



Apple iOS SDK Guide for VP3300

#80149513-001

Rev. A



Revision History

Revision	Description and Reason for Change	Date
A	Initial Release - Manual;User;VP3300;SDK;iOS	7/19/2017

Contents

1	IDTech iOS SDK Reference Guide for VP3300	1
2	Connecting to VP3300 BLE With Automatic Location	3
3	Connecting to VP3300 BLE With Manual Location	4
4	Important Security Notice	5
4.1	Applicability	5
4.2	What Does PA-DSS Mean to You?	5
4.3	Third Party Applications	6
4.4	PA-DSS Guidelines	6
4.5	More Information	11
5	VP3300 Main Transaction Commands	13
5.1	EMV Methods	13
5.2	MSR/CTLS MSD	15
6	EMV LCD Display Callback	16
7	EMV PIN Callback	17
8	Sending Direct Commands	20
9	Core Implementation VP3300: Objective-C	21
9.1	Integrating with IDTech framework	21
9.2	Import the necessary framework/libraries	21
9.3	Add Import statements to utilize frameworks	27
9.4	Amend the view controller interface to include the framework delegate classes:	27
9.5	Implement any/all of the optional delegate protocols used to receive data from IDT_VP3300_↔ Delegate:	27
9.6	Call the Singleton instance of the IDT_VP3300 framework object:	28
9.7	Sample Project Tutorial	28
9.7.1	Step 1: Create New Project	28
9.7.2	Step 2: Import Frameworks	29
9.7.3	Step 3: Design Interface	29

9.7.4	Step 4: Configure Header File	30
9.7.5	Step 5: Configure Method File	34
10	Core Implementation VP3300: Swift	39
10.1	Integrating with IDTech framework	39
10.2	Import the Necessary Framework/Libraries	39
10.3	Create a Bridging Header File	45
10.4	Add Import Statement to the Bridging Header File	50
10.5	Amend the View Controller Interface	50
10.6	Implement Optional Delegate Protocols	50
10.7	Allocate/Initialize IDT_VP3300 Object	51
10.8	Sample Project Tutorial	51
10.8.1	Step 1: Create New Project	52
10.8.2	Step 2: Import Frameworks	53
10.8.3	Step 3: Design Interface	53
10.8.4	Step 4: Configure the Bridging Header and View Controller Files	54
10.8.5	Step 5: Finalize the View Controller File	58
11	VP3300 Error Code Reference	64
12	Enumeration Reference	66
13	EMV Tag Reference	70
14	LCD Foreign Language Mapping Table	83
15	Hierarchical Index	85
15.1	Class Hierarchy	85
16	Class Index	86
16.1	Class List	86
17	Class Documentation	87
17.1	ADF_Info Struct Reference	87
17.2	AIDEntry Struct Reference	87
17.2.1	Detailed Description	87
17.3	APDUResponse Class Reference	87
17.3.1	Detailed Description	88
17.3.2	Method Documentation	88
17.3.2.1	clear()	88
17.3.2.2	sharedController()	88
17.4	ApplicationID Struct Reference	89
17.4.1	Detailed Description	90

17.5 CAKey Struct Reference	90
17.5.1 Detailed Description	91
17.6 CRLEntry Struct Reference	91
17.6.1 Detailed Description	91
17.7 ICCReaderStatus Struct Reference	91
17.7.1 Detailed Description	92
17.8 IDT_VP3300 Class Reference	92
17.8.1 Detailed Description	94
17.8.2 Method Documentation	95
17.8.2.1 activateTransaction:timeout:(NSMutableDictionary< NSString *, NSString * > *_Nullable tags,[timeout] int timeout)	95
17.8.2.2 assignBypassDelegate:(id< IDT_VP3300_Delegate > del)	96
17.8.2.3 close()	97
17.8.2.4 config_getSerialNumber:(NSString **response)	97
17.8.2.5 createFastEMVData:(IDTEMVData *emvData)	97
17.8.2.6 ctls_cancelTransaction()	97
17.8.2.7 ctls_getAllConfigGroups:(NSData **response)	97
17.8.2.8 ctls_getAllConfigurationGroups:(NSDictionary< NSString *, NSDictionary * > **response)	98
17.8.2.9 ctls_getConfigurationGroup:response:(int group,[response] NSDictionary **response)	98
17.8.2.10 ctls_removeAllCAPK()	98
17.8.2.11 ctls_removeApplicationData:(NSString *AID)	98
17.8.2.12 ctls_removeCAPK:(NSData *capk)	99
17.8.2.13 ctls_removeConfigurationGroup:(int group)	99
17.8.2.14 ctls_retrieveAIDList:(NSArray **response)	99
17.8.2.15 ctls_retrieveApplicationData:response:(NSString *AID,[response] NSDictionary **response)	100
17.8.2.16 ctls_retrieveCAPK:key:(NSData *capk,[key] NSData **key)	100
17.8.2.17 ctls_retrieveCAPKList:(NSArray **keys)	101
17.8.2.18 ctls_retrieveTerminalData:(NSData **tlv)	101
17.8.2.19 ctls_setApplicationData:(NSData *tlv)	101
17.8.2.20 ctls_setCAPK:(NSData *key)	101
17.8.2.21 ctls_setConfigurationGroup:(NSData *tlv)	102
17.8.2.22 ctls_setTerminalData:(NSData *tlv)	102
17.8.2.23 ctls_startTransaction()	103
17.8.2.24 ctls_startTransaction:type:timeout:tags:(double amount,[type] int type,[timeout] int timeout,[tags] NSMutableDictionary *tags)	103
17.8.2.25 device_addTLVToTerminalData:(NSData *tlv)	104
17.8.2.26 device_cancelConnectToAudioReader()	104
17.8.2.27 device_cancelTransaction()	105

17.8.2.28 device_connectedBLEDevice()	105
17.8.2.29 device_connectToAudioReader()	105
17.8.2.30 device_controlLED:control:(Byte indexLED,[control] Byte control)	105
17.8.2.31 device_controlUserInterface:(NSData *values)	106
17.8.2.32 device_disableBLEDeviceSearch()	107
17.8.2.33 device_disconnectBLE()	107
17.8.2.34 device_enableBLEDeviceSearch:(NSUUID *identifier)	107
17.8.2.35 device_getAutoPollTransactionResults:(IDTEMVData **result)	108
17.8.2.36 device_getBLEFriendlyName()	108
17.8.2.37 device_getCashTranRiskPara:(NSData **tlv)	108
17.8.2.38 device_getDateTime:(NSData **dateTime)	108
17.8.2.39 device_getDrlReaderRiskPara:tlv:(Byte index,[tlv] NSData **tlv)	109
17.8.2.40 device_getFirmwareVersion:(NSString **response)	109
17.8.2.41 device_getHardwareInfo:(NSString **response)	109
17.8.2.42 device_getMerchantRecord:record:(int index,[record] NSData **record)	110
17.8.2.43 device_getModuleVer:(NSString **moduleVer)	110
17.8.2.44 device_getPollMode:(NSData **mode)	110
17.8.2.45 device_getProcessorType:(NSData **type)	111
17.8.2.46 device_getProductType:(NSData **type)	111
17.8.2.47 device_getResponseCodeString:(int errorCode)	111
17.8.2.48 device_getTransactionResults:(NSData **results)	112
17.8.2.49 device_getUIDofMCU:(NSString **response)	112
17.8.2.50 device_isAudioReaderConnected()	112
17.8.2.51 device_isConnected:(IDT_DEVICE_Types device)	112
17.8.2.52 device_pingDevice()	113
17.8.2.53 device_rebootDevice()	113
17.8.2.54 device_retrieveTerminalData:(NSData **responseData)	113
17.8.2.55 device_sendGen2Cmd:response:(NSData *tlv,[response] NSData **response)	114
17.8.2.56 device_sendIDGCommand:subCommand:data:response:(unsigned char command,[subCommand] unsigned char subCommand,[data] NSData *data,[response] NSData **response)	114
17.8.2.57 device_sendIDGCommandV3:subCommand:data:response:(unsigned char command,[subCommand] unsigned char subCommand,[data] NSData *data,[response] NSData **response)	114
17.8.2.58 device_setAudioVolume:(float val)	115
17.8.2.59 device_setBLEFriendlyName:(NSString *friendlyName)	115
17.8.2.60 device_setBurstMode:(int mode)	115
17.8.2.61 device_setDateTime()	116
17.8.2.62 device_setMerchantRecord:enabled:merchantID:merchantURL:(int index,[enabled] BOOL enabled,[merchantID] NSString *merchantID,[merchantURL] NSString *merchantURL)	116
17.8.2.63 device_setPassThrough:(BOOL enablePassThrough)	116

17.8.2.64 device_setPollMode:(int mode)	117
17.8.2.65 device_setQuickChipHIDMode:(Byte mode)	117
17.8.2.66 device_setTerminalData:(NSData *tags)	117
17.8.2.67 device_startRKI()	118
17.8.2.68 device_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:(double amount,[amtOther] double amtOther,[type] int type,[timeout] int timeout,[tags] NSData *tags,[forceOnline] BOOL forceOnline,[fallback] BOOL fallback)	118
17.8.2.69 device_syncClock()	119
17.8.2.70 device_syncTime()	119
17.8.2.71 disableAudioDetection()	119
17.8.2.72 emv_authenticateTransaction:(NSData *tags)	119
17.8.2.73 emv_callbackResponseLCD:selection:(int mode,[selection] unsigned char selection)	120
17.8.2.74 emv_callbackResponseMSR:(NSData *MSR)	121
17.8.2.75 emv_callbackResponsePIN:KSN:PIN:(EMV_PIN_MODE_Types mode,[KSN] NSData *KSN,[PIN] NSData *PIN)	121
17.8.2.76 emv_cancelTransaction()	121
17.8.2.77 emv_completeOnlineEMVTransaction:hostResponseTags:(BOOL isSuccess,[hostResponseTags] NSData *tags)	121
17.8.2.78 emv_disableAutoAuthenticateTransaction:(BOOL disable)	122
17.8.2.79 emv_getEMVConfigurationCheckValue:(NSString **response)	123
17.8.2.80 emv_getEMVKernelCheckValue:(NSString **response)	123
17.8.2.81 emv_getEMVKernelVersion:(NSString **response)	123
17.8.2.82 emv_getEMVKernelVersionExt:(NSString **response)	123
17.8.2.83 emv_getEMVL2Version:(NSString **response)	124
17.8.2.84 emv_getTerminalMajorConfiguration:(NSUInteger **configuration)	124
17.8.2.85 emv_removeAllApplicationData()	124
17.8.2.86 emv_removeApplicationData:(NSString *AID)	125
17.8.2.87 emv_removeCAPK:index:(NSString *rid,[index] NSString *index)	125
17.8.2.88 emv_removeCRLList()	126
17.8.2.89 emv_removeTerminalData()	126
17.8.2.90 emv_retrieveAIDList:(NSArray **response)	126
17.8.2.91 emv_retrieveApplicationData:response:(NSString *AID,[response] NSDictionary **responseAID)	127
17.8.2.92 emv_retrieveCAPK:index:response:(NSString *rid,[index] NSString *index,[response] CAKey **response)	128
17.8.2.93 emv_retrieveCAPKFile:index:response:(NSString *rid,[index] NSString *index,[response] NSData **response)	128
17.8.2.94 emv_retrieveCAPKList:(NSArray **response)	129
17.8.2.95 emv_retrieveCRLList:(NSMutableArray **response)	129
17.8.2.96 emv_retrieveTerminalData:(NSDictionary **responseData)	130
17.8.2.97 emv_retrieveTransactionResult:retrievedTags:(NSData *tags,[retrievedTags] NSDictionary **retrievedTags)	130

