSR/INSERT).

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.21 - (RETURN_CODE) ctls_getAllConfigGroups: (NSData **) *response*

Get All Configuration Groups

Retrieves all Configuration Groups.

Parameters

response	Groups returned as Data stream

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.22 - (RETURN_CODE) ctls_getAllConfigurationGroups: (NSDictionary< NSString *, NSDictionary * > **) *response*

Get All Configuration Groups

Retrieves all Configuration Groups.

Parameters

response	Groups returned in a dictionary Key = <String>Group Value = <Dictionary<tag,value>>

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.23 - (RETURN_CODE) ctls_getConfigurationGroup: (int) *group* response:(NSDictionary **) *response*

Get Configuration Group

Retrieves the Configuration for the specified Group. Group 0 = terminal settings.

Parameters

group	Configuration Group
response	Group TLV returned as a dictionary

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.24 - (RETURN_CODE) ctls_nfcCommand: (NSData ∗) *systemCode* response:(NSData ∗∗) *response*

NFC Command

This command uses nfcCmdPkt[0] in command data field to implement different functions. This command should be used in Pass-Through mode and command with “Poll for a NFC Tag” data should be used first. Command with other data can only be used once the “Poll for a NFC Tag” command has indicated that a NFC tag is present.

Parameters

nfcCmdPkt	System Code • Poll for NFC Tag: nfcCmdPkt[0] = 0xff, nfcCmdPkt[1] = timeout value (in seconds) • Tag1 Static Get All Data: nfcCmdPkt[0] = 0x11 • Tag1 Static Read a Byte: nfcCmdPkt[0] = 0x12, nfcCmdPkt[1] = Address of Data • Tag1 Static Write a Byte: nfcCmdPkt[0] = 0x13 nfcCmdPkt[1] = Address of Data, nfcCmdPkt[2] = Data to be written • Tag1 Static Write a Byte NE: nfcCmdPkt[0] = 0x14, nfcCmdPkt[1] = Address of Data, nfcCmdPkt[2] = Data to be written • Tag1 Dynamic Read a Segment: nfcCmdPkt[0] = 0x15, nfcCmdPkt[1] = Address of Segment • Tag1 Dynamic Read 8 Bytes: nfcCmdPkt[0] = 0x16, nfcCmdPkt[1] = Address of Data • Tag1 Dynamic Write 8 Bytes: nfcCmdPkt[0] = 0x17, nfcCmdPkt[1] = Address of Data, nfcCmdPkt[2]∼nfcCmdPkt[9] = Data to be written • Tag1 Dynamic Write 8 Bytes NE: nfcCmdPkt[0] = 0x18, nfcCmdPkt[1] = Address of Data, nfcCmdPkt[2]∼nfcCmdPkt[9] = Data to be written • Tag2 Read Data (16 bytes): nfcCmdPkt[0] = 0x21, nfcCmdPkt[1] = Address of Data • Tag2 Write Data (4 bytes): nfcCmdPkt[0] = 0x22, nfcCmdPkt[1] = Address of Data, nfcCmdPkt[2]∼nfcCmdPkt[5] = Data to be written • Tag2 Select Sect: nfcCmdPkt[0] = 0x23, nfcCmdPkt[1] = Sect number • Tag3 Read Data: – nfcCmdPkt[0] = 0x41, – nfcCmdPkt[1] = Number of services, value n – nfcCmdPkt[2]∼nfcCmdPkt[2n+1]: Service code list – nfcCmdPkt[2n+2]: Number of blocks, value m. – nfcCmdPkt[2n+3....]: Block list, length is 2m∼3m • Tag3 Write Data: – nfcCmdPkt[0] = 0x41, – nfcCmdPkt[1] = Number of services, value n – nfcCmdPkt[2]∼nfcCmdPkt[2n+1]: Service code list – nfcCmdPkt[2n+2]: Number of blocks, value m. – nfcCmdPkt[2n+3....]: Block list, length is 2m∼3m – nfcCmdPkt[...]: Block data, length is 16m • Tag4 Command: nfcCmdPkt[0] = 0x81, nfcCmdPkt[1]∼nfcCmdPkt[n]:data
response	Response as explained in FeliCA Lite-S User's Manual
ip	IP Address of target device (optional)

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.25 - (RETURN_CODE) ctls_removeAllCAPK

Remove All Certificate Authority Public Key

Removes all the CAPK for CTLS

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:
RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.26 - (RETURN_CODE) ctls_removeApplicationData: (NSString ∗) *AID*

Remove Application Data by AID

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

Parameters

AID	Name of ApplicationID as Hex String (example "A0000000031010") Must be between 5 and 16 bytes

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.27 - (RETURN_CODE) ctls_removeCAPK: (NSData ∗) *capk*

Remove Certificate Authority Public Key

Removes the CAPK for CTLS as specified by the RID/Index

Parameters

capk	6 byte CAPK = 5 bytes RID + 1 byte INDEX

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.28 - (RETURN_CODE) ctls_removeConfigurationGroup: (int) *group*

Remove Configuration Group

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

Parameters

group	Configuration Group

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.29 - (RETURN_CODE) ctls_resetConfigurationGroup: (int) *group*

Reset Configuration Group

This command allows resetting a dataset to its default configuration If the file exists, it will be overwritten. If not, it will be created

Parameters

group	Configuration group

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_VP3300::device_getResponseCodeString:()

15.8.2.30 - (RETURN_CODE) ctls_retrieveAIDList: (NSArray ∗∗) *response*

Retrieve AID list

Returns all the AID names installed on the terminal for CTLS.

Parameters

response	array of AID name as NSData∗

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.31 - (RETURN_CODE) ctls_retrieveApplicationData: (NSString ∗) *AID* response:(NSDictionary ∗∗) *response*

Retrieve Application Data by AID

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

Parameters

AID	Name of ApplicationID as Hex String (example "A0000000031010"). Must be between 5 and 16 bytes
response	The TLV elements of the requested AID

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.32 - (RETURN_CODE) ctls_retrieveCAPK: (NSData *) *capk* key:(NSData **) *key*

Retrieve Certificate Authority Public Key

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

Parameters

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

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.33 - (RETURN_CODE) ctls_retrieveCAPKList: (NSArray **) *keys*

Retrieve the Certificate Authority Public Key list

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

Parameters

keys	NSArray of NSData CAPK name (RID + Index)

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.34 - (RETURN_CODE) ctls_retrieveTerminalData: (NSData **) *tlv*

Retrieve Terminal Data

Retrieves the Terminal Data for CTLS. This is configuration group 0 (Tag DFEE2D - > DFEE2D0100). The terminal data can also be retrieved by ctls_getConfigurationGroup(0).

Parameters

tlv	Response returned as a TLV

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.35 - (RETURN_CODE) ctls_setApplicationData: (NSData *) *tlv*

Set Application Data by AID

Sets the Application Data for CTLS as specified by TLV data

Parameters

tlv	Application data in TLV format The first tag of the TLV data must be the group number (DFEE2D). The second tag of the TLV data must be the AID (9F06)

Example valid TLV, for Group #2, AID a0000000035010: "DFEE2D01029f0607a0000000051010DFEE4B0101D↩FEE2E0110DFEE4D0114DFEE4C0106"

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.36 - (RETURN_CODE) ctls_setCAPK: (NSData *) *key*

Set Certificate Authority Public Key

Sets the CAPK for CTLS as specified by the CAKey structure

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.37 - (RETURN_CODE) ctls_setConfigurationGroup: (NSData ∗) *tlv*

Set Configuration Group

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

Parameters

tlv	Configuration Group Data in TLV format The first tag of the TLV data must be the group number (DFEE2D). A second tag must exist

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.38 - (RETURN_CODE) ctls_setTerminalData: (NSData ∗) *tlv*

Set Terminal Data

Sets the Terminal Data for CTLS transaction and general terminal settings as specified by the TLV. If the value already exists in terminal data, it will be updated. If the value does not exist, it will be added.

Parameters

tlv	TerminalData configuration data

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_VP8800::device_getResponseCodeString:()

15.8.2.39 - (RETURN_CODE) ctls_startTransaction: (double) *amount* type:(int) *type* timeout:(int) *timeout* tags:(NSMutableDictionary ∗) *tags*

Start a CTLS Transaction Request

Authorizes the CTLS transaction for an CTLS card

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)
exponent	Number of characters after decimile point
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request.

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

NOTE ON INTERFACE SELECTION: For the NEO2, tag DFEF37 is used to determine which interfaces to use for the transaction:

- 01 = MSR Only
- 02 = CTLS Only
- 03 = MSR + CTLS
- 04 = EMV Only
- 05 = EMV + MSR
- 06 = EMV + CTLS
- 07 = EMV + MSR + CLTS. This API method will automatically send DFEF37 with a value of 02 if this tag is not provided.

15.8.2.40 - (RETURN_CODE) ctls_updateBalance: (NSData *) *statusCode* authCode:(NSData *) *authCode* date:(NSData *) *date* time:(NSData *) *time*

Update Balance

This command is the authorization respose sent by the issuer to the terminal including the Authorization Status (OK or NOT OK).

This command is also being ussed in some implementations (i.e. EMEA) to communicate the results of Issuer Authorization to the reader in order to display the correct LCD messages. With this command, the POS passes the authorization result (OK/NOT OK), and possibly the Authorization Code (authCode)/Date/Time to the terminal.

Parameters

statusCode	00:ok, 01:not ok, 02: (ARC response 89 for Interac)
authCode	Authorization code from host. Six bytes. Optional
date	Transaction date. If null, uses current terminal date. 3 bytes compressed numeric YYMMDD (tag value 9A)
time	Transaction time. If null, uses current terminal time. 3 bytes compressed numeric HHMMSS (tag value 9F21)

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.41 - (RETURN_CODE) device_addTLVToTerminalData: (NSData *) *tlv*

Add Terminal Data

Adds the specified TLV to the current terminal data .

Parameters

tlv	The data to set, overwriting any existing value

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.42 - (RETURN_CODE) device_antennaControl: (bool) *turnON*

Antenna Control

The Antenna Control command turns the RF Antenna ON or OFF.

Parameters

turnON	TRUE = ON, FALSE = OFF

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.43 - (RETURN_CODE) device_buzzerOnOff

Buzzer OnOff

Causes the buzzer to beep once

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.44 - (RETURN_CODE) device_cancelTransaction

Cancels Transaction request (all interfaces, CTLS/MSR/INSERT).

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.45 - (RETURN_CODE) device_certificateType: (int *) *type*

Device Certificate Type

Returns the device certificate type

Parameters

type	0 = Unknown, 1 = Demo, 2 = Production

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.46 - (NSUUID∗) device_connectedBLEDevice

Returns the UUID of the connected BLE device

- NEO2

  Returns

  NSUUID UUID of the connected BLE device. Returns nil if no BLE device connected.

15.8.2.47 - (RETURN_CODE) device_controlLED: (Byte) *indexLED* control:(Byte) *control*

Control LED

Controls the LED for the reader. This command will only operate in pass-through mode

Parameters

indexLED	The LED to control starting from the left • 00: LED 0 • 01: LED 1 • 02: LED 2 • 03: LED 3 • FF: All LEDs
control	Turns chosen LED(s) OFF (00) or ON (01)

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.48 - (RETURN_CODE) device_controlUserInterface: (NSData ∗) *values*

Control User Interface

Controls the Display, Beep, and LED

Parameters

values	Four bytes to control the user interface elements Byte[0] = LCD Message • 00h: Idle Message (Welcome) • 01h: Present Card (Please Present Card) • 02h: Timeout or Transaction Canceled (No Card) • 03h: Transaction between reader and card is in progress (Processing...) • 04h: Transaction success (Thank You) • 05h: Transaction failed (Failed) • 06h: Amount (Amount $ 0.00 Tap Card) • 07h: Balance or Offline available funds (Balance $ 0.00) • 08h: Insert card (Use Chip & PIN) • 09h: Try again (Tap Again) • 0Ah: Tells the customer to present only one card (Present 1 Card Only) • 0Bh: Tells the customer to wait for authentication/authorization (Wait) • FFh: Indicates the command is setting the LED/Buzzer only Byte[1] = Beep Indicator • 00h: No beep • 01h: Single beep • 02h: Double beep • 03h: Triple beep • 04h: Quadruple beep • 05h: Single long beep (200 ms) • 06h: Single long beep (400 ms) • 07h: Single long beep (600 ms) • 08h: Single long beep (800 ms) Byte[2] = LED • 00h: LED 0 (Power LED) • 01h: LED 1 • 02h: LED 2 • 03h: LED 3 • FFh: All LEDs Byte[3] = LED Power • 00h: LED OFF • 01h: LED ON

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.49 - (RETURN_CODE) device_createDirectory: (NSString *) *directoryName*

Create Directory

This command adds a subdirectory to the indicated path

Parameters

directoryName	The directory name. The data for this command is an ASCII string with the complete path and directory name you want to create. You do not need to specify the root directory. Indicate subdirectories with a forward slash (/)

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.50 - (RETURN_CODE) device_deleteDirectory: (NSString *) *filename*

Delete Directory

This command deletes an empty directory

Parameters

filename	Complete path of the directory you want to delete.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.51 - (RETURN_CODE) device_deleteFile: (NSString *) *filename* isSD:(BOOL) *isSD*

Delete File

This command deletes a file or group of files

Parameters

filename	Complete path and file name of the file you want to delete. You do not need to specify the root directory. Indicate subdirectories with a foward slash (/)
isSD	True = Delete from SD card, False = Delete from Flash

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.52 - (bool) device_disableBLEDeviceSearch

Stops searching for Bluetooth Low Energy devices in range

- NEO2

Returns

bool If successful, polling was in progress and has stopped. If unsuccessful, BLE Device Search was not in progress.