17.8.2.98	emv_setApplicationData:configData:(NSString *aidName,[configData] NSDictionary *data)	131
17.8.2.99	emv_setCAPK:(CAKey key)	132
17.8.2.100	emv_setCAPKFile:(NSData *file)	132
17.8.2.101	emv_setCRLEntries:(NSData *data)	133
17.8.2.102	emv_setTerminalData:(NSDictionary *data)	134
17.8.2.103	emv_setTerminalMajorConfiguration:(int configuration)	134
17.8.2.104	emv_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:(double amount,[amtOther] double amtOther,[type] int type,[timeout] int timeout,[tags] NSData *tags,[forceOnline] BOOL forceOnline,[fallback] BOOL fallback)	135
17.8.2.105	cc_exchangeAPDU:response:(NSData *dataAPDU,[response] APDUResponse **response)	136
17.8.2.106	cc_getICCRReaderStatus:(ICCRReaderStatus **readerStatus)	136
17.8.2.107	cc_getKeyFormatForICCDUKPT:(NSData **format)	136
17.8.2.108	cc_powerOffICC:(NSString **error)	137
17.8.2.109	cc_powerOnICC:(NSData **response)	137
17.8.2.110	cc_setKeyFormatForICCDUKPT:(NSData *encryption)	137
17.8.2.111	isConnected()	137
17.8.2.112	msr_cancelMSRSwipe()	138
17.8.2.113	msr_startMSRSwipe()	138
17.8.2.114	processBypassResponse:(NSData *data)	138
17.8.2.115	scanForBLEDeviceNames:serviceUUIDs:options:(NSTimeInterval scanTime,[serviceUUIDs] nullable NSArray< CBUUID * > *serviceUUIDs,[options] nullable NSDictionary< NSString *, id > *options)	138
17.8.2.116	scanForBLEDevices:serviceUUIDs:options:(NSTimeInterval scanTime,[serviceUUIDs] nullable NSArray< CBUUID * > *serviceUUIDs,[options] nullable NSDictionary< NSString *, id > *options)	139
17.8.2.117	SDK_version()	139
17.8.2.118	setBLEDeviceTypeVP3300()	139
17.8.2.119	setReaderAttached:(BOOL attached)	139
17.8.2.120	setServiceUUID:(nullable NSArray< CBUUID * > *serviceUUIDs)	139
17.8.2.121	sharedController()	140
17.8.3	Property Documentation	140
17.8.3.1	bypassDelegate	140
17.8.3.2	delegate	140
17.9	<IDT_VP3300_Delegate> Protocol Reference	140
17.9.1	Detailed Description	141
17.9.2	Method Documentation	141
17.9.2.1	bluetoothDeviceNames:(NSArray *names)	141
17.9.2.2	bluetoothPickerAlert:(UIAlertView *view)	141
17.9.2.3	bypassData:(NSData *data)	141
17.9.2.4	ctlsEvent:scheme:data:(Byte event,[scheme] Byte scheme,[data] Byte data)	141

17.9.2.5	<code>dataInOutMonitor:incoming:(NSData *data,[incoming] BOOL isIncoming)</code>	142
17.9.2.6	<code>deviceMessage:(NSString *message)</code>	142
17.9.2.7	<code>emvTransactionData:errorCode:(IDTEMVData *emvData,[errorCode] int error)</code>	142
17.9.2.8	<code>gen2Data:(NSData *tlv)</code>	142
17.9.2.9	<code>lcdDisplay:lines:(int mode,[lines] NSArray *lines)</code>	143
17.9.2.10	<code>pinRequest:key:PIN:startTO:intervalTO:language:(EMV_PIN_MODE_Types mode,[key] NSData *key,[PIN] NSData *PIN,[startTO] int startTO,[intervalTO] int intervalTO,[language] NSString *language)</code>	143
17.9.2.11	<code>plugStatusChange:(BOOL deviceInserted)</code>	143
17.9.2.12	<code>swipeMSRData:(IDTMSRData *cardData)</code>	144
17.10	<code>IDTech_ApplicationID Struct Reference</code>	144
17.10.1	Detailed Description	145
17.11	<code>IDTech_TerminalData Struct Reference</code>	145
17.11.1	Detailed Description	145
17.12	<code>IDTechEMV Class Reference</code>	146
17.12.1	Detailed Description	146
17.12.2	Method Documentation	147
17.12.2.1	<code>accelerateRead:(BOOL enabled)</code>	147
17.12.2.2	<code>cancelTransaction()</code>	147
17.12.2.3	<code>completeOnlineEMVTransaction:resultCode:issuerAuthenticationData:issuerScripts:(EMV_COMPLETION_RESULT result,[resultCode] NSString *rc,[issuerAuthenticationData] NSString *iad,[issuerScripts] NSString *scripts)</code>	147
17.12.2.4	<code>confirmApplication:(int index)</code>	147
17.12.2.5	<code>confirmApplicationCancel()</code>	148
17.12.2.6	<code>getAllTags()</code>	148
17.12.2.7	<code>getBuildVersion()</code>	148
17.12.2.8	<code>getEMVConfigurationChecksum()</code>	148
17.12.2.9	<code>getEMVKernelChecksum()</code>	148
17.12.2.10	<code>getEMVKernelVersion()</code>	148
17.12.2.11	<code>getTag:(NSString *tagName)</code>	149
17.12.2.12	<code>preloadEMVCardData()</code>	150
17.12.2.13	<code>removeApplicationData:(NSString *AID)</code>	150
17.12.2.14	<code>retrieveAIDList()</code>	150
17.12.2.15	<code>retrieveApplicationData:(NSString *AID)</code>	150
17.12.2.16	<code>retrieveApplicationData:additionalTags:(NSString *AID,[additionalTags] NSData **additionalTags)</code>	151
17.12.2.17	<code>retrieveTerminalData()</code>	151
17.12.2.18	<code>sendSystemError()</code>	151
17.12.2.19	<code>setApplicationData:(IDTech_ApplicationID data)</code>	151
17.12.2.20	<code>setApplicationData:additionalTags:(IDTech_ApplicationID data,[additionalTags] NSData *additionalTags)</code>	151
17.12.2.21	<code>setTag:tagData:(NSString *tagName,[tagData] NSData *data)</code>	152

17.12.2.22	setTerminalData:(IDTech_TerminalData data)	152
17.12.2.23	sharedController()	152
17.12.2.24	startEMVTransaction:timeout:transactionType:additionalTags:(float amount,[timeout] int timeout,[transactionType] int type,[additionalTags] NSMutableDictionary *tags)	152
17.12.3	Property Documentation	153
17.12.3.1	delegate	153
17.13	<IDTechEMV_Delegate> Protocol Reference	153
17.13.1	Detailed Description	154
17.13.2	Method Documentation	154
17.13.2.1	confirmApplicationSelection:retry:(NSArray *labelArray,[retry] BOOL tryAgain)	154
17.13.2.2	emvTransactionData:errorCode:performReversal:(IDTEMVData *emvData,[error↵ Code] int error,[performReversal] BOOL reversal)	154
17.13.2.3	emvTransactionMessage:(MESSAGE_Types message)	154
17.13.2.4	languagePreference:(NSData *lang)	154
17.14	IDTEMVData Class Reference	155
17.14.1	Detailed Description	156
17.14.2	Method Documentation	156
17.14.2.1	clear()	156
17.14.2.2	sharedController()	156
17.14.3	Property Documentation	156
17.14.3.1	resultCode	156
17.14.3.2	resultCodeV2	156
17.15	IDTMSRData Class Reference	157
17.15.1	Detailed Description	159
17.15.2	Method Documentation	159
17.15.2.1	clear()	159
17.15.2.2	sharedController()	159
17.15.3	Property Documentation	159
17.15.3.1	captureEncodeType	159
17.15.3.2	captureEncryptType	159
17.15.3.3	event	160
17.15.3.4	readStatus	160
17.16	MaskAndEncryption Struct Reference	160
17.16.1	Detailed Description	161
17.16.2	Member Data Documentation	161
17.16.2.1	encryptionOption	161
17.16.2.2	maskOption	161
17.17	PowerOnStructure Struct Reference	161
17.17.1	Detailed Description	162
17.18	TerminalData Struct Reference	162

17.18.1 Detailed Description	163
Index	164

Chapter 1

IDTech iOS SDK Reference Guide for VP3300



IDTech.framework is an Apple Framework that will be provided by IDTech as the main interface between iOS applications, the VP3300 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.

- [Connecting to VP3300 BLE With Automatic Location](#)
- [Connecting to VP3300 BLE With Manual Location](#)
- [Core Implementation VP3300: Objective-C](#)
- [Core Implementation VP3300: Swift](#)

- [Important Security Notice](#)
- [VP3300 Main Transaction Commands](#)
- [EMV LCD Display Callback](#)
- [EMV PIN Callback](#)
- [Sending Direct Commands](#)
- [EMV Tag Reference](#)
- [Enumeration Reference](#)
- [VP3300 Error Code Reference](#)
- [LCD Foreign Language Mapping Table](#)

Chapter 2

Connecting to VP3300 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 VP3300, define your class as the delegate, and then enable the SDK device search.

First, make sure your class is defined as a delegate for the SDK

```
//Objective-C
@interface ViewController : UIViewController<IDT_VP3300_Delegate>

//Swift
class ViewController: UIViewController, IDT_VP3300_Delegate
```

Then execute the following code. In this example, we are asking SDK to search for 2 seconds for all VP3300 in the vicinity

```
//Objective-C
NSString* SPS_SERVICE_3300 = @"1820";
[[IDT_VP3300 sharedController] setDelegate:self];
[[IDT_VP3300 sharedController] scanForBLEDevices:2.0 serviceUUIDs:@[[CBUUID UUIDWithString:
    SPS_SERVICE_3300]] options:nil];

//Swift
IDT_VP3300.sharedController().delegate = self
let svc = CBUUID.init(string: "1820")
let svcArray = Array<CBUUID>.init(arrayLiteral: svc)
IDT_VP3300.sharedController()?.scan(forBLEDevices: 2.0, serviceUUIDs:
    svcArray, 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 automatically process the selection.

Chapter 3

Connecting to VP3300 BLE With Manual Location

A version of VP3300 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 its 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 VP3300 has a default friendly name of "Enzytek SPS_S". This is the default name the SDK uses to connect to the first VP3300 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 \(IDT_VP3300\)](#)

If the VP3300 friendly name is different than "Enzytek SPS-S", then the SDK can be set to recognize this new name: [device_setBLEFriendlyName: \(IDT_VP3300\)](#)

The API method to start BLE scanning : `IDT_VP3300::device_enableBLEDeviceSearch:identifier:()`

The very first time a VP3300 is used with a particular iOS device, it must be located by its friendly name (as the device identifier hasn't been calculated by the iOS). If there are multiple VP3300's in range, it will stop at the first one located.

```
[[IDT_VP3300 sharedController] device_enableBLEDeviceSearch:nil];
```

Once the device is connected, its UUID can be retrieved with [device_connectedBLEDevice \(IDT_VP3300\)](#)

```
NSString* identifier = [[IDT_VP3300 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_VP3300 sharedController] device_enableBLEDeviceSearch:identifier];
```

Scanning for BLE devices has an impact on battery life. Once you start scanning with `IDT_VP3300::device_enableBLEDeviceSearch: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 \(IDT_VP3300\)](#)

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 4

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.

4.1 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.

4.2 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.

4.3 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.

4.4 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 PA-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 data for example, for point-of-sale (POS) receipts. PCI Data Security Standard Requirement 3.3

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

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

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

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

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

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

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

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

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

3. Provide secure authentication features

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

Note: These password controls are not intended to apply to employees who only have access to one card number at a time to facilitate a single transaction. These controls are applicable for access by employees with administrative capabilities, for access to servers with cardholder data, and for access controlled by the payment application. This requirement applies to the payment application and all associated tools used to view or access cardholder data.

3.1.10 If a payment application session has been idle for more than 15 minutes, the application requires the user to re-authenticate. PCI Data Security Standard Requirement 8.5.15.

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

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

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

4. Log payment application activity

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

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

5. Develop secure payment applications

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

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

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

5.1.2 Separate development/test, and production environments

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

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

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

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

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

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

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

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

- Back-out or product de-installation procedures

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

6. Protect wireless transmissions

6.1 For payment applications using wireless technology, the wireless technology must be implemented securely. Payment applications using wireless technology must facilitate use of industry best practices (for example, IEEE 802.11i) to implement strong encryption for authentication and transmission. Controls must be in place to protect the implemented wireless network from unknown wireless access points and clients. This includes testing the end users wireless deployment on a quarterly basis to detect unauthorized access points within the system. Change wireless vendor defaults, including but not limited to default wireless encryption keys, passwords, and SSID 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.

4.5 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 5

VP3300 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.

5.1 EMV Methods

Start EMV Transaction

[emv_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback: \(IDT_VP3300\)](#)

Start CTLS Transaction (EMV or MSD)

[IDT_VP3300::ctls_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:\(\)](#)

Start Any Interface Transaction

[device_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback: \(IDT_VP3300\)](#)

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

[emv_authenticateTransaction: \(IDT_VP3300\)](#)

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: \(IDT_VP3300\)](#)), 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: \(IDT_VP3300\)](#)

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: \(IDT_VP3300\)](#)

[emv_removeTerminalData \(IDT_VP3300\)](#)

[emv_setTerminalData: \(IDT_VP3300\)](#)

[ctls_setTerminalData: \(IDT_VP3300\)](#)

[ctls_setConfigurationGroup: \(IDT_VP3300\)](#)

```
ctls_removeConfigurationGroup: (IDT_VP3300)
ctls_getConfigurationGroup:response: (IDT_VP3300)
```

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

AID Management

```
emv_retrieveApplicationData:response: (IDT_VP3300)
emv_removeApplicationData: (IDT_VP3300)
emv_setApplicationData:configData: (IDT_VP3300)
emv_retrieveAIDList: (IDT_VP3300)
ctls_retrieveApplicationData:response: (IDT_VP3300)
ctls_removeApplicationData: (IDT_VP3300)
ctls_setApplicationData: (IDT_VP3300)
ctls_retrieveAIDList: (IDT_VP3300)
```

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_VP3300)
emv_removeCAPK:index: (IDT_VP3300)
emv_setCAPK: (IDT_VP3300)
emv_retrieveCAPKList: (IDT_VP3300)
ctls_retrieveCAPK:key: (IDT_VP3300)
ctls_removeCAPK: (IDT_VP3300)
ctls_setCAPK: (IDT_VP3300)
IDT_VP3300::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_VP3300)
emv_removeCRLList (IDT_VP3300)
emv_setCRLEntries: (IDT_VP3300)
```

Methods for Certificate Revocation List management.

APDU Communication

```
device_setPassThrough: (IDT_VP3300)
icc_powerOnICC: (IDT_VP3300)
icc_powerOffICC: (IDT_VP3300)
```

```
icc_exchangeAPDU:response: (IDT_VP3300)
```

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

5.2 MSR/CTLS MSD

Request Swipe or Tap

[msr_startMSRSwipe \(IDT_VP3300\)](#)

[ctls_startTransaction \(IDT_VP3300\)](#)

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 [ID←TEMVData](#) to `deviceDelegate::emvTransactionData:()`

Cancel Swipe

[msr_cancelMSRSwipe \(IDT_VP3300\)](#)

[ctls_cancelTransaction \(IDT_VP3300\)](#)

Both methods perform identical operation. Cancels the Swipe/Tap request.

Chapter 6

EMV LCD Display Callback

During an EMV transaction, without a built-in LCD display on the VP3300, LCD Display messages will be returned as an EMV Callback.

In the available protocols for the [IDT_VP3300](#) class, there is a protocol as follows:

```
// Objective-C
- (void) lcdDisplay:(int)mode lines:(NSArray*)lines;

// Swift
func lcdDisplay(_ mode: Int32, lines: [Any]!)
```

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

To evaluate what kind of LCD message, you interpret the mode as follows:

- 1- LCD_DISPLAY_MODE_MENU: Menu selection, response required with selected menu index #, or 0 to cancel
- 2- LCD_DISPLAY_MODE_PROMPT: Message Prompt, response required 'E' for Enter/Accept, or 'C' for cancel
- 3- LCD_DISPLAY_MODE_MESSAGE: Display Message, no response required
- 8 - LCD_DISPLAY_MODE_LANGUAGE_SELECT: Language selection, response required with selected language index #
- 16 - LCD_DISPLAY_MODE_CLEAR_SCREEN: Request to clear LCD screen of information

If the mode is LCD_DISPLAY_MODE_MESSAGE or LCD_DISPLAY_MODE_CLEAR_SCREEN, these do not pause the EMV transaction. These two modes are for displaying a message (no response required), or for clearing the screen.

If the mode is LCD_DISPLAY_MODE_MENU, LCD_DISPLAY_MODE_PROMPT, or LCD_DISPLAY_MODE_LANGUAGE_SELECT, the provided message must be displayed, and then the EMV transaction pauses until a response is sent to [emv_callbackResponseLCD:selection: \(IDT_VP3300\)](#).

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

Chapter 7

EMV PIN Callback

The VP8800 has the ability to interface with external PED (PIN Entry Device). For it to be valid for live transactions, the PED must be paired with the VP8800 using shared keys.

When the kernel encounters a PIN CVM, it will send a request for PIN to the protocol. When properly paired, the kernel will pass the correct key (DUKPT or Session), and information what kind of PIN CVM it is, either Online DUKPT, Online Session, or Offline (DUKPT). It is then expected the PED will securely collect the PIN, and pass it back to the kernel in its encrypted form, including the KSN if DUKPT.

If the PED is not paired with the VP3300, it is possible to collect a plaintext offline PIN and pass it back to the kernel. THIS MUST BE USED FOR DEMO PURPOSES ONLY. Collecting, transmitting and processing unprotected PIN data violates PCI and introduces considerable liability to the merchant.

To implement, first make the class a delegate of [IDT_VP3300](#) ([IDT_VP3300_Delegate](#)). Then include the delegate protocols to capture PIN request.

```
// Objective-C
- (void) pinRequest:(EMV_PIN_MODE_Types)mode key:(NSData*)key PAN:(NSData*)PAN startTO:(int)startTO
  intervalTO:(int)intervalTO language:(NSString*)language;

// Swift
func pinRequest(_ mode: EMV_PIN_MODE_Types, key: Data!, pan PAN: Data!, startTO: Int32, intervalTO: Int32,
  language: String!)
```

When `pinRequest` is called, this indicates the kernel encountered a PIN CVM and would like an external PIN supplied to continue the transaction. If the PED is paired, the delegate will receive all relevant key information. If the PED is not paired, the key information will not be provided.

Once the PIN is collected, it must be sent back to kernel. If the PED is paired, the PIN data will be encrypted, and the KSN must be provided if DUKPT was used. If the PED is not paired, there is no KSN, and the PIN data would be plaintext (example "1234" = `NSData*` object 0x31323334). The mode passed would be the same mode that was received in the delegate, unless a `PIN_CANCEL` is desired, then the mode would be `EMV_PIN_MODE_CANCEL`, and no KSN or PIN data is provided.

```
// Objective-C
-(RETURN_CODE) emv_callbackResponsePIN:(EMV_PIN_MODE_Types)mode KSN:(NSData*)KSN PIN:(NSData*)PIN;

// Swift
func emv_callbackResponsePIN(_ mode: EMV_PIN_MODE_Types, ksn KSN: Data, pin PIN: Data) -> RETURN_CODE
```

To enable PIN in the terminal, tag 9F33 must be configured accordingly. This means a kernel configuration

IDTech iOS SDK Guide for VP3300 #80149513-001

Chapter 8

Sending Direct Commands

The main purpose of IDTech.framework for VP3300 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 VP3300 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 VP3300 can be accomplished by a Protocol 2 command. The IDTech.framework uses the following the command to send Protocol 2 commands to VP3300:

[device_sendIDGCommand:subCommand:data:response: \(IDT_VP3300\)](#)

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

Chapter 9

Core Implementation VP3300: Objective-C

IDTech Framework includes class libraries to interface with the VP3300. This guide assume a fair understanding of Xcode 5.0+ and general Apple iOS programming knowledge.