NOTE: BLE only scans when there are no devices currently connected. After the SDK connects to any IDTech device, the scanning will pause automatically.

15.8.2.53 - (RETURN_CODE) device_disBlueLED

Disable Blue LED Sequence

Stop the blue LEDs on the ViVOpay Vendi reader from flashing in the left to right sequence and the LEDs off. Contactless function is also disabled at the same time

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.54 - (void) device_disconnectBLE

Disconnect from BLE -

Will disconnect from existing BLE connection. You can now set another BLE Friendly Name to attach to another device.

15.8.2.55 - (bool) device_enableBLEDeviceSearch: (NSUUID ∗) *identifier*

Begins searching for Bluetooth Low Energy devices in range

- NEO2

  This method will UNPAIR and DISCONNECT from any current PAIRED and CONNECTED devices before starting a new BLE Device Search. This function is for PAIRING a new Device and connecting to it. Once a device is PAIRED/CONNECTED, the search will terminate. If a paired devices becomes disconnected (out of range, power cycle), the BLE search will automatically start again to reconnect to the devices once available

Parameters

identifier	This will only connect to a device with this calculated UUID identifier. If nil is passed, it will connect to the first devices with the default friendly name (device_getBLEFriendlyName / IDT_NEO2::device_setBLEFriendlyName)

Returns

bool If successful, polling has started

Any of the following BLE status messages may be returned to the deviceMessage delegate:

- This device does not support Bluetooth Low Energy.
- This app is not authorized to use Bluetooth Low Energy.
- Bluetooth on this device is currently powered off.
- The BLE Manager is resetting; a state update is pending.
- Bluetooth LE is turned on and ready for communication.

- The state of the BLE Manager is unknown.

  Note: a Devices UUID is calculated by the iOS device using a combiniation of the iOS device UUID and the BLE device MAC address. This value is not known until after it connects for the first time, and then every time after that, it will be the same value. This value can be retrieved by IDT_Device::connectedBLEDevice() after the device connects.

15.8.2.56 - (RETURN_CODE) device_enableL100PassThrough: (BOOL) *enablePassThrough*

Enable L100 Passthrough

Enable passthrough mode for direct communication to L100 hooked up to NEO2 device

Parameters

enablePassThrough	True = passthrough ON, False = passthrough OFF

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.57 - (RETURN_CODE) device_enableL80PassThrough: (BOOL) *enablePassThrough*

Enable L80 Passthrough

Enables Passthrough mode for direct communicaiton to L80 hooked up to NEO2 device

Parameters

enablePassThrough	True = passthrough ON, False = passthrough OFF

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.58 - (RETURN_CODE) device_enaBlueLED: (NSData ∗) *dataCmd*

Enable the blue LED and modify the behaviour

Control the blue LED behaviour on the Vendi reader

Parameters

dataCmd	LED control. Minimum 4 bytes, maximum 25 bytes. First byte is cycle, next three bytes are the sequence. Then sequence can repeat up to 8 times. • Byte 0 = Cycle (0 = Cycle once, 1 = Repeat) • Byte 1 = LED state map – bit 7 = Left blue LED – bit 6 = Center blue LED – bit 5 = Right blue LED – bit 4 = Yellow LED – bit 3 = Reserved – bit 2 = Reserved – bit 1 = Reserved – bit 0 = Reserved • Bytes 2 & 3 = Duration (Given in multiples of 10 ms, i.e. 10/20/30 etc.) If Cycle = 1, more pairs be after 3

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.59 - (RETURN_CODE) device_enterStandbyMode

Enter standby mode

Puts unit into low power standby mode

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.60 - (RETURN_CODE) device_exchangeContactlessData: (NSData ∗) *sendData* receiveData:(NSData ∗∗) *receiveData*

Exchange Contactless Data

The Echange Contactless Data command allows the host device to send, via the ViVOpay reader, application-level APDUs to a PICC that supports ISO 14443-4 Protocol. The reader sends the PICC response back to the host

Parameters

sendData	APDU Out
receiveData	APDU response

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.61 - (RETURN_CODE) device_extendedErrorCondition: (BOOL) *enable*

Extended Error Condition

Enables/disables extended error condition for commands 02-40, 61-xx, 62-xx, 83-41 when error is 0xd0a or 0xd0b

Parameters

enable	TRUE = enable log, FALSE = disable log

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.62 - (RETURN_CODE) device_get1050BootloaderVersion: (NSString $**$) *version*

Get 1050 Bootloader Version

Polls the device for the 1050 bootloader version

Parameters

version	Response returned of the 1050 bootloader version

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.63 - (RETURN_CODE) device_get1050DeviceTreeVersion: (NSString $**$) *deviceTree*

Get 1050 Device Tree

Polls the NEO2/3 for the 1050 Device Tree

Parameters

deviceTree	The device tree returned as a NSString

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.64 - (RETURN_CODE) device_get1050FuseStatus: (NSData $**$) *status*

RT1050 SRK Fuse Status

This command retrieves the status fo the reader's RT1050 SRK fuse

Parameters

status	First 8 bytes of the SRK Fuse, if programmed. Otherwise, an empty array is returned

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.65 - (RETURN_CODE) device_getAutoPollTransactionResults: (IDTEMVData **) *result*

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

Parameters

result	The transaction results

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:. When no data is available, return code = RETURN_CODE_NO_DATA_AVAILABLE

15.8.2.66 - (NSString*) device_getBLEFriendlyName

Get BLE Friendly Name

Returns

NSString Returns the default friendly name to be used when discovering any BLE devices

15.8.2.67 - (RETURN_CODE) device_getBootloaderVersion: (NSString **) *response*

Get Bootloader Version

Polls the device for the bootloader version

Parameters

response	Response return of the bootloader version from the device

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.68 - (RETURN_CODE) device_getDeviceTreeVersion: (NSString **) *deviceTree* is1050:(BOOL) *is1050*

Get Device Tree

Polls the NEO2/3 for the Device Tree

Parameters

deviceTree	The device's tree returned as a NSString

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.69 - (RETURN_CODE) device_getFirmwareVersion: (NSString **) *response*

Polls device for Firmware Version

Parameters

response	Response returned of Firmware Version

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPay::device_getResponseCodeString:()

15.8.2.70 - (RETURN_CODE) device_getHardwareInfo: (NSString **) *response*

Get Hardware Info

Returns an ASCII string for the hardware information

response The ASCII character string

ASCII | Description

HW,VPVendi<CR><LF>K21F Rev9 | Vendi HW,VP3300 Audio Jack<CR><LF>K21F Rev9 | Unipay III HW,V↩PUnipay1.5<CR><LF>K21F Rev9 | Unipay 1.5 HW,VPUniPay1.5TTK<CR><LF>K21F Rev9 | UniPay 1.5 TTK HW,VP3300 USB<CR><LF>K21F Rev9 | VP3300 USB, VP3300 USB OEM (iBase/Cake same code, string ident = "") HW,VP3300 USB-E<CR><LF>K21F Rev9 | VP3300 USB-E HW,VP3300 USB-C<CR><LF>K21F Rev9 | VP3300 USB-C HW,VPVP3300 Bluetooth<CR><LF>K21F Rev9 | VP3300 Bluetooth HW,.VP6300<CR><L↩F>K81F.Rev4 | VP6300

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.71 - (RETURN_CODE) device_getLightSensorVal: (UInt16 *) *lightVal*

Get Light Sensor Value

Gets the value from the sensor

Parameters

lightVal	Value of the light sensor

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.72 - (RETURN_CODE) device_getMerchantRecord: (int) *index* record:(NSData ∗∗) *record*

Get Merchant Record Gets the merchant record for ApplePay VAS

Parameters

index	Merchant Record index, valid values 1-6
record	Data returned containing 99 bytes: Byte 0 = Merchand Index Byte 1 = Merchant Enabled (1 = enabled) Byte 2 - 33 = Merchant Protocol Hash-256 value Byte 34 = Length of Merchant URL Bytes 35 - 99 = URL

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.73 - (RETURN_CODE) device_getModuleVer: (NSString ∗∗) *moduleVer*

Get Module Version Information

Gets the 16 byte UID of the MCU

Parameters

uid	The string representation of the UID

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.74 - (RETURN_CODE) device_getMsrSecurePara: (BOOL) *b0* b1:(BOOL) *b1* b2:(BOOL) *b2* b3:(BOOL) *b3* tlv:(NSData ∗∗) *tlv*

Get MSR Secure Parameters

Gets the parameters from the flash setting

Parameters

b0	True = T1 force encryption
b1	True = T2 force encryption
b2	True = T3 force encryption
b3	True = T3 force encryption when card type is 80
tlv	MSR secure parameters TLV objects

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.75 - (RETURN_CODE) device_getPollMode: (NSData **) *mode*

Get Poll Mode

Gets the current poll mode of the device

Parameters

mode	Response from the device of the current poll mode

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.76 - (RETURN_CODE) device_getProcessorType: (NSData **) *type*

Get Processor Type

Returns a processor type TLV

type Processor type

Processor Type | Description

45 00 | ARM7/ LPC21xx 4D 00 | ARM Cortex-M4/ K21 Family 4E 00 | ARM Cortex-M4/ K81 Family

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.77 - (RETURN_CODE) device_getProductType: (NSData **) *type*

Get Product Type

Returns a "product type" value in a proprietary TLV

type Product type

Product Type | Description

42 37 00 | ViVOpay 5000 43 33 00 | ViVOpay 4500 43 35 00 | ViVOpay Vend 43 36 00 | Vendi (NEO, string ident = "") 43 37 00 | ViVOpay Kiosk1 (ATM1, string ident = "") 43 38 00 | Kiosk2 43 39 00 | Kiosk3 (NEO, string ident = "") 55 31 00 | UniPay 1.5 (NEO, string ident = "") 55 33 00 | UniPay III (NEO) 55 33 31 | VP3300, VP3300 OEM (NEO) (iBase/Cake same code, string ident = "") 55 33 32 | VP3300E(NEO, string ident = "") 55 33 33 | VP3300C(NEO, string ident = "") 55 33 34 | BTPay Mini (NEO) (UniPayIII + BLE, string ident = "") 56 31 00 | VP3600 56 32 00 | VP5200 56 33 00 | VP5300 56 34 00 | VP6300 56 35 00 | VP6800 56 36 00 | VP8300 56 37 00 | VP8310 56 38 00 | VP8800 56 39 00 | VP8810 56 40 00 | VP9000 44 30 00 | QX120 44 31 00 | Mx8Series 44 32 00 | NETs 44 33 00 | Magtek 44 35 00 | ICP

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.78 - (NSString ∗) device_getResponseCodeString: (int) *errorCode*

Get Response Code String

Interpret a response code and return string description.

Parameters

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

Returns

Verbose error description

15.8.2.79 - (RETURN_CODE) device_getRT1050FirmwareVersion: (NSString ∗∗) *response*

Get RT1050 Firmware Version

Gets the version for the RT1050 Firmware

Parameters

response	Response returned of the RT1050 firmware

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.80 - (RETURN_CODE) device_getSpecialFunctionOrFeature: (NSData ∗∗) *feature* addRequirement:(NSData ∗∗) *addRequirement*

Get Special Function or Feature Configuration Command

The Get Special Function or Feature Configuration command returns the specific special configuration.

Parameters

feature	Function/Feature ID
addRequirement	Additional Requirement

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.81 - (RETURN_CODE) device_getTransactionResults: (NSData ∗∗) *results*

Get Transaction Results

Gets the transaction results when the reader is functioning in "Auto Poll" mode

Parameters

results	The transaction results

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.82 - (RETURN_CODE) device_getTransArmorID: (NSString **) *TID*

Get TransArmor ID

Gets the TransArmor ID from the device

Parameters

TID	TransArmor ID

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.83 - (RETURN_CODE) device_getUIDofMCU: (NSString **) *response*

Get UID of MCU

Returns the UID of the device

Parameters

response	The module UID information

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.84 - (bool) device_isConnected: (IDT_DEVICE_Types) *device*

Is Device Connected

Returns the connection status of the requested device

Parameters

device	Check connectivity of device type

```
typedef enum{
IDT_DEVICE_NEO2_IOS = 16
}IDT_DEVICE_Types;
```

15.8.2.85 - (RETURN_CODE) device_listDirectory: (NSString ∗) *directoryName* recursive:(BOOL) *recursive* onSD:(BOOL) *onSD* directory:(NSString ∗∗) *directory*

List Directory

This command retrieves a directory listing of user accessible files from the reader

Parameters

directoryName	The directory name. If null, root directory is selected
recursive	Include sub-directories
onSD	True = use SD card, False = use Flash
directory	The returned directory information

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.86 - (RETURN_CODE) device_listenForNotifications: (BOOL) *enable*

Listen for Notifcations

Instructs SDK to listen for unsolicited data

Parameters

enable	True = Listen, False = Don't listen

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.87 - (RETURN_CODE) device_loadCertCA: (Byte) *type* CertData:(NSData ∗) *cert*

Load CA Certificate

Loads the CA certificate data onto the device

Parameters

type	The type of certificate -00h: Application CA -01h: TLS CA
cert	The certificate data bytes

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.88 - (RETURN_CODE) device_logClear

Log Clear

Instructs the device to delete all log data

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.89 - (RETURN_CODE) device_logEnable: (BOOL) *enable*

Enable Log

Instructs the device to enable/disable the log

Parameters

enable	True = enable log, False = disable log

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.90 - (RETURN_CODE) device_logRead: (NSData ∗∗) *response*

Log Read

Instructs the device to output all log data

Parameters

response	Full response received from NEO2 device, including log

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.91 - (RETURN_CODE) device_lowPowerMode: (BOOL) *stopMode* wakeOnTrans:(BOOL) *wakeOnTrans*

Enter Low Power Mode

Puts the terminal in sleep or stop mode, with the option to wake on swipe/tap

Parameters

stopMode	True = Stop Mode (POR required), False = Sleep Mode (resume from last instruction)
wakeOnTrans	True = Swipe/Tap will wake from low power, False = Will not wake from swipe/tap

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.92 - (RETURN_CODE) device_offYellowLED

Turn Off Yellow LED

Turn off the ViVOpay Vendi reader yellow LED. This LED is located below the three blue LEDs

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.93 - (RETURN_CODE) device_onYellowLED

Turn On Yellow LED

Turn on the ViVOpay Vendi reader yellow LED. This LED is located below the three blue LEDs

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.94 - (RETURN_CODE) device_pingDevice

Ping Device

Pings the reader. If it is connected, returns success, otherwise returns timeout

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.95 - (RETURN_CODE) device_pollForToken: (Byte) *seconds* card:(Byte **) *card* serialNumber:(NSData **) *serialNumber*

Poll for Token

Once Pass-Through Mode is started, ViVOpay will not poll for any cards until the “Poll for Token” command is received. This command tells ViVOpay to start polling for a Type A or Type B PICC until a PICC is detected or a timeout occurs.

This command automatically turns the RF Antenna on.

If a PICC is detected within the specified time limit, ViVOpay activates it and responds back to the terminal with card related data such as the Serial Number. If no PICC is detected within the specified time limit, ViVOpay stops polling and responds back indicating that no card was found. No card related data is returned in this case

Parameters

timeout	Timeout, in seconds to wait for card to be detected
card	Card Type: • 00h None (Card Not Detected or Could not Activate) • 01h ISO 14443 Type A (Supports ISO 14443-4 Protocol) • 02h ISO 14443 Type B (Supports ISO 14443-4 Protocol) • 03h Mifare Type A (Standard) • 04h Mifare Type A (Ultralight) • 05h ISO 14443 Type A (Does not support ISO 14443-4 Protocol) • 06h ISO 14443 Type B (Does not support ISO 14443-4 Protocol) • 07h ISO 14443 Type A and Mifare (NFC phone)

Parameters

serialNumber	Serial Number or the UID of the PICC
ident	Device ID to send command to. If not specified, current SDK default device will be used.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.96 - (RETURN_CODE) device_queryFile: (NSString ∗) *directory* filename:(NSString ∗) *filename* isSD:(BOOL) *isSD* exists:(BOOL ∗) *exists* timestamp:(NSDate ∗∗) *timestamp* fileSize:(int ∗) *fileSize*

Query File

Report if tthe file exists, and if so will report the file timestamp and the file size

Parameters

directory	The file directory to search in. If blank, will use the root directory
filename	Name of the file to retrieve
isSD	True = query SD card, False = query internal storage
exists	True = file exists, False = file does not exist
timestamp	If the file exists, reports the timestamp of the file
fileSize	If the file exists, reports the size of the file

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.97 - (RETURN_CODE) device_readFileFromSD: (NSString ∗) *directory* filename:(NSString ∗) *filename* fileData:(NSData ∗∗) *fileData*

Read File from SD card

Reads a file from the SD card

Parameters

directory	The file directory to read from. If empty, the root directory is used
filename	The name of the file to retrieve
fileData	The contents of the file if it exists

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.98 - (RETURN_CODE) device_rebootDevice

Reboot Device

- NEO2
- VP3300 -VP8800 -UniPayI_V

Executes a command to restart the device.

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.99 - (RETURN_CODE) device_resetNVM

Reset non-volatile memory

The Set Configuration Defaults and Keep Encryption Key command provides an external method for resetting parameters in non-volatile memory (NVM) to their default values. When the reader receives this command, it erases EEPROM (but retains encryption keys). After completing initialization, the reader reboots.

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.100 - (RETURN_CODE) device_retrieveTerminalData: (NSData **) *responseData*

Retrieve Terminal Data

Retrieves the Terminal Data . The Terminal Data will be in the response parameter responseData

Parameters

responseData	The response returned from the method as NSData

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD

- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.101 - (RETURN_CODE) device_rrcConnect

RRC Connect

The RRC Connect command allows a host to establish an RRC connection to a reader. A host must first establish an RRC connection to the reader before issuing other RRC IDG commands.

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.102 - (RETURN_CODE) device_rrcDisconnect

RRC Disonnect

The RRC Disonnect command allows a host to terminate the RRC connection to a reader.

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.103 - (RETURN_CODE) device_rrcDownloadApp: (NSString ∗) *appName* appData:(NSData ∗) *appData*

RRC Download Application

The RRC Download Application command allows the transfer of a compressed application file from a host to a reader, extracts it, and performs signature verification on its contents

The reader receives all the chunked data from the host and, once completed, the reader combines it into one file

Parameters

appName	The name of the application that appears on the Application Manager list
appData	The compressed application file to be sent in chunks

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.104 - (RETURN_CODE) device_rrcInstallApp: (NSString ∗) *appName*

RRC Install Application

The RRC Install Application command installs a downloaded application to a reader. Only installed applications can run on the reader.

Parameters

appName	The name of the application that will appear on the Application Manager list

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.105 - (RETURN_CODE) device_rrcRunApp: (NSString ∗) *appName*

RRC Run application

This command allows the reader to run an installed application

Parameters

appName	The name of the installed application to run

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.106 - (RETURN_CODE) device_rrcUninstallApp: (NSString ∗) *appName*

RRC Uninstall Application

The RRC Uninstall Application command uninstalls an application on a reader device. The application will remain in the file system of the reader but cannot be executed.

Parameters

appName	The name of the application that appears on the Application Manager list

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.107 - (RETURN_CODE) device_sendIDGCommand: (unsigned char) *command* subCommand:(unsigned char) *subCommand* data:(NSData ∗) *data* response:(NSData ∗∗) *response*

Send NEO IDG Command Send a NEO IDG ViVOtech 2.0 command

Parameters

command	One byte command as per NEO IDG Reference Guide
subCommand	One byte sub-command as per NEO IDG Reference Guide
data	Command data (if applicable)
response	Returns next Command response

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_UniMagIII::device_getResponseCodeString:()

15.8.2.108 - (RETURN_CODE) device_sendIDGCommandV3: (unsigned char) *command* subCommand:(unsigned char) *subCommand* data:(NSData ∗) *data* response:(NSData ∗∗) *response*

Send NEO IDG Command Send a NEO IDG ViVOtech 3.0 command

Parameters

command	One byte command as per NEO IDG Reference Guide
subCommand	One byte sub-command as per NEO IDG Reference Guide
data	Command data (if applicable)
response	Returns next Command response

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_UniMagIII::device_getResponseCodeString:()

15.8.2.109 - (void) device_setBLEFriendlyName: (NSString ∗) *friendlyName*

Set BLE Friendly Name

Parameters

friendlyName	Sets the default friendly name to be used when discovering any BLE devices

15.8.2.110 - (RETURN_CODE) device_setBurstMode: (int) *mode*

Send Burst Mode

Sets the burst mode forthe device.

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.111 - (RETURN_CODE) device_setMerchantRecord: (int) *index* enabled:(bool) *enabled* merchantID:(NSString ∗) *merchantID* merchantURL:(NSString ∗) *merchantURL*

Set Merchant Record Sets the merchant record for ApplePay VAS

Parameters

index	Merchant Record index, valid values 1-6
enabled	Merchant Enabled/Valid flag
merchantID	Merchant unique identifer registered with Apple. Example com.idtechproducts.applePay
merchantURL	Merchant URL, when applicable

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.112 - (RETURN_CODE) device_setPassThrough: (BOOL) *enablePassThrough*

Set Pass Through

Sets Pass-Through mode on NEO2

Parameters

enablePassThrough	TRUE = Pass through enabled

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.113 - (RETURN_CODE) device_setPollMode: (int) *mode*

Send Poll Mode

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

Parameters

mode	0 = Auto Poll, 1 = Poll On Demand

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.114 - (RETURN_CODE) device_setSpecialFunctionOrFeature: (NSData ∗) *feature* addRequirement:(NSData ∗) *addRequirement*

Set Special Function or Feature Configuration Command

The Set Special Function or Feature Configuration command sets a specific special configuration.

Parameters

feature	Function/Feature ID
addRequirement	Additional Requirement

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.115 - (RETURN_CODE) device_setTerminalData: (NSData ∗) *tags*

Set Terminal Data

Sets the Terminal Data .

Parameters

responseData	The data to set

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.116 - (RETURN_CODE) device_setTransArmorEncryption: (NSData ∗) *cert*

Set TransArmor Encryption

Sets the TransArmor encryption from the given certificate

Parameters

cert	Certificate in PEM format or DER format. PEM format must be string data (converted to binary) starting with "---". DER format is binary data.

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.117 - (RETURN_CODE) device_setTransArmorID: (NSString ∗) *TID*

Set TransArmor ID

Sets the TransArmor ID

Parameters

TID	TransArmor ID

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.118 - (RETURN_CODE) device_startTransaction: (double) *amount* type:(int) *type* timeout:(int) *timeout* tags:(NSData *) *tags*

Start a Transaction Request

Authorizes the CTLS transaction for an CTLS card

The tags will be returned in the callback routine.

Parameters

amount	Transaction amount value (tag value 9F02)
exponent	Number of characters after decimile point
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request.

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

NOTE ON INTERFACE SELECTION: For the NEO2, tag DFEF37 is used to determine which interfaces to use for the transaction:

- 01 = MSR Only
- 02 = CTLS Only
- 03 = MSR + CTLS
- 04 = EMV Only
- 05 = EMV + MSR
- 06 = EMV + CTLS
- 07 = EMV + MSR + CLTS. This API method will automatically send DFEF37 with a value of 07 if this tag is not provided.

  USE tag DFEF3C for fallback support and timeout waiting for insertion byte 1: = fallback support 01 = YES, 00 = NO byte 2-3 = timeout in BCD. Example 60 seconds is 0060

15.8.2.119 - (RETURN_CODE) emv_authenticateTransaction: (NSData *) *tags*

Authenticate Transaction

Authenticates a transaction after startTransaction successfully executes.

By default, auto authorize is ENABLED. If auto authorize is DISABLED, this function must be called after a result EMV_RESULT_CODE_START_TRANSACTION_SUCCESS returned to emvTransactionData delegate protocol is received after a startTransaction call. If auto authorize is ENABLED (default), this method will automatically be executed after receiving the result EMV_RESULT_CODE_START_TRANSACTION_SUCCESS after startTransaction. The auto authorize can be enabled/disabled with IDT_DEVICE::disableAutoAuthenticateTransaction:()

The purpose of this step is to allow the merchant the chance to evaluate the data captured from the matching Application (if found) before the kernel authenticates the transaction data. This would allow, for instance, the merchant to be told what card is being used, and if it is a specific type (like a store card), perform an action like reducing the amount before proceeding (as a promotion in using that card).