9.1 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_VP3300 objects](#)
- [Sample Project Tutorial](#)

9.2 Import the necessary framework/libraries

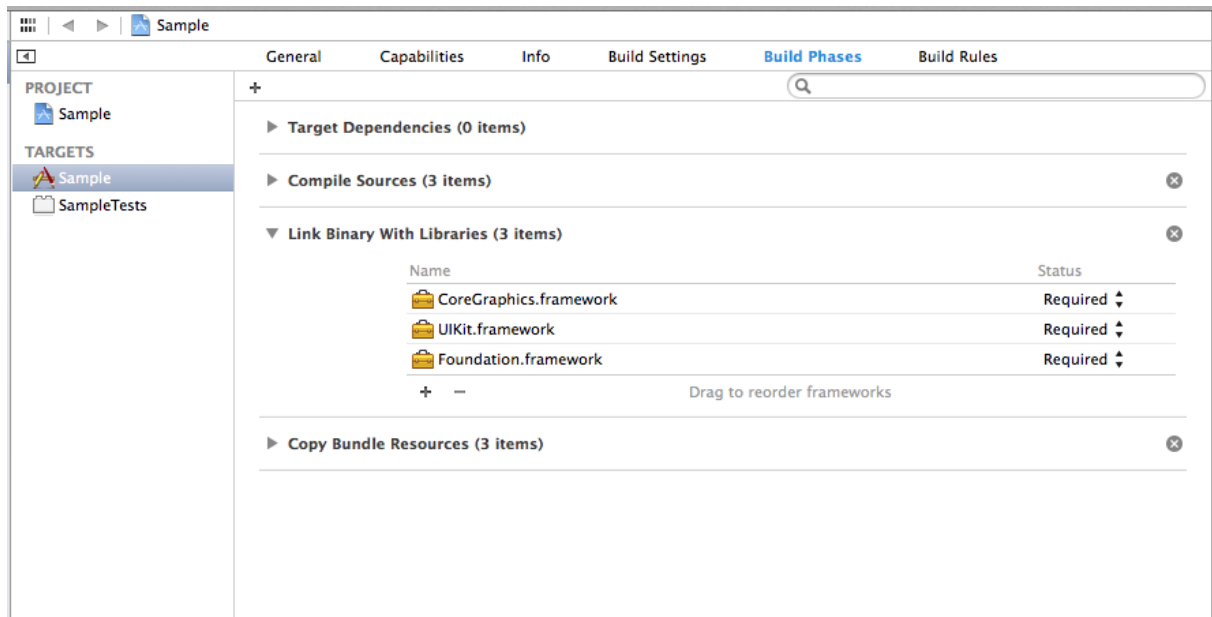
Communicating with IDTech Devices requires the following framework/libraries to be imported into the project:

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

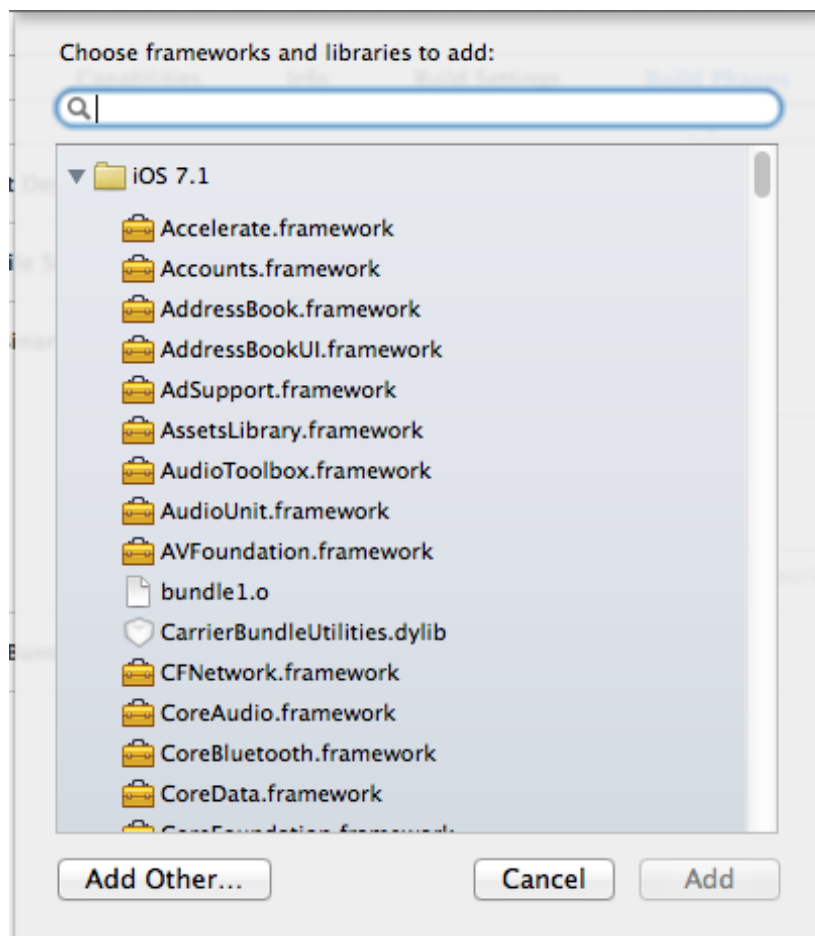
Also, import the following resource bundle:

- IDTech.bundle

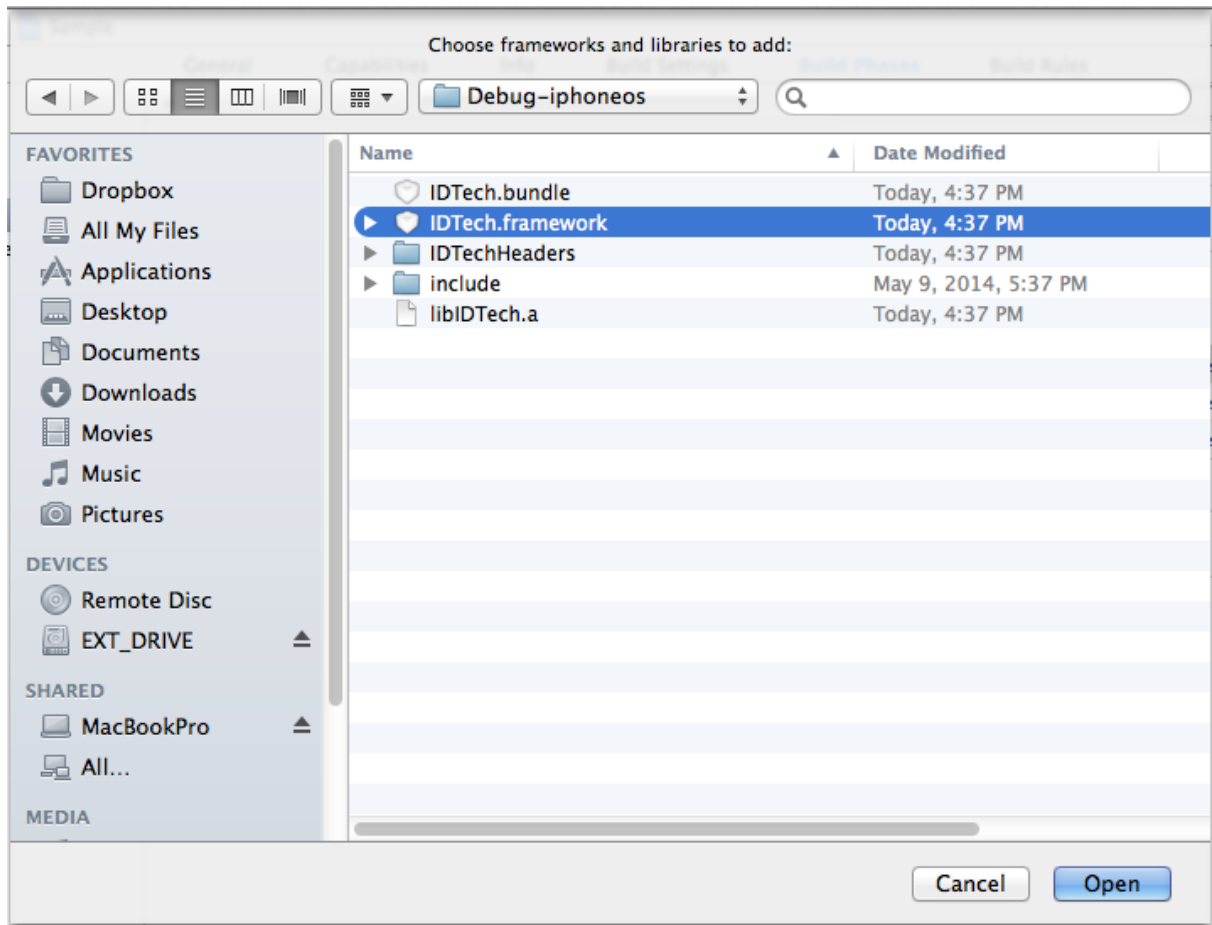
Under Build Phases, select Link Binary With Libraries and click the Add (+) button



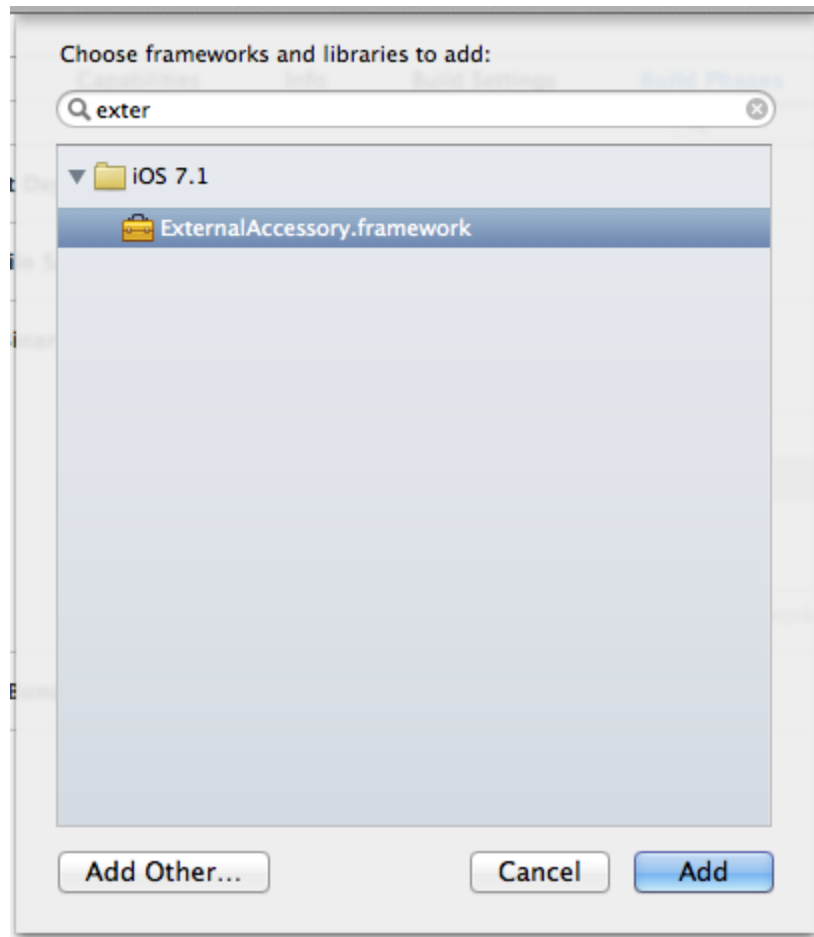
On the Choose Frameworks screen, click "Add Other" in the lower left