Parameters

tags	Any tags to modify original tags submitted with startTransaction. Passed as NSData. Example, tag 9F0C with amount 0x000000000100 would be 0x9F0C06000000000100 Tag DFEE1A can be used to specify tags to be returned in response, in addition to the default tags. Example DFEE1A049F029F03 will return tags 9F02 and 9F03 with the response

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.120 - (RETURN_CODE) emv_callbackResponseKSN: (NSData *) *KSN*

Callback Response Get ETC DUKPT key KSN

Provides a status code to device request of DUKPT IK loaded status, from callback type GET KSN

Parameters

KSN	The KSN data

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.121 - (RETURN_CODE) emv_callbackResponseLCD: (int) *mode* selection:(unsigned char) *selection*

Callback Response LCD Display

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

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.122 - (RETURN_CODE) emv_callbackResponseMSR: (NSData *) *MSR*

Callback Response MSR Entry

Provides MSR information to kernel after a callback was received with type EMV_CALLBACK_MSR

Parameters

MSR	Swiped track data

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.123 - (RETURN_CODE) emv_callbackResponsePIN: (EMV_PIN_MODE_Types) *mode* KSN:(NSData *) *KSN* PIN:(NSData *) *PIN*

Callback Response PIN Request

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

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.124 - (RETURN_CODE) emv_callbackResponsePIN_ETC: (EMV_PIN_MODE_Types) *type* ksn:(NSData ∗) *KSN* pin:(NSData ∗) *PIN*

Callback Response PIN Entry for ETC

Provides (or cancels) PIN entry information to kernel after a callback was received with callback type PINPAD ETC

Parameters

type	If cancel button is pressed during PIN entry, then this value is EMV_PIN_MODE_CANCEL • If PIN bypass is pressed during PIN entry, then this value is EMV_PIN_MODE_BYPASS. Otherwise the value can be EMV_PIN_MODE_ONLINE_DUKPT, EMV_PIN_MODE_ONLINE_MKSK, or EMV_PIN_MODE_OFFLINE
KSN	If enciphered PIN, this is either PINK DUKPT Key or PIN Session Key or PIN Pairing DUKPT
PIN	If enciphered PIN, this is an ecrypted PIN block. If device does not implement pairing fuctions, this plaintext PIN

15.8.2.125 - (RETURN_CODE) emv_cancelTransaction

Cancels Transaction request (all interfaces, CTLS/MSR/INSERT).

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.126 - (RETURN_CODE) emv_completeOnlineEMVTransaction: (BOOL) *isSuccess* hostResponseTags:(NSData ∗) *tags*

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

isSuccess	Determines if connection to host was successful: • TRUE: Online processing with the host (issuer) was completed • FALSE: Online processing could not be completed due to connection error with the host (issuer). No further data (tags) required.
tags	Host response tag (see below)

Host response tag:

Tag	Length	Description
8A	2	Data element Authorization Response Code. Mandatory
91	8-16	Issuer Authentication Data. Optional
71	0-256	Issuer Scripts. Optional
72	0-256	Issuer Scripts. Optional
DFEE1A	0-256	Tag list of additional tags to return

Tag DFEE1A will force additional tags to be returned. Example DFEE1A049F029F03 will return tags 9F02 and 9F03 with the response

Returns

RETURN_CODE:

- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0xFE00: Authorization Accepted - RETURN_CODE_EMV_AUTHORIZATION_ACCEPTED
- 0xFE01: Online Failure - RETURN_CODE_EMV_AUTHORIZATION_UNABLE_TO_GO_ONLINE
- 0xFE02: Technical Issue - RETURN_CODE_EMV_AUTHORIZATION_TECHNICAL_ISSUE
- 0xFE03: Declined - RETURN_CODE_EMV_AUTHORIZATION_DECLINED
- 0xFE04: Issuer Referral - RETURN_CODE_EMV_AUTHORIZATION_ISSUER_REFERRAL

15.8.2.127 - (void) emv_disableAutoAuthenticateTransaction: (BOOL) *disable*

Disable Auto Authenticate Transaction

If auto authenticate is DISABLED, authenticateTransaction must be called after a successful startEMV execution.

Parameters

disable	FALSE = auto authenticate ENABLED, TRUE = auto authenticate DISABLED

15.8.2.128 - (RETURN_CODE) emv_exchangeCerts: (NSData **) *cert* nonce:(NSData **) *nonce* signature:(NSData **) *signature*

Exchange Certificates, Nonces, and Keys

Use this command to send the ETC certificate, nonce, and signature The returned data is the NEO2 certificate, nonce, and signature

Parameters

cert	Send ETC certificate for signature verification, Receives NEO2 certificate for signature verification
nonce	Send ETC random nonce, Receives NEO2 random nonce
signature	Send ETC signature, Receives NEO2 signature. Signature of (CertETC_SV || NONCE_ETC) with PKCS1-v1_5 padding

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.129 - (RETURN_CODE) emv_generateDUKPT: (NSData *) *cert* signature:(NSData *) *signature* key:(NSData **) *key*

Generate DUKPT IK using KEK

Use this command to send the encrypted KEK and signature generated by the ETC. NEO2 returns the DUKPT IK in TR-31 format encrypted with the KEK and signature

Parameters

cert	ETC certificate for signature verification
signature	Signature of (KEK || NONCE_ETC) with PKCS1-v1_5 padding
key	ASN.1 structure of DUKPT IK used between NEO2 and ETC

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.130 - (RETURN_CODE) emv_getBatteryPercentage: (NSString **) *response*

Battery Percentage

Polls the device for the current battery percentage

Parameters

response	Returns the battery percentage represented as a whole number 5 - 100%

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.8.2.131 - (RETURN_CODE) emv_getBatteryVoltage: (NSString **) *response*

Battery Voltage

Polls the device for the current battery voltage

Parameters

response	Returns battery voltage string representing millivolts

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.132 - (RETURN_CODE) emv_getEMVConfigurationCheckValue: (NSString **) *response*

Get EMV Configuration Check Value

Polls device for the EMV Configuration Check Value

Parameters

response	Response returned of the Check Value of the Configuration

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.133 - (RETURN_CODE) emv_getEMVKernelCheckValue: (NSString **) *response*

Get Kernel Check Value

Polls the device for the Kernel Check Value

Parameters

response	Response returned of the Check Value of the Kernel

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.134 - (RETURN_CODE) emv_getEMVKernelVersion: (NSString **) *response*

Get EMV Kernel Version

Polls the device for the EMV Kernel Version

Parameters

response	The kernel version response in a string

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.135 - (RETURN_CODE) emv_getEMVKernelVersionExt: (NSString **) *response*

Get Extended EMV Kernel Version

Polls the device for the extended EMV kernel version

Parameters

response	The extended kernel verion response in a string

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.136 - (RETURN_CODE) emv_getEMVL2Version: (NSString **) *response*

Polls device for EMV L2 Version

Parameters

response	Response returned of Level 2 Version

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.137 - (RETURN_CODE) emv_getTerminalMajorConfiguration: (NSUInteger **) *configuration*

Gets the terminal major configuration in ICS .

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.138 - (RETURN_CODE) emv_removeAllApplicationData

Remove All Application Data

Removes all the application data for the EMV kernel

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.139 - (RETURN_CODE) emv_removeApplicationData: (NSString ∗) *AID*

Remove Application Data by AID

This will REMOVE the an AID configuration file and all the tlv data associated with that AID.

Parameters

AID	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters If null or empty string is passed, it will remove ALL application data

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to BTPay::device_getResponseCodeString:()

15.8.2.140 - (RETURN_CODE) emv_removeCAPK: (NSString ∗) *rid* index:(NSString ∗) *index*

Remove Certificate Authority Public Key

Removes the CAPK as specified by the RID/Index passed as a parameter in the CAKey structure

Parameters

rid	RID of the key to remove
index	Index of the key to remove

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER

- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.141 - (RETURN_CODE) emv_removeCRLList

Remove Certificate Revocation List

Removes all CRLEntry entries

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.142 - (RETURN_CODE) emv_removeTerminalData

Remove Terminal Data

This will remove ALL configurable TLV data associated with the terminals Kernel configuration.

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.143 - (RETURN_CODE) emv_retrieveAIDList: (NSArray ∗∗) *response*

Retrieve AID list

Returns all the AID names on the terminal. Populates response parameter with an Array of NSString∗ with AID names. Each AID name represent a unique configuration file to be loaded/used when a matching application is found on a card during a transaction.

Parameters

response	Returns a NSArray of NSString of AID Names

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to BTPay::device_getResponseCodeString:()

15.8.2.144 - (RETURN_CODE) emv_retrieveApplicationData: (NSString ∗) *AID* response:(NSDictionary ∗∗) *responseAID*

Retrieve Application Data by AID

Retrieves the configuration information for a provided AID name, if that AID file exists on the terminal.

The TLV data in that AID is returned as a NSDictionary, with the Key being the tag name as a NSString representation of the tag hex value (example "9F06"), and the Object being the Value as NSData (example 0xa0000000031010).

The data returned will be from the range of allowable kernel EMV tags. Please see "EMV Tag Reference" at the end of this document for the listing.

Parameters

AID	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters
responseAID	The response returned from the method as a dictionary with Key/Object to match TagValues as follows:

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.145 - (RETURN_CODE) emv_retrieveCAPK: (NSString ∗) *rid* index:(NSString ∗) *index* response:(CAKey ∗∗) *response*

Retrieve Certificate Authority Public Key

Retrieves the CAPK as specified by the RID/Index passed as a parameter in the CAKey structure. The CAPK will be in the response parameter

Parameters

rid	The RID of the key to retrieve
index	The Index of the key to retrieve
response	Response returned as a CAKey

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.146 - (RETURN_CODE) emv_retrieveCAPKFile: (NSString ∗) *rid* index:(NSString ∗) *index* response:(NSData ∗∗) *response*

Retrieve Certificate Authority Public Key

- BTPay 200

Retrieves the CAPK as specified by the RID/Index passed as a parameter in the CAKey structure. The CAPK will be in the response parameter

Parameters

rid	The RID of the key to retrieve
index	The Index of the key to retrieve
response	Response returned as a NSData object with the following data: • 5 bytes RID • 1 byte Index • 1 byte Hash Algorithm • 1 byte Encryption Algorithm • 20 bytes HashValue • 4 bytes Public Key Exponent • 2 bytes Modulus Length • Variable bytes Modulus>

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.147 - (RETURN_CODE) emv_retrieveCAPKList: (NSArray **) *response*

Retrieve the Certificate Authority Public Key list

Returns all the CAPK RID and Index. Populates response parameter with an array of NSString items, 12 characters each, characters 1-10 RID, characters 11-12 index.

Parameters

response	Response returned contains an NSArray of NSString items, 12 characters each, characters 1-10 RID, characters 11-12 index. Example "a00000000357" = RID a00000003, Index 57

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.148 - (RETURN_CODE) emv_retrieveCRLList: (NSMutableArray **) *response*

Retrieve the Certificate Revocation List

Returns all the RID in the CRL.

Parameters

response	Response returned as an NSArray of NSData objects 9-bytes each: • 5-bytes RID, 1-byte Index, 3-byte Serial Number

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.149 - (RETURN_CODE) emv_retrieveTerminalData: (NSDictionary ∗∗) *responseData*

Retrieve Terminal Data

Retrieves the tag values associated with the terminal configuration file. This will be a combination of uneditable major configuration tags for the kernel configuration (example 9F33, Terminal Capabilities), and editable tags set with IDT_Device::emv_setTerminalData:() (example DF13, Terminal Action Code - Default)

The TLV data returned as a NSDictionary, with the Key being the tag name as a NSString representation of the tag hex value (example "DF13"), and the Object being the Value as NSData (example 0x00058003FF).

The data returned will be from the range of allowable kernel EMV tags. Please see "EMV Tag Reference" at the end of this document for the listing.

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.150 - (RETURN_CODE) emv_retrieveTransactionResult: (NSData ∗) *tags* retrievedTags:(NSDictionary ∗∗) *retrievedTags*

Retrieve Transaction Results

Retrieves the requested tag values (if they exist) from the last transaction.

The TLV data returned as a NSDictionary, with the Key being the tag name as a NSString representation of the tag hex value (example "5A"), and the Object being the Value as NSData (example 0x41359276429372938).

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.151 - (RETURN_CODE) emv_setApplicationData: (NSString *) *aidName* configData:(NSDictionary *) *data*

Set Application Data by AID

Sets the configuration information for a provided AID name, with TLV data that populates a NSDictionary.

The TLV data for the AID is sent as a NSDictionary, with the Key being the tag name as a NSString representation of the tag hex value (example "9F06"), and the Object being the Value as NSData (example 0xa0000000031010).

The data for the AID configuration will will be from the range of allowable kernel EMV tags. Please see "EMV Tag Reference" at the end of this document for the listing.

NOTES: There is no minimum defined set of AID TLV data that must be provided, other than 9F06 for the AID name.

If this AID is selected and matched during an EMV transaction, any data in this AID will either OVERRIDE the same data in the terminal configuration file, or PROVIDE the data if it is non-existant in the terminal configuration file.

AID configuration information is provided during L3 certification. Dummy/stub AID data can be used pre-certification to test EMV transaction as long as at least tag 9F06 is defined that makes up the AID configuration locator.

There are convenience utilities to turn a TLV NSData object into a NSDictionary, and a NSDictionary into a NSData object in IDTUtility:

```
+(NSDictionary*) TLVtoDICT:(NSData*)param;
+(NSData*) DICTotTLV:(NSDictionary*)tags;
```

Also utilities to turn a HEX/ASCII string to NSDATA and back again

```
+ (NSData *)hexToData:(NSString*)str ;
+(NSString*) dataToHexString:(NSData*)data;
```

Parameters

aidName	aidName AID name. Example "a0000000031010"
data	NSDictionary with Tags/Values for the AID configuration file

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT

- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.152 - (RETURN_CODE) emv_setCAPK: (CAKey) *key*

Set Certificate Authority Public Key

Sets the CAPK as specified by the CAKey structure

Parameters

key	CAKey containing the RID, Index, and key data to set

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.153 - (RETURN_CODE) emv_setCAPKFile: (NSData ∗) *file*

Set Certificate Authority Public Key

Sets the CAPK as specified by the CAKey raw format

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.154 - (RETURN_CODE) emv_setCRLEntries: (NSData ∗) *data*

Set Certificate Revocation List

Sets the CRL list

Parameters

data	CRLEntries as a repeating occurance of CRL: CRL1 CRL2 . . . CRLn. CRL format is • 5Bytes RID • 1Byte CA public key Index • 3Bytes Certificate Serial Number

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_UniPayII::device_getResponseCodeString:()

15.8.2.155 - (RETURN_CODE) emv_setTerminalData: (NSDictionary ∗) *data*

Set Terminal Data

Sets the terminal configuration information, with TLV data that populates a NSDictionary.

The TLV data for the terminal configuration is sent as a NSDictionary, with the Key being the tag name as a NS↩ String representation of the tag hex value (example "DF13"), and the Object being the Value as NSData (example 0x00080039FF).

The data for the terminal configuration will will be from the range of allowable kernel EMV tags. Please see "EMV Tag Reference" at the end of this document for the listing.

NOTES: There is an uneditable set of tags that make up the current kernel configuration major parameters. Any attempt to set those will return an error.

If an AID is selected and matched during an EMV transaction, any data in that AID will either OVERRIDE the same data in the terminal configuration file, or PROVIDE the data if it is non-existant in the terminal configuration file.

There are convenience utilities to turn a TLV NSData object into a NSDictionary, and a NSDictionary into a NSData object in IDTUtility:

```
+(NSDictionary*) TLVtoDICT:(NSData*)param;
+(NSData*) DICTotTLV:(NSDictionary*)tags;
```

Also utilities to turn a HEX/ASCII string to NSDATA and back again

```
+ (NSData *)hexToData:(NSString*)str ;
+(NSString*) dataToHexString:(NSData*)data;
```

Parameters

data	NSDictionary with Tags/Values for the Terminal configuration file

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to device_getResponseCodeString:

15.8.2.156 - (RETURN_CODE) emv_setTerminalMajorConfiguration: (int) *configuration*

Sets the terminal major configuration in ICS .

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.157 - (RETURN_CODE) emv_startTransaction: (double) *amount* amtOther:(double) *amtOther* type:(int) *type* timeout:(int) *timeout* tags:(NSData ∗) *tags* forceOnline:(BOOL) *forceOnline* fallback:(BOOL) *fallback*

Start EMV Transaction Request

Authorizes the EMV transaction for an ICC card

The tags will be returned in the emvTransactionData delegate protocol.

By default, auto authorize is ENABLED. If auto authorize is DISABLED, this function will complete with a return of EMV_RESULT_CODE_START_TRANSACTION_SUCCESS to emvTransactionData delegate protocol, and then IDT_NEO2::emv_authenticateTransaction() must be executed. If auto authorize is ENABLED (default), IDT_N↩EO2::emv_authenticateTransaction() will automatically be executed after receiving the result EMV_RESULT_C↩ODE_START_TRANSACTION_SUCCESS. The auto authorize can be enabled/disabled with emv_disableAuto↩AuthenticateTransaction:

Parameters

amount	Transaction amount value (tag value 9F02)
amtOther	Other amount value, if any (tag value 9F03)
type	Transaction type (tag value 9C).
timeout	Timeout value in seconds.
tags	Any other tags to be included in the request. Passed as NSData. Example, tag 9F0C with amount 0x000000000100 would be 0x9F0C06000000000100 If tags 9F02 (amount),9F03 (other amount), or 9C (transaction type) are included, they will take priority over these values supplied as individual parameters to this method. Tag DFEE1A can be used to specify tags to be returned in response, in addition to the default tags. Example DFEE1A049F029F03 will return tags 9F02 and 9F03 with the response
forceOnline	TRUE = do not allow offline approval, FALSE = allow ICC to approve offline if terminal capable
autoAuthenticate	Will automatically execute Authenticate Transacation after start transaction returns successful
fallback	Indicate if it supports fallback to MSR

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

NOTE ON INTERFACE SELECTION: For the NEO2, tag DFEF37 is used to determine which interfaces to use for the transaction:

- 01 = MSR Only
- 02 = CTLS Only
- 03 = MSR + CTLS
- 04 = EMV Only
- 05 = EMV + MSR
- 06 = EMV + CTLS
- 07 = EMV + MSR + CLTS. This API method will automatically send DFEF37 with a value of 04 if this tag is not provided.

15.8.2.158 - (RETURN_CODE) emv_verifyDUKPTLoaded: (NSData *) *KCV*

Verify DUKPT IK Loaded on ETC

Use this command to verify the DUKPT IK isloaded into the ETC. NEO2 is activated and it can request PIN from ETC after this command

Parameters

KCV	ASN.1 structure of KCV

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with IDT_Device::device_getResponseCodeString:()

15.8.2.159 - (RETURN_CODE) felica_authentication: (NSData *) *key*

FeliCa Authentication

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

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

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

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.160 - (RETURN_CODE) felica_read: (NSData ∗) *serviceCode* numBlocks:(int) *numBlocks* blockList:(NSData ∗) *blockList* blocks:(NSData ∗∗) *blocks*

FeliCa Read

Reads up to 4 blocks.

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

serviceCode	Service Code List. Each service code in Service Code List = 2 bytes of data
numBlocks	Number of blocks
blockList	Blocks to read. Maximum 4 block requests
blocks	Blocks read. Each block 16 bytes.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.161 - (RETURN_CODE) felica_readWithMac: (int) *numBlocks* blockList:(NSData ∗) *blockList* blocks:(NSData ∗∗) *blocks*

FeliCa Read with MAC Generation

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

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

numBlocks	Number of blocks
blockList	Block to read. Each block in blockList Maximum 3 block requests
blocks	Blocks read. Each block 16 bytes.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.162 - (RETURN_CODE) felica_requestService: (NSData ∗) *nodeCode* response:(NSData ∗∗) *response*

FeliCa Request Service

Perform functions a Felica Request Service

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

nodeCode	Node Code
response	Response as explained in FeliCA Lite-S User's Manual

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.163 - (RETURN_CODE) felica_SendCommand: (NSData ∗) *command* response:(NSData ∗∗) *response*

FeliCa Send Command

Send a Felica Command

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

command	Command data from settlement center to be sent to felica card
response	Response data from felica card

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.164 - (RETURN_CODE) felica_write: (NSData ∗) *serviceCode* blockCount:(int) *blockCount* blockList:(NSData ∗) *blockList* data:(NSData ∗) *data* statusFlag:(NSData ∗∗) *statusFlag*

FeliCa Write

Writes a block

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

serviceCode	Service Code list. Each service code must be be 2 bytes
blockCount	Block Count
blockList	Block list.
data	Block to write. Must be 16 bytes.
statusFlag	Status flag response as explained in FeliCA Lite-S User's Manual, Section 4.5

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.165 - (RETURN_CODE) felica_writeWithMac: (int) *blockNumber* data:(NSData ∗) *data*

FeliCa Write with MAC Generation

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

NOTE: The reader must be in Pass Through Mode for FeliCa commands to work.

Parameters

blockNumber	Number of block
data	Block to write. Must be 16 bytes.

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.166 - (RETURN_CODE) icc_exchangeAPDU: (NSData *) *dataAPDU* response:(APDUResponse **) *response*

Exchange APDU (unencrypted)

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

Parameters

dataAPDU	APDU data packet
response	Unencrypted/encrypted parsed APDU response

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.167 - (RETURN_CODE) icc_getICCReaderStatus: (ICCReaderStatus **) *readerStatus*

Get Reader Status

Returns the reader status

Parameters

readerStatus	Pointer that will return with the ICCReaderStatus results.

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

```
ICCReaderStatus* readerStatus;
RETURN_CODE rt = [[IDT_Device sharedController] getICCReaderStatus:&readerStatus];
if(RETURN_CODE_DO_SUCCESS != rt){
NSLog(@"Fail");
}
else{
NSString *sta;
if(readerStatus->iccPower)
sta =@"[ICC Powered]";
else
sta = @"[ICC Power not Ready]";
if(readerStatus->cardSeated)
sta =[NSString stringWithFormat:@"%@,[Card Seated]", sta];
else
sta =[NSString stringWithFormat:@"%@,[Card not Seated]", sta];
```

15.8.2.168 - (RETURN_CODE) icc_getKeyFormatForICCDUKPT: (NSData **) *format*

Get Key format for ICC DUKPT

Specifies how data is being encrypted with Data Key or PIN key

Parameters

format	Response return from method: -'TDES' : Encrypted card data with TDES if DUKPT Key had been loaded -'AES' : Encrypted card data with AES if DUKPT Key had been loaded -NONE' : No Encryption

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.169 - (RETURN_CODE) icc_powerOffICC: (NSString **) *error*

Power Off ICC

Powers down the ICC

Parameters

error	Returns the error, if any

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE.

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

15.8.2.170 - (RETURN_CODE) icc_powerOnICC: (NSData **) *response*

Power On ICC

Power up the currently selected microprocessor card in the ICC reader

Parameters

response	Response returned. If Success, ATR String. If Failure, ASCII encoded data of error string

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE.

15.8.2.171 - (RETURN_CODE) icc_setKeyFormatForICCDUKPT: (NSData *) *encryption*

Set Key format for ICC DUKPT

Specifies how data will be encrypted with Data Key or PIN key

Parameters

encryption	The type of encryption to be used -0x00 : Encrypted card data with TDES if DUKPT Key had been loaded -0x01 : Encrypted card data with AES if DUKPT Key had been loaded -Other Data : No Encryption

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.172 - (bool) isConnected

Check if device is connected

15.8.2.173 - (RETURN_CODE) msr_cancelMSRSwipe

Disable MSR Swipe

Cancels Swipe.

Cancels Transaction request (all interfaces, CTLS/MSR/INSERT).