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



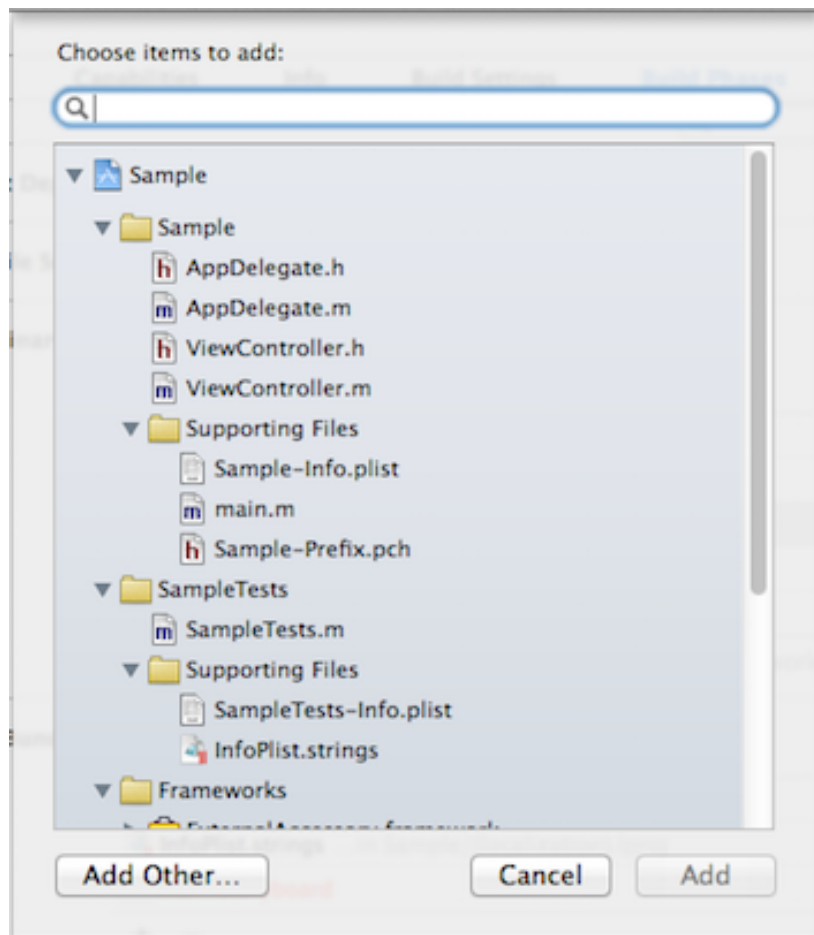
Link the ExternalAccessory framework. On the Choose Frameworks screen, type "exter" into the search bar, select ExternalAccessory.framework and click "Open"

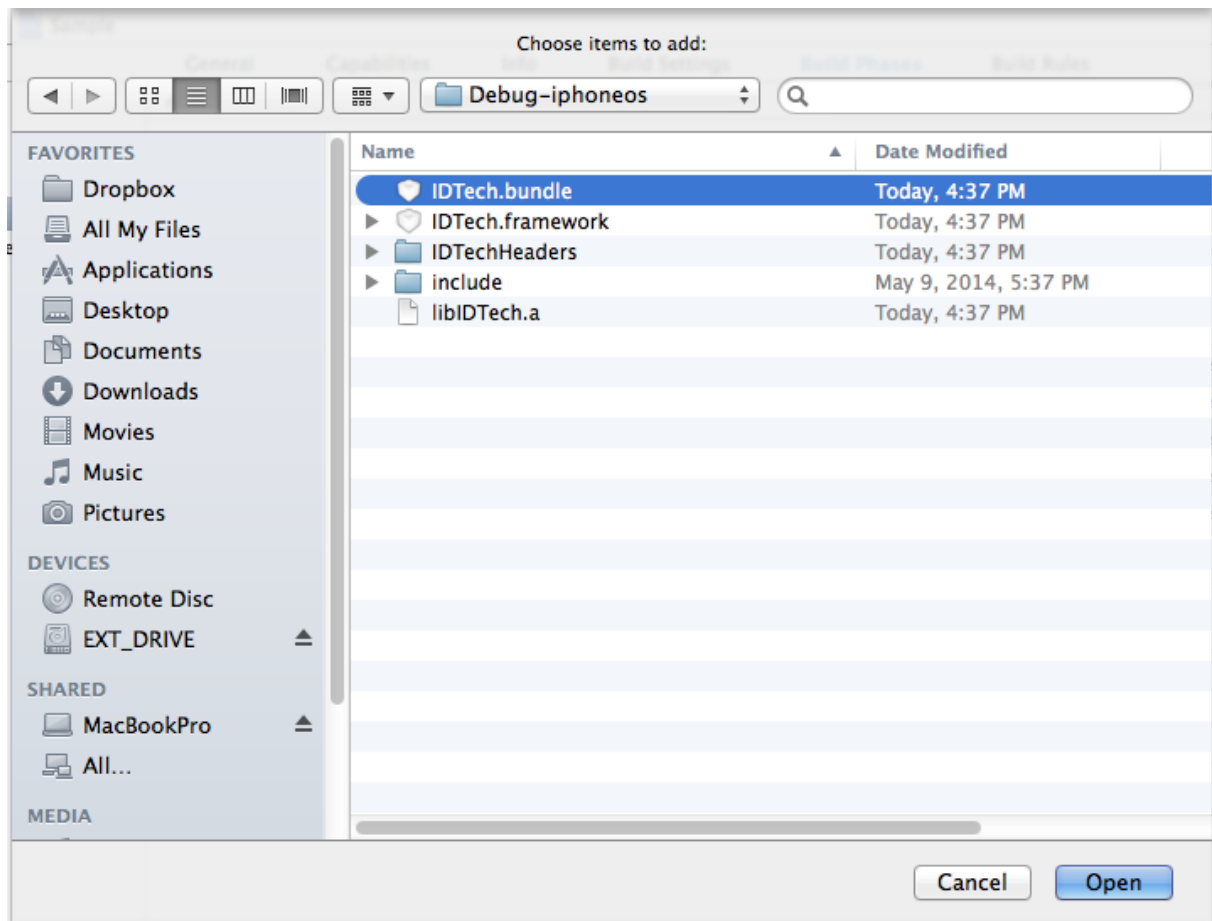


Repeat process for MediaPlayer.framework, AVFoundation.framework, AudioToolbox.framework, and CFNetwork.framework.

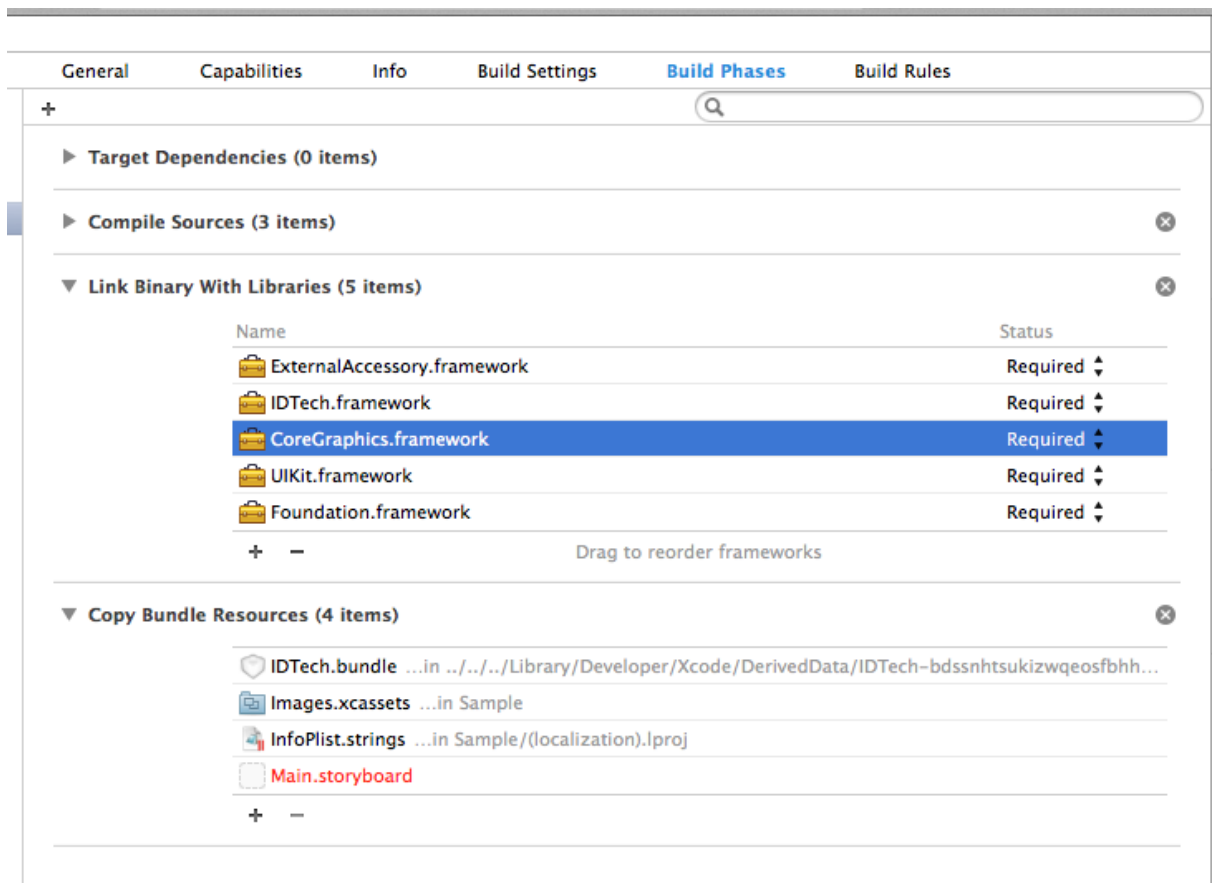
Link another library. Under Copy Bundle, click the Add (+) button, click "Add Other", navigate to and select the IDTech.bundle file and click "Open"







The Build Phases should now include the required frameworks/libraries for the VP3300



9.3 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>
```

9.4 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_VP3300_Delegate>
```

9.5 Implement any/all of the optional delegate protocols used to receive data from IDT_VP3300_Delegate:

```
- (void) deviceConnected;
- (void) deviceDisconnected;
- (void) plugStatusChange: (BOOL) deviceInserted;
- (void) dataInOutMonitor: (NSData*) data incoming: (BOOL) isIncoming;
```

```

- (void) swipeMSRData:(IDTMSRData*) cardData;

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

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

- (void) pinRequest:(EMV_PIN_MODE_Types) mode key:(NSData*) key PAN:(NSData*) PAN startTO:(int) startTO
    intervalTO:(int) intervalTO language:(NSString*) language;

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

```

9.6 Call the Singleton instance of the IDT_VP3300 framework object:

A Singleton instance has been established in the [IDT_VP3300](#) 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 VP3300 with Bluetooth Low Energy capabilities, you need to execute [device_enableBLEDeviceSearch: \(IDT_VP3300\) identifier:\(\)](#).

```

- (void) viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [[IDT_VP3300 sharedController] setDelegate:self];
    [[IDT_VP3300 sharedController] device_enableBLEDeviceSearch:IDT_DEVICE_VP3300_IOS identifier:
        nil];
}

```

9.7 Sample Project Tutorial

Using Xcode 6.4, we will create a sample project that will interface with the VP3300 and will perform the following activities:

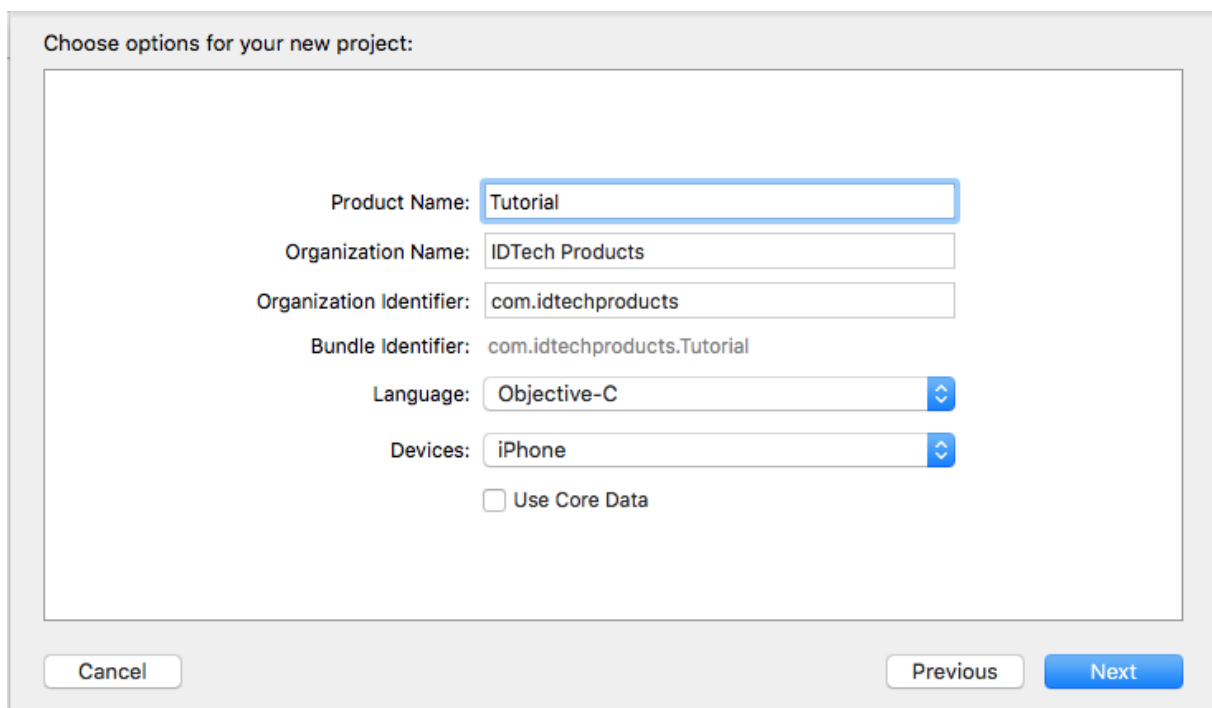
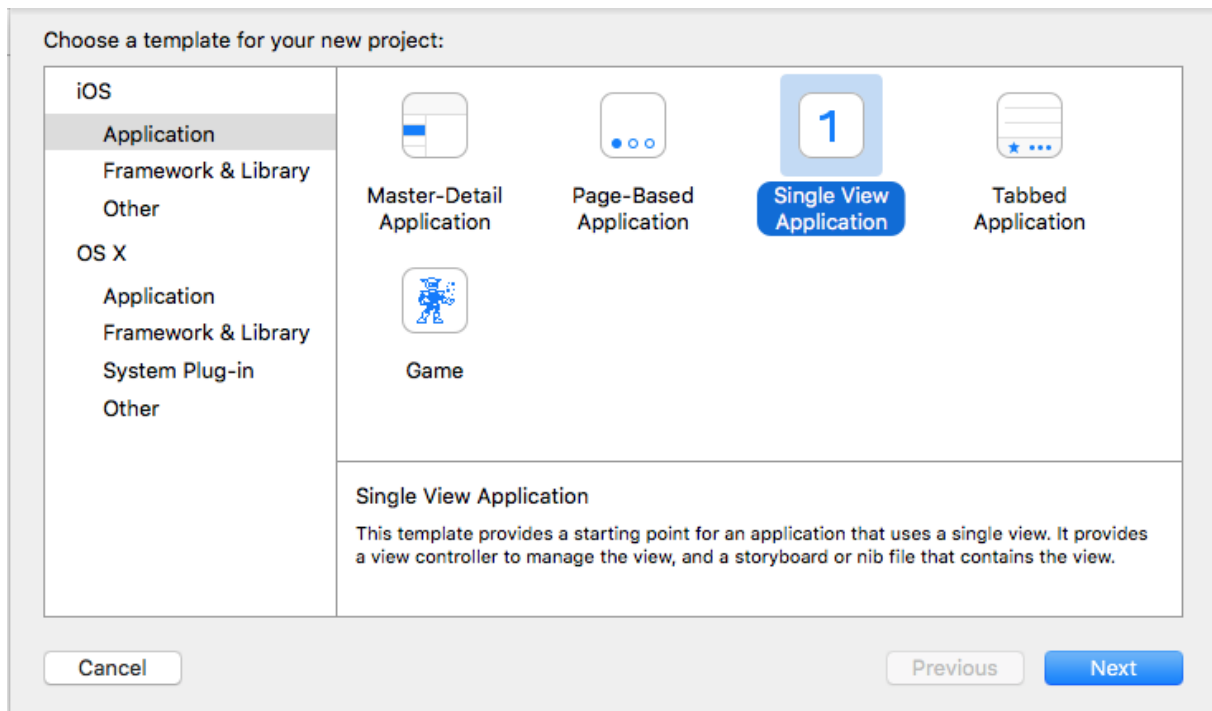
- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for CTLS/MSR/ICC (tap/swipe/insert)
- Enable Bluetooth Scanning
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

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

9.7.1 Step 1: Create New Project

Create a new Single View Application in Xcode



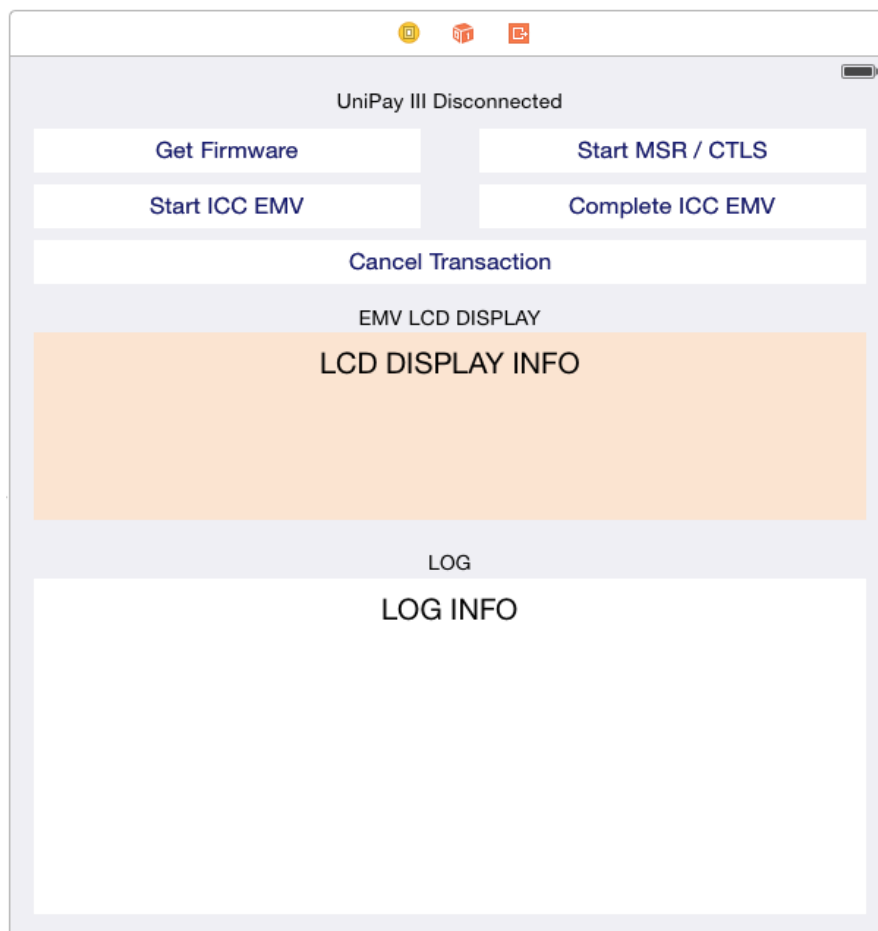
9.7.2 Step 2: Import Frameworks

[Import the necessary framework/libraries](#)

9.7.3 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 VP3300 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)



9.7.4 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_VP3300_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(n nonatomic, strong) UITextView *lcdTextView;
@property(n nonatomic, strong) UITextView *logTextView;
@property(n 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="VP3300 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-6l-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" lineBreakMode="tailTruncation"
                                baselineAdjustment="alignBaselines" adjustsFontSizeToFit="NO"
                                translatesAutoresizingMaskIntoConstraints="NO" id="oIl-9l-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="0.9882352948000005" 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-All"/>
        </constraints>
        <fontDescription key="fontDescription" type="system" pointSize="14"/>
        <state key="normal" title="Get Firmware">
            <color key="titleColor" red="0.06800394908000001" 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="TZ0-wC-gU1"/>
        </constraints>
        <fontDescription key="fontDescription" type="system" pointSize="14"/>
        <state key="normal" title="Scan for VP3300 BLE Device">
            <color key="titleColor" red="0.06800394908000001" 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="0.06800394908000001" 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="cHl-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-9l-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-qg-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-9l-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-9l-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-9l-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-9l-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-9l-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>

```

9.7.5 Step 5: Configure Method File

In the header file, perform the following:

- set delegate and initialize `IDT_VP3300` singleton object in the `viewDidLoad` method.

Reference: [Call the Singleton instance of the IDT_VP3300 framework object](#)

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [[IDT_VP3300 sharedInstance] setDelegate:self];
}

```

- Implement protocol delegate `IDT_VP3300Delegate::deviceDisconnected()` and `IDT_VP3300Delegate::deviceConnected()` to monitor connect/disconnect events and modify our connection label upon change. Reference: [Implement optional delegate protocols](#)

```
-(void) deviceConnected{
    [connectedLabel setText:@"VP3300 CONNECTED"];
}
-(void) deviceDisconnected{
    [connectedLabel setText:@"VP3300 DISCONNECTED"];
}
```

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

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

-(void) swipeMSRData:(IDTMSRData*)cardData{
    NSLog(@"--MSR event Received, Type: %d, data: %s", 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: %s", @"ISO/ABA"]]
                    break;
                case CAPTURE_ENCODE_TYPE_AAMVA:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %s", @"AA/MVA"]]
                    break;
                case CAPTURE_ENCODE_TYPE_Other:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %s", @"Other"]]
                    break;
                case CAPTURE_ENCODE_TYPE_Raw:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %s", @"Raw"]]
                    break;
                case CAPTURE_ENCODE_TYPE_JIS_I:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %s", @"CAPTURE_ENCODE_TYPE_JIS_I"]]];
                    break;
                case CAPTURE_ENCODE_TYPE_JIS_II:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %s", @"CAPTURE_ENCODE_TYPE_JIS_II"]]];
                    break;
                default:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encode Type: %s", @"UNKNOWN"]]
                    break;
            }
            switch (cardData.captureEncryptType) {
                case CAPTURE_ENCRYPT_TYPE_AES:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %s", @"AES"]]
                    break;
                case CAPTURE_ENCRYPT_TYPE_TDES:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %s", @"TDES"]]
                    break;
                case CAPTURE_ENCRYPT_TYPE_NO_ENCRYPTION:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %s", @"NONE"]]
                    break;
                default:
                    [self appendMessageToResults:[NSString stringWithFormat:@"Encrypt Type: %s", @"UNKNOWN"]]
                    break;
            }
            [self appendMessageToResults:[NSString stringWithFormat:@"Full card data: %s", cardData.cardData]];
            [self appendMessageToResults:[NSString stringWithFormat:@"Track 1: %s", cardData.track1]];
            [self appendMessageToResults:[NSString stringWithFormat:@"Track 2: %s", cardData.track2]];
            [self appendMessageToResults:[NSString stringWithFormat:@"Track 3: %s", 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:@"Unencrypted 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 `plugStatusChange:()` to automatically attempt connection attempt.

```

- (void) plugStatusChange: (BOOL) deviceInserted{
    if (deviceInserted) {
        [self appendMessageToResults: @"device Attached. Attempting connection"];
        [[IDT_VP3300 sharedController] device_connectToAudioReader];
    }
    else{
        [self appendMessageToResults: @"device removed."];
    }
}