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.174 - (RETURN_CODE) msr_getConfiguration: (NSData **) *config*

Get MSR Configuration

Gets the MSR configuration data

Parameters

config	Configuration data retrieved

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.175 - (RETURN_CODE) msr_getMSRTrack: (int *) *val*

Get MSR Track

Returns the track value of the MSR

Parameters

val	The value of the current MSR tracks setting • 0 : Any track • 1 : Track 1 • 2 : Track 2 • 3 : Track 1, Track 2 • 4 : Track 3 • 5 : Track 1, Track 3 • 6 : Track 2, Track 3 • 7 : Track 1, Track 2 , Track 3

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.176 - (RETURN_CODE) msr_retrieveWhiteList: (NSData **) *value*

Retrieve MSR White List

Retrieves the whitelist

Parameters

value	The whitelist data which is in ASN.1 block format

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.177 - (RETURN_CODE) msr_setConfiguration: (NSData *) *config*

Set MSR Configuration

Sets MSR configuration data

Parameters

config	Configuration data to send

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.178 - (RETURN_CODE) msr_setMSRTrack: (int) *val*

Set MSR Track

Sets the track value of the MSR

Parameters

val	The value of the current MSR tracks setting • 0 : Any track • 1 : Track 1 • 2 : Track 2 • 3 : Track 1, Track 2 • 4 : Track 3 • 5 : Track 1, Track 3 • 6 : Track 2, Track 3 • 7 : Track 1, Track 2 , Track 3

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.179 - (RETURN_CODE) msr_startMSRSwipe

Enable MSR Swipe

Enables CLTS and MSR, waiting for swipe or tap to occur. Returns IDTEMVData to deviceDelegate::emv↩ TransactionData:()

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with device_getResponseCodeString:

15.8.2.180 - (RETURN_CODE) pin_cancelPin

Cancel PIN Command

This command can cancel IDT_Device:getEncryptedPIN:keyType:line1:line2:line3:() and IDT_Device::getNumeric↩ :minLength:maxLength:messageID:language:() and IDT_Device::getAmount:maxLength:messageID:language:() and IDT_Device::getCardAccount:max:line1:line2:() and IDT_Device::pin_getFunctionKey() and IDT_Device::get↩ EncryptedData:minLength:maxLength:messageID:language:()

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.181 - (RETURN_CODE) pin_captureAmountInput: (int) *minPIN* maxPIN:(int) *maxPIN* message:(NSString ∗) *message* signature:(NSData ∗) *signature*

Capture Amount Input

Parameters

minPIN	Minimum PIN Length
maxPIN	Maximum PIN Length
message	LCD Message
signature	Display message signed by Numeric Private Key using RSAPSS algorithm: 1. Calculate 32 bytes Hash for "<Display Flag><Key max="" length>=""><Key Min Length><Plaintext display="" message>=""> 2. Using RSAPSS algorithm calculate the Hash to be 256 bytes Raw Data 3. Using Numeric Private Key to sign the Raw Data to be 256 bytes signature

Results returned to pinpadData delegate

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.182 - (RETURN_CODE) pin_captureFunctionKey

Capture Function Key

Captures a function key entry on the pinpad

Results returned to pinpadData delegate

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.183 - (RETURN_CODE) pin_captureNumericInput: (bool) *mask* minPIN:(int) *minPIN* maxPIN:(int) *maxPIN* message:(NSString ∗) *message* signature:(NSData ∗) *signature*

Capture Numeric Input

Parameters

mask	True = mask input with "∗", False = no masking of input
minPIN	Minimum PIN Length
maxPIN	Maximum PIN Length
message	LCD Message
signature	Display message signed by Numeric Private Key using RSAPSS algorithm: 1. Calculate 32 bytes Hash for "<Display Flag><Key max="" length>=""><< Key Min Length><Plaintext display="" message>=""><" 2. Using RSAPSS algorithm calculate the Hash to be 256 bytes Raw Data 3. Using Numeric Private Key to sign the Raw Data to be 256 bytes signature

Results returned to pinpadData delegate

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.184 - (RETURN_CODE) pin_capturePin: (int) *type* PAN:(NSString ∗) *PAN* minPIN:(int) *minPIN* maxPIN:(int) *maxPIN* message:(NSString ∗) *message*

Capture PIN

Parameters

type	PAN and Key Type • 00h = MKSK to encrypt PIN, Internal PAN (from MSR) • 01h = DUKPT to encrypt PIN, Internal PAN (from MSR) • 10h = MKSK to encrypt PIN, External Plaintext PAN • 11h = DUKPT to encrypt PIN, External Plaintext PAN • 20h = MKSK to encrypt PIN, External Ciphertext PAN • 21h = DUKPT to encrypt PIN, External Ciphertext PAN

Parameters

PAN	Personal Account Number (if internal, value is 0)
minPIN	Minimum PIN Length
maxPIN	Maximum PIN Length
message	LCD Message

Results returned to pinpadData delegate

Returns

RETURN_CODE: Values can be parsed with device_getResponseCodeString:

15.8.2.185 - (void) scanForBLEDeviceNames: (NSTimeInterval) *scanTime* serviceUUIDs:(nullable NSArray< CBUUID * > *) *serviceUUIDs* options:(nullable NSDictionary< NSString *, id > *) *options*

Begins searching for Bluetooth Low Energy devices in range and return NSArray with found device names

- VP3300, NEO2

  This will initiate a bluetooth searech, and return an UIAlertView to the delegate bluetoothPickerAlert

Parameters

scanTime	Amount of time to search for devices before returning alert view.
serviceUUIDs	A null terminated array of CBUUID to filter for specific device type. Sending nil will report all ble devices in range
options	scan options

Note: Please refer to CBCentralManager scanForPeripheralsWithServices for information about serviceUU↩ IDs and options parameters

15.8.2.186 - (void) scanForBLEDevices: (NSTimeInterval) *scanTime* serviceUUIDs:(nullable NSArray< CBUUID * > *) *serviceUUIDs* options:(nullable NSDictionary< NSString *, id > *) *options*

Begins searching for Bluetooth Low Energy devices in range and return UIAlertView with choices

- VP3300, NEO2

  This will initiate a bluetooth searech, and return an UIAlertView to the delegate bluetoothPickerAlert

Parameters

scanTime	Amount of time to search for devices before returning alert view.
serviceUUIDs	A null terminated array of CBUUID to filter for specific device type. Sending nil will report all ble devices in range
options	scan options

Note: Please refer to CBCentralManager scanForPeripheralsWithServices for information about serviceUU↩ IDs and options parameters

15.8.2.187 + (NSString*) SDK_version

SDK Version

Returns the current version of IDTech.framework

Returns

Framework version

15.8.2.188 - (void) setServiceScanFilter: (NSArray< CBUUID * > *) *filter*

Set BLE Service Scan Filter.

When searching for BLE devices, this will limit the service search to the provided service ID's

Example data format: NSArray<CBUUID *> *filter = [[NSArray alloc] initWithObjects:[CBUUID UUIDWithString:"1820"], nil];

Parameters

filter	The array of services to filter for

15.8.2.189 - (void) setServiceUUID: (nullable NSArray< CBUUID * > *) *serviceUUIDs*

Set BLE to filter for specified service UUID

Parameters

setServiceUUID	Sets the UUID filter to be used when discovering BLE devices

15.8.2.190 + (IDT_NEO2*) sharedController

Singleton Instance

Establishes an singleton instance of IDT_NEO2 class.

Returns

Instance of IDT_NEO2

15.8.2.191 - (RETURN_CODE) updateFirmwareNeo2: (FIRMWARE_TYPE) *type* data:(NSData *) *firmwareData*

Update NEO 2 Firmware

Reserved for system use

15.8.3 Property Documentation

15.8.3.1 - (id< IDT_NEO2_Delegate >) delegate [read], [write], [atomic], [strong]

- Reference to IDT_NEO2_Delegate.

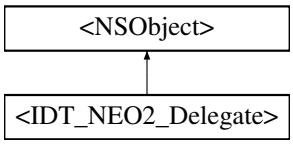
The documentation for this class was generated from the following file:

- Source_iOS/IDT_NEO2.h

15.9 <IDT_NEO2_Delegate> Protocol Reference

`#import <IDT_NEO2.h>`

Inheritance diagram for <IDT_NEO2_Delegate>:



Instance Methods

- (void) - deviceConnected

  Fires when device connects. If a connection is established before the delegate is established (no delegate to send initial connection notification to), this method will fire upon establishing the delegate.
- (void) - deviceDisconnected

  Fires when device disconnects.
- (void) - dataInOutMonitor:incoming:
- (void) - swipeMSRData:
- (void) - deviceMessage:
- (void) - updateStatus:currentBlock:totalBlocks:error:
- (RETURN_CODE) - sendPKUpdate:
- (RETURN_CODE) - sendPKUpdateBLE:
- (void) - pinRequest:key:PAN:startTO:intervalTO:language:
- (void) - bluetoothPickerAlert:
- (void) - bluetoothDeviceNames:
- (RETURN_CODE) - activateTransaction:timeout:
- (void) - pinpadData:keySN:event:
- (void) - lcdDisplay:lines:
- (void) - emvTransactionData:errorCode:

15.9.1 Detailed Description

Protocol methods established for IDT_NEO2 class

15.9.2 Method Documentation

15.9.2.1 - (RETURN_CODE) activateTransaction: (NSMutableDictionary< NSString *, NSString * > *_Nullable) *tags* timeout:(int) *timeout* [optional]

Activate Transaction

- VP3300 Initiates a CTLS transaction

Use this command when the ctls reader is in “Poll on Demand” mode to begin an EMV or contactless MagStripe Card transaction. When the reader is in “Poll on Demand” mode, the RF is turned on only after receiving an Activate Transaction command. When a valid Activate Transaction command is sent to the ctls reader, it starts polling for cards.

If the ctls reader does not find a supported card (an AID that matches one of the configured AIDs in the reader) for the specified time duration, it times out and ends the transaction. If the ctls reader finds a card within the specified time interval, it attempts to carry out the transaction. The transaction flow between the reader and the card depends on the type of card detected.

If the transaction is successful, the reader returns the data in CTLSResponse. If the transaction is not successful, yet it proceeded into the transaction state machine, the reader returns a Failed Transaction Record in the response data. The presence and format of the Clearing Record, Track Data and Failed Transaction record depends on the type of card that was detected.

Note: While an Activate command is in progress, only a Cancel may be sent. Do not send other commands until Activate Transaction has completed, because the reader will interpret these as a Cancel Transaction command.

Parameters

tags	Activate TLV tags
timeout	Timeout value in seconds.

Activate TVL Tag	Description	Format	Length
9A	Transaction Date	n6 (YYMMDD)	3
9C	Transction Type	n2	2
5F2A	Transaction Currency Code	n2	2
5F36	Transaction Currency Exponent	n1	1
9F02	Amount, Authorized	n12	6
9F03	Amount Other	n12	6
9F1A	Terminal Country Code	n3	2
9F21	Transaction Time	n6 (HHMMSS}	3
9F5A	Terminal Transaction Type	b	1

Transaction Types: 0x00 = Purchase Goods/Services, 0x20 = Refund Terminal Transaction Type (Interac) 0x00 = Purchase, 0x01 = Refund

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0xFF00: Accept the online transaction RETURN_CODE_EMV_APPROVED
- 0xFF01: Decline the online transaction RETURN_CODE_EMV_DECLINED
- 0xFF02: Request to go online RETURN_CODE_EMV_GO_ONLINE
- 0xFF03: Transaction is terminated RETURN_CODE_EMV_FAILED
- 0xFF05: ICC format error or ICC missing data error RETURN_CODE_EMV_SYSTEM_ERROR
- 0xFF07: ICC didn't accept transaction RETURN_CODE_EMV_NOT_ACCEPTED
- 0xFF0A: Application may fallback to magstripe technology RETURN_CODE_EMV_FALLBACK
- 0xFF0C: Transaction was cancelled RETURN_CODE_EMV_CANCEL

- 0xFF0D: Timeout RETURN_CODE_EMV_TIMEOUT
- 0xFF0F: Other EMV Error RETURN_CODE_EMV_OTHER_ERROR
- 0xFF10: Accept the offline transaction RETURN_CODE_EMV_OFFLINE_APPROVED
- 0xFF11: Decline the offline transaction RETURN_CODE_EMV_OFFLINE_DECLINED

Converting TLV to NSMutableDictionary

EMV data is received in TLV (Tag, Length, value) format: `950500000080009B02E8009F2701018A025↩ A339F26080C552B9364D55CE5`

This data contains the following EMV tags/values:

Tag	Length	Value
9502	06	000000001995
9A	03	140530
9C	01	00

An example how to create an NSMutableDictionary with these values follows.

```
-(NSMutableDictionary*) createTLVDict{

NSMutableDictionary *emvTags = [[NSMutableDictionary alloc] initWithCapacity:0];

[emvTags setObject:@"000000001995" forKey:@"9502"];
[emvTags setObject:@"140530" forKey:@"9A"];
[emvTags setObject:@"00" forKey:@"9C"];

return emvTags;

}
```

15.9.2.2 - (void) bluetoothDeviceNames: (NSArray ∗) *names* [optional]

Bluetooth Device Names When a bluetooth name scan is requested, this delegate will return a NSArray with the names of the found devices

Parameters

names	NSArray of device names:

15.9.2.3 - (void) bluetoothPickerAlert: (UIAlertView ∗) *view* [optional]

Bluetooth Picker Alert When a bluetooth scan is requested, this delegate will return an UIAlertView for displaying to allow the selection of a found device

Parameters

view	UIAlert↩ View:

15.9.2.4 - (void) dataInOutMonitor: (NSData *) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the device can be monitored through this delegate.

Parameters

data	The serial data represented as a NSData object
isIncoming	The direction of the data • TRUE specifies data being received from the device, • FALSE indicates data being sent to the device.

15.9.2.5 - (void) deviceMessage: (NSString *) *message* [optional]

Receives messages from the framework

Parameters

message	String message transmitted by framework

15.9.2.6 - (void) emvTransactionData: (IDTEMVData *) *emvData* errorCode:(int) *error* [optional]

EMV Transaction Data

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

Parameters

emvData	EMV Results Data. Result code, card type, encryption type, masked tags, encrypted tags, unencrypted tags and KSN
error	The error code as defined in the errors.h file

15.9.2.7 - (void) lcdDisplay: (int) *mode* lines:(NSArray *) *lines* [optional]

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

Parameters

mode	LCD Display Mode: • 0x01: Menu Display. A selection must be made to resume the transaction • 0x02: Normal Display get function key. A function must be selected to resume the transaction • 0x03: Display without input. Message is displayed without pausing the transaction • 0x04: List of languages are presented for selection. A selection must be made to resume the transaction • 0x10: Clear Screen. Command to clear the LCD screen

15.9.2.8 - (void) pinpadData: (NSData ∗) *value* keySN:(NSData ∗) *KSN* event:(EVENT_PINPAD_Types) *event* [optional]

Pinpad data delegate protocol

Receives data from pinpad methods

Parameters

value	encrypted data returned from IDT_VP3300::pin_getEncryptedData:minLength:maxLength:messageID:language:(), or encrypted account number returned from IDT_VP3300::pin_getCardAccount:max:line1:line2:(). String value returned from IDT_VP3300::pin_getAmount:maxLength:messageID:language:() or IDT_VP3300::pin_getNumeric:minLength:maxLength:messageID:language:(). PINblock returned from IDT_VP3300::pin_getEncryptedPIN:keyType:line1:line2:line3:()
KSN	Key Serial Number returned from IDT_VP3300::pin_getEncryptedPIN:keyType:line1:line2:line3:(), IDT_VP3300::pin_getCardAccount:max:line1:line2:() or IDT_VP3300::pin_getEncryptedData:minLength:maxLength:messageID:language:()
event	EVENT_PINPAD_Types PINpad event that solicited the data capture

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

15.9.2.9 - (void) pinRequest: (EMV_PIN_MODE_Types) *mode* key:(NSData ∗) *key* PAN:(NSData ∗) *PAN* startTO:(int) *startTO* intervalTO:(int) *intervalTO* language:(NSString ∗) *language* [optional]

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

Parameters

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

15.9.2.10 - (RETURN_CODE) sendPKUpdate: (NSData ∗) *pkFile* [optional]

Sends a PK Update

Starts a device firmware upatate using the provided path to the .pk file. Update proceeds on a background thread. Use protocol UpdateStatus to monitor progress. Keep device attached and do interrupt process until PK_STATU↩ S_FAILED or PK_STATUS_COMPLETED has been received

Parameters

pkFile	The .pk file

Returns

RETURN_CODE:

- 0x0000: Success: no error - RETURN_CODE_DO_SUCCESS
- 0x0001: Disconnect: no response from reader - RETURN_CODE_ERR_DISCONNECT
- 0x0002: Invalid Response: invalid response data - RETURN_CODE_ERR_CMD_RESPONSE
- 0x0003: Timeout: time out for task or CMD - RETURN_CODE_ERR_TIMEDOUT
- 0x0004: Invalid Parameter: wrong parameter - RETURN_CODE_ERR_INVALID_PARAMETER
- 0x0005: MSR Busy: SDK is doing MSR or ICC task - RETURN_CODE_SDK_BUSY_MSR
- 0x0006: PINPad Busy: SDK is doing PINPad task - RETURN_CODE_SDK_BUSY_PINPAD
- 0x0007: Unknown: Unknown error - RETURN_CODE_ERR_OTHER
- 0x0100 through 0xFFFF refer to IDT_Device::getResponseCodeString:()

15.9.2.11 - (RETURN_CODE) sendPKUpdateBLE: (NSData ∗) *pkData* [optional]

Sends a PK Update (Faster)

Starts a device firmware upatate using the provided path to the .pk file. Update proceeds on a background thread. Use protocol UpdateStatus to monitor progress. Keep device attached and do interrupt process until PK_STATU↩ S_FAILED or PK_STATUS_COMPLETED has been received

Data will be sent in packet size of 244 bytes. Sleep timers are modified or removed where possible to achieve faster times During transfer, no response will be sent by the device.

Parameters

pkData	The .pk file

Returns

RETURN_CODE: Values can be parsed with IDT_Device::getResponseCodeString:()

15.9.2.12 - (void) swipeMSRData: (IDTMSRData ∗) *cardData* [optional]

Receives card data from MSR swipe.

Parameters

cardData	Captured card data from MSR swipe

15.9.2.13 - (void) updateStatus: (PK_STATUS_Type) *type* currentBlock:(int) *currentBlock* totalBlocks:(int) *totalBlocks* error:(RETURN_CODE) *error* [optional]

Reports PK Update status.

Parameters

type	The stage of the PK update
currentBlock	The number of the block that has transferred
totalBlocks	The total number of blocks to transfer
error	The error condition when failure is encoutered

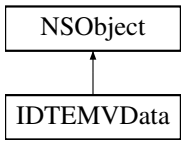
The documentation for this protocol was generated from the following file:

- Source_iOS/IDT_NEO2.h

15.10 IDTEMVData Class Reference

`#import <IDTEMVData.h>`

Inheritance diagram for IDTEMVData:



Instance Methods

- (void) - clear

Class Methods

- (IDTEMVData ∗) + sharedController

Properties

- EMV_RESULT_CODE_Types resultCode
- EMV_RESULT_CODE_V2_Types resultCodeV2
- int encryptionMode

  0 = TDES, 1 = AES
- int cardType

  0 = Contact, 1 = Contactless
- bool hasAdvise

  TRUE if response has an Advise request.
- bool hasReversal

  TRUE if response has reversal request.
- NSDictionary ∗ unencryptedTags

  Unencrypted EMV Tags. Key = tag name (NSString), Object = tag value (NSData)
- NSDictionary ∗ encryptedTags

Encrypted EMV Tags. Key = tag name (NSString), Object = tag value (NSData)

- NSDictionary ∗ maskedTags

  Encrypted EMV Tags. Key = tag name (NSString), Object = tag value (NSData)

- NSData ∗ KSN

  Key Serial Number for encrypted EMV tags.

- IDTMSRData ∗ cardData

  Card data returned from fallback or non-icc swipe during emv transaction.

15.10.1 Detailed Description

Encapsulating data class for EMV data capture

15.10.2 Method Documentation

15.10.2.1 - (void) clear

clears all IDTEMVData properties

15.10.2.2 + (IDTEMVData ∗) sharedController

Singleton instance of IDTEMVData

15.10.3 Property Documentation

15.10.3.1 - (EMV_RESULT_CODE_Types) resultCode [read], [write], [atomic]

Result Code. Uses enumeration EMV_RESULT_CODE_Types

```
typedef enum{
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_Types;
```

15.10.3.2 - (EMV_RESULT_CODE_V2_Types) resultCodeV2 [read], [write], [atomic]

Result Code. Uses enumeration EMV_RESULT_CODE_V2_Types

```
 typedef enum{
     EMV_RESULT_CODE_V2_NO_RESPONSE = -1,
EMV_RESULT_CODE_V2_APPROVED_OFFLINE = 0x0000,
```

```
EMV_RESULT_CODE_V2_DECLINED_OFFLINE = 0x0001,
EMV_RESULT_CODE_V2_APPROVED = 0x0002,
EMV_RESULT_CODE_V2_DECLINED = 0x0003,
EMV_RESULT_CODE_V2_GO_ONLINE = 0x0004,
EMV_RESULT_CODE_V2_CALL_YOUR_BANK = 0x0005,
EMV_RESULT_CODE_V2_NOT_ACCEPTED = 0x0006,
EMV_RESULT_CODE_V2_USE_MAGSTRIPE = 0x0007,
EMV_RESULT_CODE_V2_TIME_OUT = 0x0008,
EMV_RESULT_CODE_V2_START_TRANS_SUCCESS = 0x0010,
EMV_RESULT_CODE_V2_SWIPE_NON_ICC = 0x0011,
EMV_RESULT_CODE_V2_TRANSACTION_CANCELLED = 0x0012,
EMV_RESULT_CODE_V2_GO_ONLINE_CTLS = 0x0023,
EMV_RESULT_CODE_CTLS_TWO_CARDS = 0x7A,
EMV_RESULT_CODE_CTLS_TERMINATE = 0x7E,
EMV_RESULT_CODE_CTLS_TERMINATE_TRY_ANOTHER = 0x7D,
EMV_RESULT_CODE_MSR_SWIPE_CAPTURED = 0x80,
EMV_RESULT_CODE_REQUEST_ONLINE_PIN = 0x81,
EMV_RESULT_CODE_REQUEST_SIGNATURE = 0x82,
EMV_RESULT_CODE_FALLBACK_TO_CONTACT = 0x83,
EMV_RESULT_CODE_FALLBACK_TO_OTHER = 0x84,
EMV_RESULT_CODE_REVERSAL_REQUIRED = 0x85,
EMV_RESULT_CODE_ADVISE_REQUIRED = 0x86,
EMV_RESULT_CODE_ADVISE_REVERSAL_REQUIRED = 0x87,
EMV_RESULT_CODE_NO_ADVISE_REVERSAL_REQUIRED = 0x88,
EMV_RESULT_CODE_UNABLE_TO_REACH_HOST = 0xFF,
EMV_RESULT_CODE_V2_FILE_ARG_INVALID = 0x1001,
EMV_RESULT_CODE_V2_FILE_OPEN_FAILED = 0x1002,
EMV_RESULT_CODE_V2_FILE_OPERATION_FAILED = 0X1003,
EMV_RESULT_CODE_V2_MEMORY_NOT_ENOUGH = 0x2001,
EMV_RESULT_CODE_V2_SMARTCARD_FAIL = 0x3001,
EMV_RESULT_CODE_V2_SMARTCARD_INIT_FAILED = 0x3003,
EMV_RESULT_CODE_V2_FALLBACK_SITUATION = 0x3004,
EMV_RESULT_CODE_V2_SMARTCARD_ABSENT = 0x3005,
EMV_RESULT_CODE_V2_SMARTCARD_TIMEOUT = 0x3006,
EMV_RESULT_CODE_V2_MSR_CARD_ERROR = 0x3007,
EMV_RESULT_CODE_V2_MSR_CARD_READ_ERROR = 0x3012,
EMV_RESULT_CODE_V2_TIMEOUT_INSERT_OR_FALLBACK = 0x3013,
EMV_RESULT_CODE_V2_PARSING_TAGS_FAILED= 0X5001,
EMV_RESULT_CODE_V2_CARD_DATA_ELEMENT_DUPLICATE = 0X5002,
EMV_RESULT_CODE_V2_DATA_FORMAT_INCORRECT = 0X5003,
EMV_RESULT_CODE_V2_APP_NO_TERM = 0X5004,
EMV_RESULT_CODE_V2_APP_NO_MATCHING = 0X5005,
EMV_RESULT_CODE_V2_AMANDATORY_OBJECT_MISSING = 0X5006,
EMV_RESULT_CODE_V2_APP_SELECTION_RETRY = 0X5007,
EMV_RESULT_CODE_V2_AMOUNT_ERROR_GET = 0X5008,
EMV_RESULT_CODE_V2_CARD_REJECTED = 0X5009,
EMV_RESULT_CODE_V2_AIP_NOT_RECEIVED = 0X5010,
EMV_RESULT_CODE_V2_AFL_NOT_RECEIVEDE = 0X5011,
EMV_RESULT_CODE_V2_AFL_LEN_OUT_OF_RANGE = 0X5012,
EMV_RESULT_CODE_V2_SFI_OUT_OF_RANGE = 0X5013,
EMV_RESULT_CODE_V2_AFL_INCORRECT = 0X5014,
EMV_RESULT_CODE_V2_EXP_DATE_INCORRECT = 0X5015,
EMV_RESULT_CODE_V2_EFF_DATE_INCORRECT = 0X5016,
EMV_RESULT_CODE_V2_ISS_COD_TBL_OUT_OF_RANGE = 0X5017,
EMV_RESULT_CODE_V2_CRYPTOGRAM_TYPE_INCORRECT = 0X5018,
EMV_RESULT_CODE_V2_PSE_BY_CARD_NOT_SUPPORTED = 0X5019,
EMV_RESULT_CODE_V2_USER_LANGUAGE_SELECTED = 0X5020,
EMV_RESULT_CODE_V2_SERVICE_NOT_ALLOWED = 0X5021,
EMV_RESULT_CODE_V2_NO_TAG_FOUND = 0X5022,
EMV_RESULT_CODE_V2_CARD_BLOCKED = 0X5023,
EMV_RESULT_CODE_V2_LEN_INCORRECT = 0X5024,
EMV_RESULT_CODE_V2_CARD_COM_ERROR = 0X5025,
EMV_RESULT_CODE_V2_TSC_NOT_INCREASED = 0X5026,
EMV_RESULT_CODE_V2_HASH_INCORRECT = 0X5027,
EMV_RESULT_CODE_V2_ARC_NOT_PRESENCED = 0X5028,
EMV_RESULT_CODE_V2_ARC_INVALID = 0X5029,
EMV_RESULT_CODE_V2_COMM_NO_ONLINE = 0X5030,
EMV_RESULT_CODE_V2_TRAN_TYPE_INCORRECT = 0X5031,
EMV_RESULT_CODE_V2_APP_NO_SUPPORT = 0X5032,
EMV_RESULT_CODE_V2_APP_NOT_SELECT = 0X5033,
EMV_RESULT_CODE_V2_LANG_NOT_SELECT = 0X5034,
EMV_RESULT_CODE_V2_TERM_DATA_NOT_PRESENCED = 0X5035,
EMV_RESULT_CODE_V2_CVM_TYPE_UNKNOWN = 0X6001,
EMV_RESULT_CODE_V2_CVM_AIP_NOT_SUPPORTED = 0X6002,
EMV_RESULT_CODE_V2_CVM_TAG_8E_MISSING = 0X6003,
EMV_RESULT_CODE_V2_CVM_TAG_8E_FORMAT_ERROR = 0X6004,
EMV_RESULT_CODE_V2_CVM_CODE_IS_NOT_SUPPORTED = 0X6005,
EMV_RESULT_CODE_V2_CVM_COND_CODE_IS_NOT_SUPPORTED = 0X6006,
EMV_RESULT_CODE_V2_CVM_NO_MORE = 0X6007,
EMV_RESULT_CODE_V2_PIN_BYPASSED_BEFORE = 0X6008,
EMV_RESULT_CODE_V2_GO_ONLINE_CL = 0xEE23
 } EMV_RESULT_CODE_V2_Types;
```

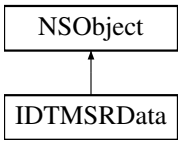
The documentation for this class was generated from the following file:

- Source_iOS/IDTEMVData.h

15.11 IDTMSRData Class Reference

`#import <IDTMSRData.h>`

Inheritance diagram for IDTMSRData:



Instance Methods

- (void) - clear

Class Methods

- (IDTMSRData ∗) + sharedController

Properties

- KEY_VARIANT_TYPE **msr_keyVariantType**
- EVENT_MSR_Types event
- CAPTURE_ENCODE_TYPE captureEncodeType
- CAPTURE_ENCRYPT_TYPE captureEncryptType
- CAPTURE_CARD_TYPE **captureCardType**
- NSData ∗ mac

  mac
- NSData ∗ macKSN

  macKSN
- NSData ∗ cardData

  Complete unparsed swipe data as received from MSR.
- NSString ∗ kbOutput

  Contains output converted to KB format.
- NSString ∗ track1

  Track 1 masked if encryption enabled or cleartext if encryption disabled.
- NSString ∗ track2

  Track 2 masked if encryption enabled or cleartext if encryption disabled.
- NSString ∗ track3

  Track 3 masked if encryption enabled or cleartext if encryption disabled.
- int track1Length

  Length of track 1 masked/clear text data.
- int track2Length

  Length of track 2 masked/clear text data.
- int track3Length

  Length of track 3 masked/clear text data.
- NSData ∗ rawData

  rawData
- NSData ∗ msr_extendedField

  MSR Extended Field.

- NSData * encTrack1

  Track 1 encoded data OR all encoded track data if encryption method combines all tracks into single blob.
- NSData * encTrack2

  Track 2 encoded.
- NSData * encTrack3

  Track 3 encoded.
- NSData * hashTrack1

  Sha-256 hash of Track 1 encoded data.
- NSData * hashTrack2

  Sha-256 hash of Track 2 encoded data.
- NSData * hashTrack3

  Sha-256 hash of Track 3 encoded data.
- NSString * RSN

  Reader Serial Number.
- NSData * KSN

  Key Serial Number.
- NSData * sessionID

  Session ID - Security level 4 only.
- unsigned char readStatus
- int errorCode

  Contains error code when data is not returned.
- bool iccPresent

  Card contains ICC.
- NSDictionary * unencryptedTags

  Unencrypted card data provided via TLV.
- NSDictionary * encryptedTags

  Encrypted card data provided via TLV.
- NSDictionary * maskedTags

  Masked card data provided via TLV.

15.11.1 Detailed Description

Encapsulating data class for MSR data capture

15.11.2 Method Documentation

15.11.2.1 - (void) clear

clears all IDTMSRData properties

15.11.2.2 + (IDTMSRData *) sharedController

Singleton instance of IDTMSRData

15.11.3 Property Documentation

15.11.3.1 - (CAPTURE_ENCODE_TYPE) captureEncodeType [read], [write], [atomic]

Encode Type of captured MSR Data.

Uses enumeration CAPTURE_ENCODE_TYPE:

```
typedef enum{
   CAPTURE_ENCODE_TYPE_ISOABA=0,
   CAPTURE_ENCODE_TYPE_AAMVA=1,
   CAPTURE_ENCODE_TYPE_Other=3,
   CAPTURE_ENCODE_TYPE_Raw=4
} CAPTURE_ENCODE_TYPE;
```

15.11.3.2 - (CAPTURE_ENCRYPT_TYPE) captureEncryptType [read], [write], [atomic]

Encrypt Type of captured MSR Data.

Uses enumeration CAPTURE_ENCODE_TYPE:

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

15.11.3.3 - (EVENT_MSR_Types) event [read], [write], [atomic]

Event type. Uses enumeration EVENT_MSR_Types

```
typedef enum{
   EVENT_MSR_UNKNOWN = 31,
   EVENT_MSR_CARD_DATA,
   EVENT_MSR_CANCEL_KEY,
   EVENT_MSR_BACKSPACE_KEY,
   EVENT_MSR_ENTER_KEY,
   EVENT_MSR_DATA_ERROR,
   EVENT_MSR_ICC_START,
   EVENT_BTPAY_CARD_DATA,
   EVENT_UNIPAYII_EMV_NO_ICC_MSR_DATA,
   EVENT_UNIPAYII_EMV_FALLBACK_DATA,
   EVENT_UNIPAY_KEYLOADING,
   EVENT_MSR_TIMEOUT
}EVENT_MSR_Types;
```

15.11.3.4 - (unsigned char) readStatus [read], [write], [atomic]

Track Read Status

- Bit 0: 1=Track 1 decode success, 0=Track 1 decode fail
- Bit 1: 1=Track 2 decode success, 0=Track 2 decode fail)
- Bit 2: 1=Track 3 decode success, 0=Track 3 decode fail)
- Bit 3: 1=Track 1 sampling data exists, 0=Track 1 sampling data does not exist
- Bit 4: 1=Track 2 sampling data exists, 0=Track 2 sampling data does not exist
- Bit 5: 1=Track 3 sampling data exists, 0=Track 3 sampling data does not exist
- Bit 6: reserved for future use
- Bit 7: reserved for future use

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

- Source_iOS/IDTMSRData.h

15.12 MaskAndEncryption Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- unsigned char prePANClear

  Leading PAN digits to display. Values '0' - '6'. Default '4'.
- unsigned char postPANClear

  Last PAN digits to display. Values '0' - '4'. Default '4'.
- unsigned char maskChar

  Last PAN digits to display. Values 0x20-0x7E. Default 0x2A '∗'.
- unsigned char displayExpDate

  Mask or display expiration date. Values '0' = mask, '1' = don't mask. Default '1'.
- unsigned char baseKeyType

  BTPay Only. Key Type. Values '0' = Data Key, '1' = Pin Key. Default '0'.
- unsigned char encryptionType

  BTPay Only. Key Type. Values '1' = TDES, '2' = AES. Default '1'.
- unsigned char encryptionOption
- unsigned char maskOption

15.12.1 Detailed Description

Mask and Encryption - Used to Set/Retrieve mask and encryption values IDT_BTPay::emv_retrieveAIDList:() IDT↩ _UniPay::emv_retrieveAIDList:().

15.12.2 Member Data Documentation

15.12.2.1 unsigned char MaskAndEncryption::encryptionOption

UniPay II Only. Bit 0: T1 force encrypt Bit 1 : T2 force encrypt Bit 2 : T3 force encrypt Bit3 : T3 force encrypt when card type is 0

15.12.2.2 unsigned char MaskAndEncryption::maskOption

UniPay II Only. Bit0: T1 mask allowed Bit1: T2 mask allowed Bit2: T3 mask allowed

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

- Source_iOS/IDTCommon.h

15.13 PowerOnStructure Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- BOOL sendIFS

  Send S(IFS) request if T=1 protocolError: Reference source not found.
- BOOL explicitPPS

Explicit PPSError: Reference source not found.

- BOOL disableAutoPPS

  No auto pps for negotiate mode.

- BOOL disableResponseCheck

  No check on response of S(IFS) request.

- unsigned char * 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.

- unsigned char ppsLength

  lenght of pps data

15.13.1 Detailed Description

Structure to set ICC power on options. Used by IDT_BTPay::icc_powerOnICC:response:() IDT_UniPay::icc_↩ powerOnICC:response:()

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

- Source_iOS/IDTCommon.h

15.14 TerminalData Struct Reference

`#include <IDTCommon.h>`

Public Attributes

- unsigned char terminalCountryCode [2]

  Terminal's location. Tag 9F1A {0x08,0x40}.

- unsigned char provideCardholderConfirmation

  Indicates whether or not cardholder may confirm application selection at EMV Selection time. Tag 58 0x00 or 0x01.

- unsigned char terminalType

  Standard parameter. Tag 9F35 See EMVCo book IV.

- unsigned char emvContact

  Indicates whether terminal supports EMV contact. Tag 9F33, byte 1, bit 6 0x00 or 0x01.

- unsigned char terminalCapabilities [3]

  Standard parameter. Tag 9F33 See EMVCo book IV.

- unsigned char additionalTerminalCapabilities [5]

  Standard parameter. Tag 9F40 See EMVCo book IV.

- unsigned char emvContactless

  Indicates whether or not terminal support scontactless in EMV mode. 0x00 or 0x01.

- unsigned char magstripe

  Indicates whether terminal supports magstripe. 0x00 or 0x01.

- unsigned char pinTimeOut

  In seconds. Time allocated to cardholder to enter PIN. Binary value Example : 0x0F for 15s.

- unsigned char batchManaged

  Indicates whether or not Batch messages are supported by Terminal. 0x00 or 0x01.

- unsigned char adviceManaged

  Indicates whether or not Advice messages are supported by Terminal (only if needed by Level3 implementation). 0x00 or 0x01.

- unsigned char pse

Indicates whether or not PSE Selection method is supported by Terminal. 0x00 or 0x01.

- unsigned char autoRun

  Indicates whether or not Terminal is configured in AutoRun. 0x00 or 0x01.

- unsigned char predefinedAmount [3]

  Fixed amount. Binary value.

- unsigned char pinByPass

  Indicates whether or not PIN bypass is supported by Terminal. 0x00 or 0x01.

- unsigned char referalManaged

  Indicates whether or not Referal managed are supported by Terminal (only if needed by Level3 implementation).. 0x00 or 0x01.

- unsigned char defaultTAC

  Indicates whether or not Default TAC are supported by Terminal. 0x00 or 0x01.

- unsigned char defaultTACDenial [5]

  Default TAC Denial value. See EMVCo book IV.

- unsigned char defaultTACOnline [5]

  Default TAC Online value. See EMVCo book IV.

- unsigned char defaultTACDefault [5]

  Default TAC Default value. See EMVCo book IV.

- unsigned char notRTS

  Indicates RTS are not supported by Terminal or not. 0x00 or 0x01.

- unsigned char notVelocity

  Indicates Velocity are not supported by Terminal or not. 0x00 or 0x01.

- unsigned char cdaType

  Supported CDA type. Value should be 0x02.

15.14.1 Detailed Description

device Terminal Configuration File - 44 bytes

Used as parameter in IDT_BTPay::setTerminalData:()

Used as return value in IDT_BTPay::emv_retrieveTerminalData:()

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

- Source_iOS/IDTCommon.h

Index