```

- 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:@"Unencrypted 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_VP3300 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_VP3300 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_VP3300 sharedController] device_enableBLEDeviceSearch:nil];
}

-(IBAction) startTransaction:(id)sender{
    logTextView.text = @"";
    RETURN_CODE rt = [[IDT_VP3300 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_VP3300 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_VP3300 sharedController] ctls_cancelTransaction];
    if (RETURN_CODE_DO_SUCCESS == rt){
        [self appendMessageToResults:@"CancelMSR/CTLS: OK."];
    }
}

```

Chapter 10

Core Implementation VP3300: Swift

IDTech Framework includes class libraries to interface with the VP3300. This guide assume a fair understanding of Xcode 7.3+ and general Apple iOS programming knowledge.

10.1 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_VP3300 Object](#)
- [Sample Project Tutorial](#)

10.2 Import the Necessary Framework/Libraries

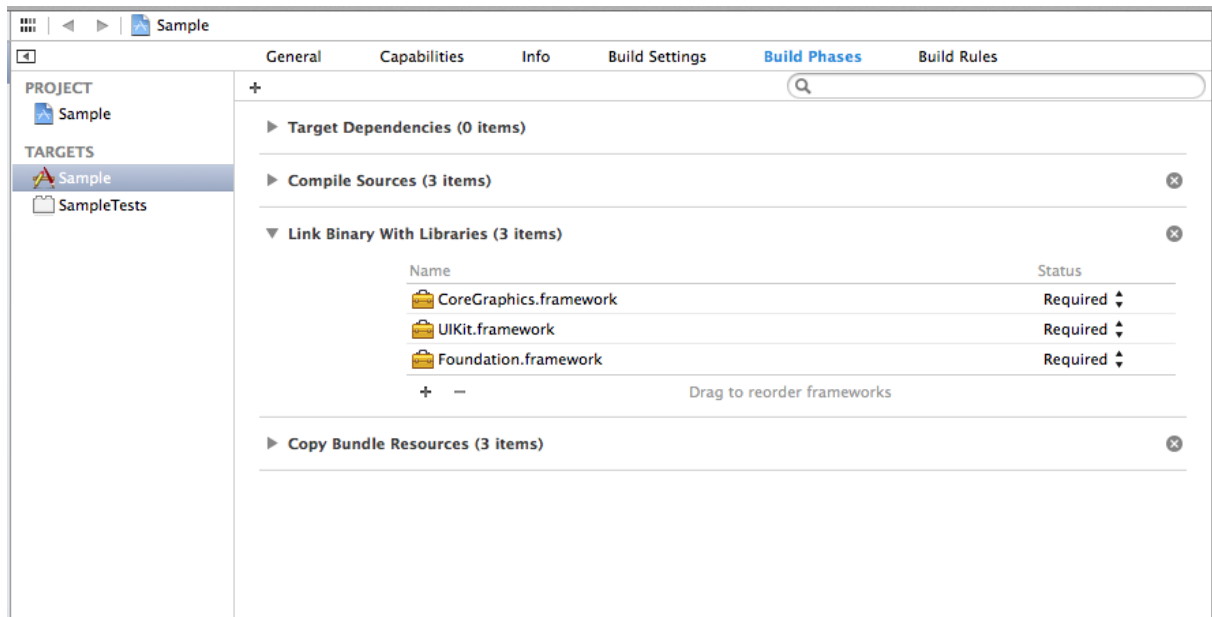
Communicating with IDTech Devices requires the following framework/libraries to be imported into the project:

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

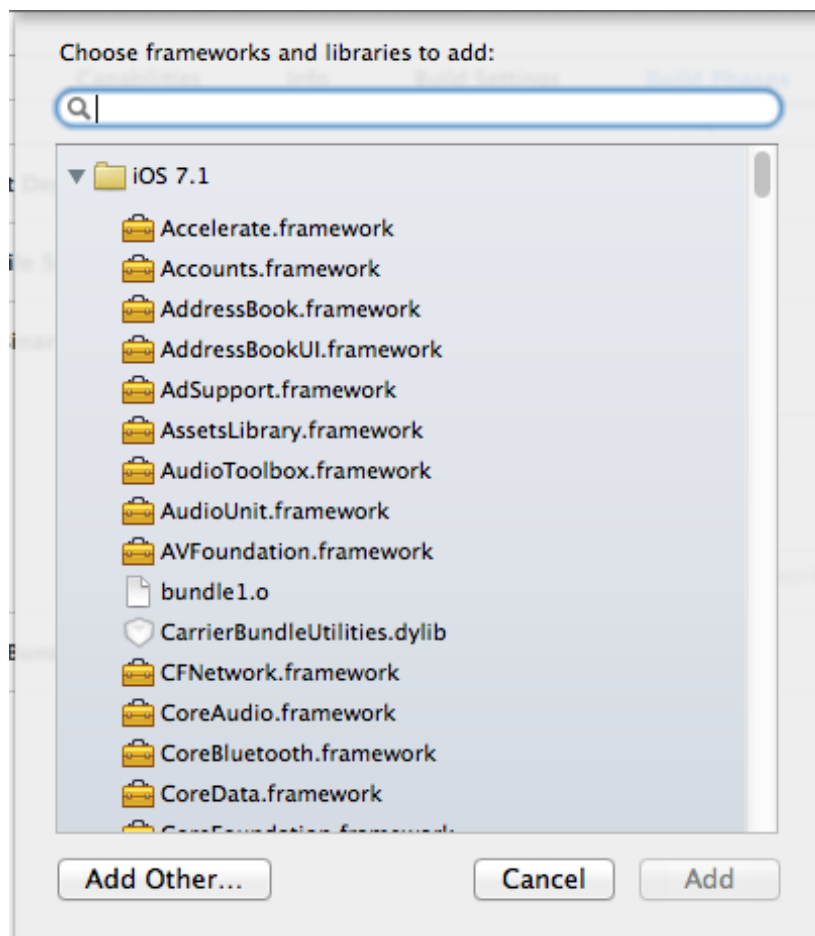
Also, import the following resource bundle:

- IDTech.bundle

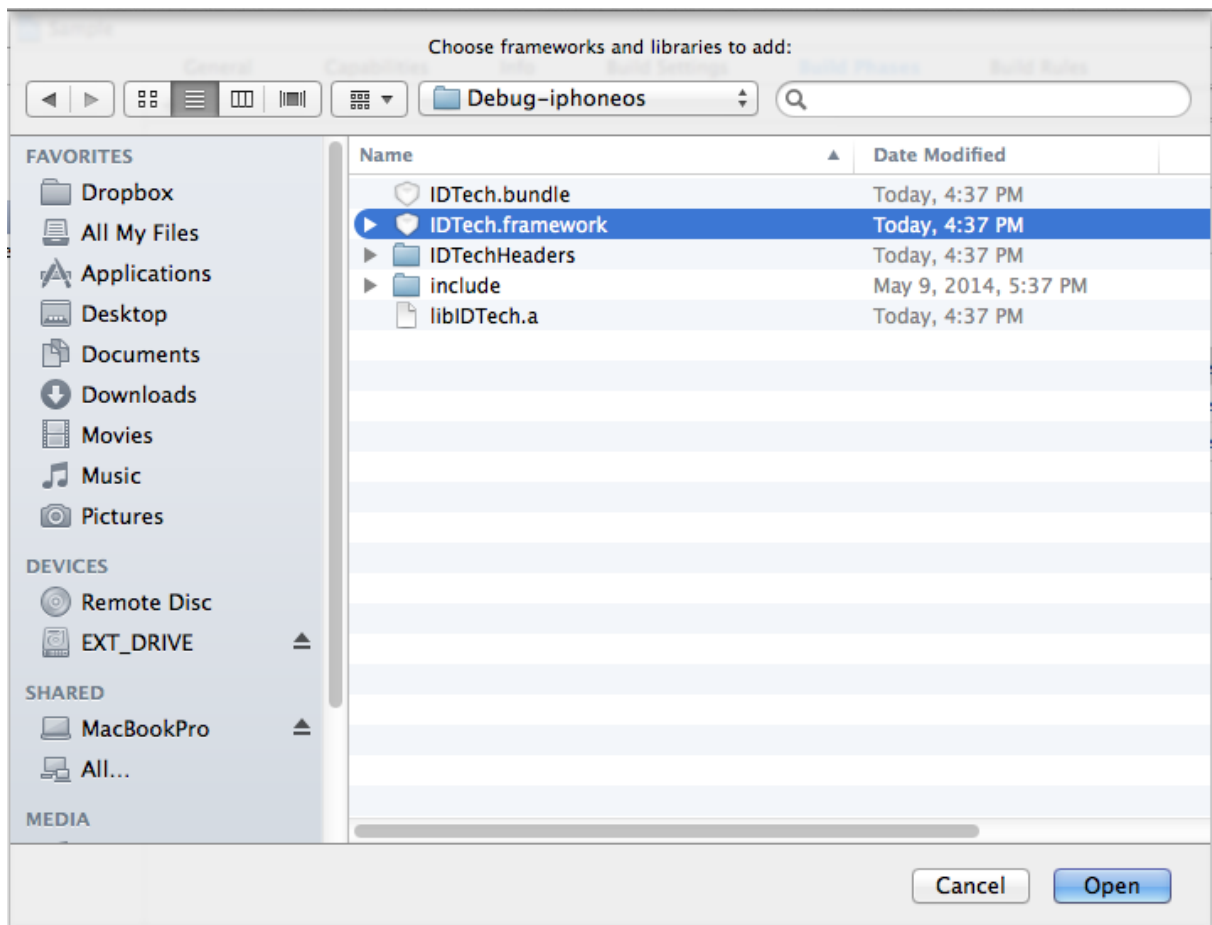
Under Build Phases, select Link Binary With Libraries and click the Add (+) button



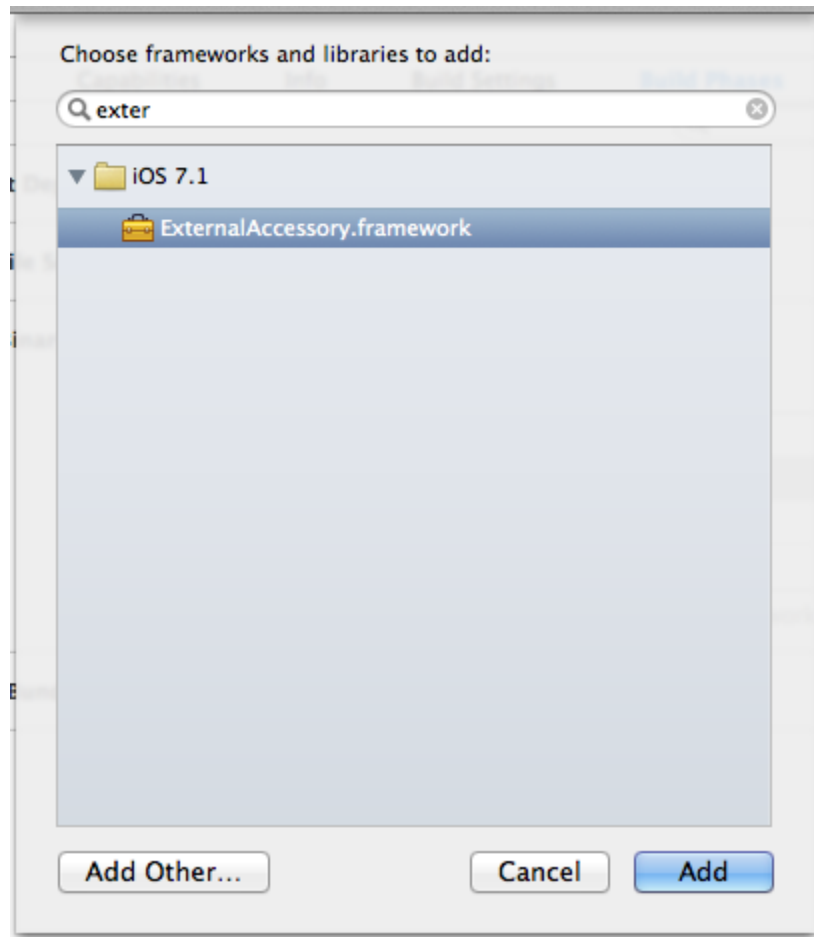
On the Choose Frameworks screen, click "Add Other" in the lower left



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 ExternalAccessory.framework and click "Add"

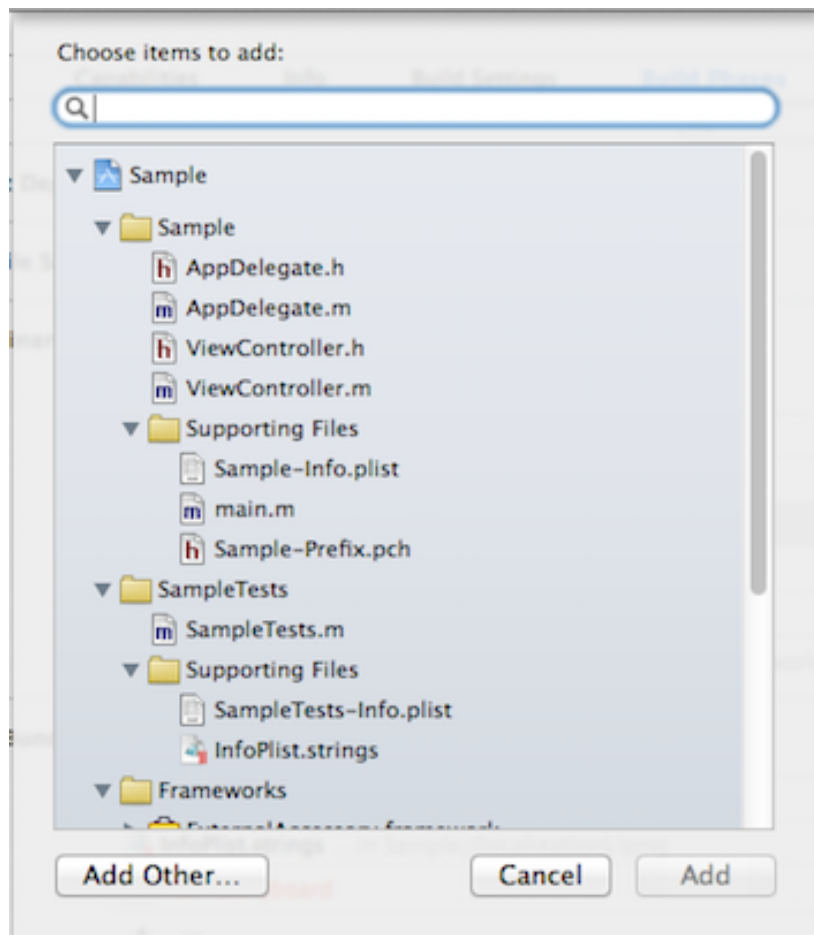


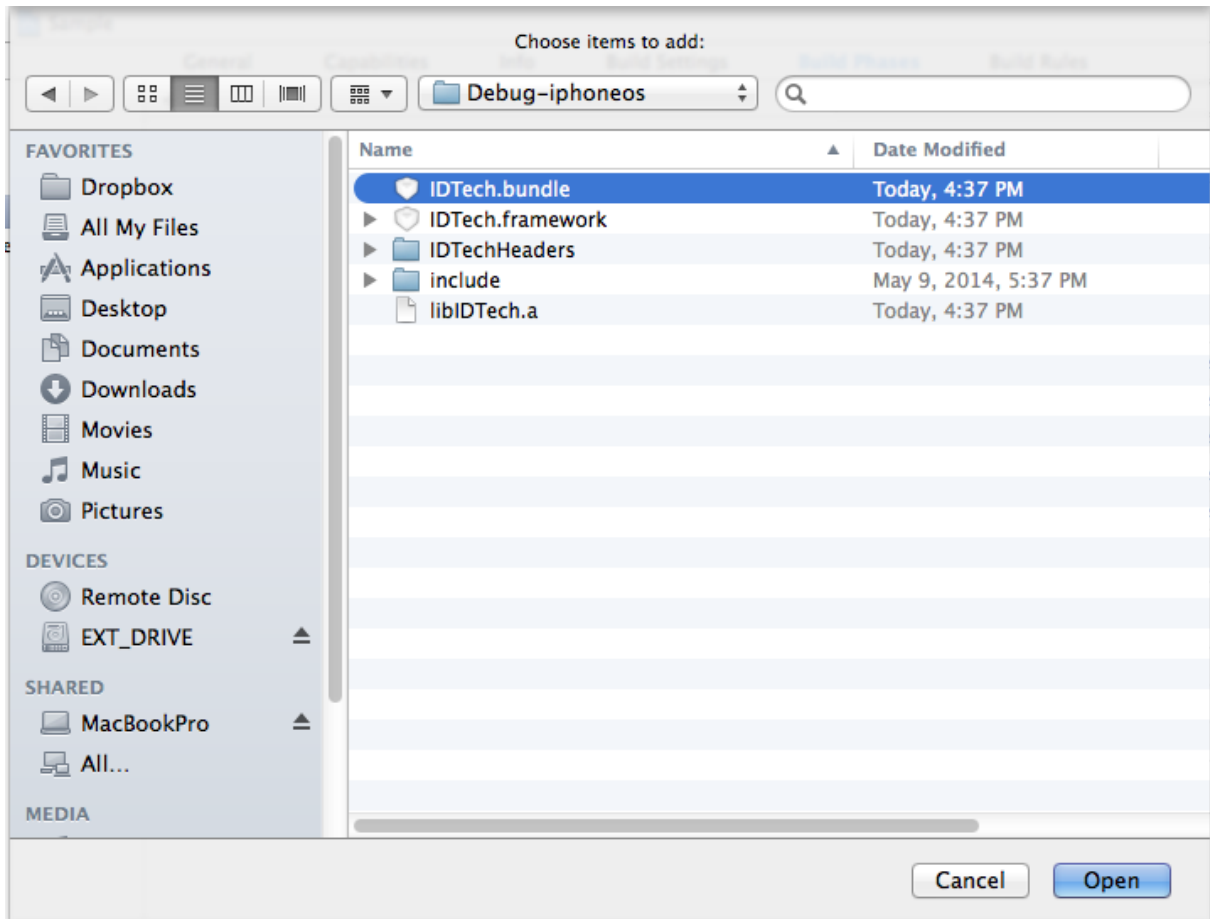
Repeat process for MediaPlayer.framework, AVFoundation.framework, AudioToolbox.framework, and CFNetwork.framework

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

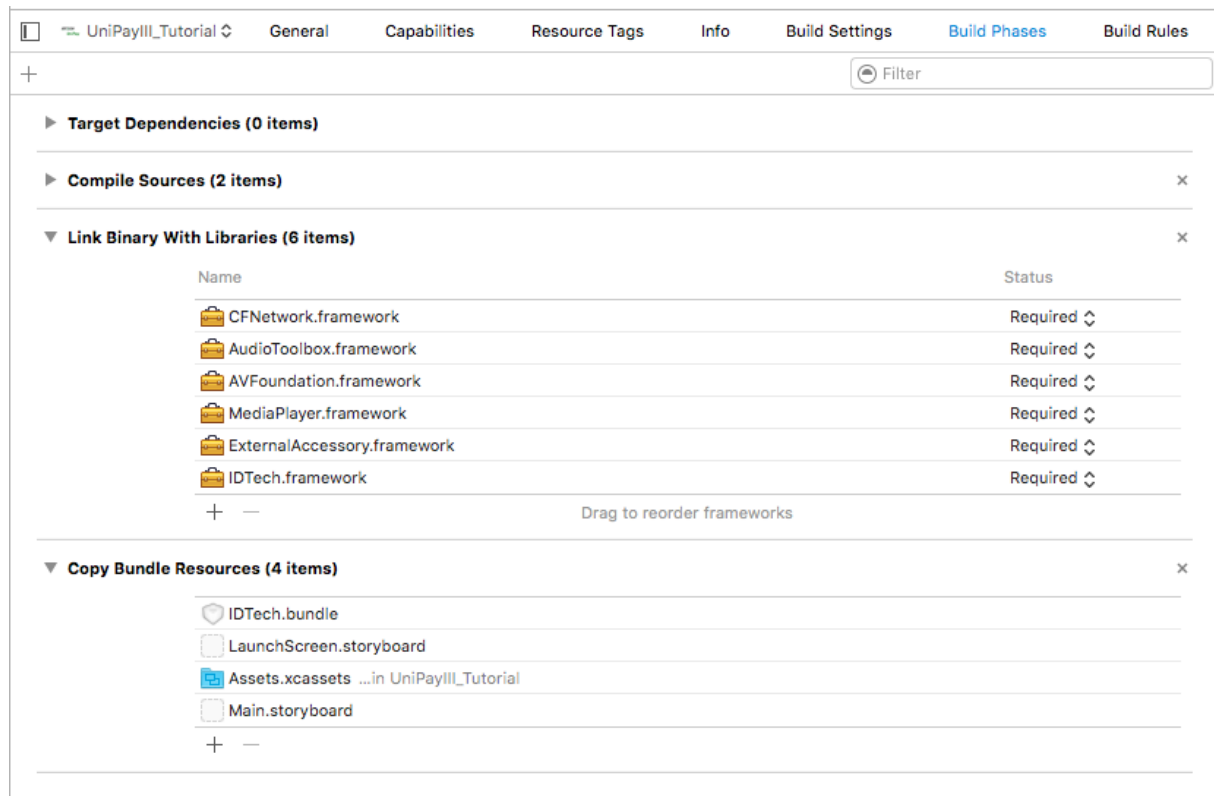
▼ Copy Bundle Resources (3 items)

- ☐ LaunchScreen.storyboard
 - ☒ Assets.xcassets ...in UniPayIII_Tutorial
 - ☐ Main.storyboard
- + —





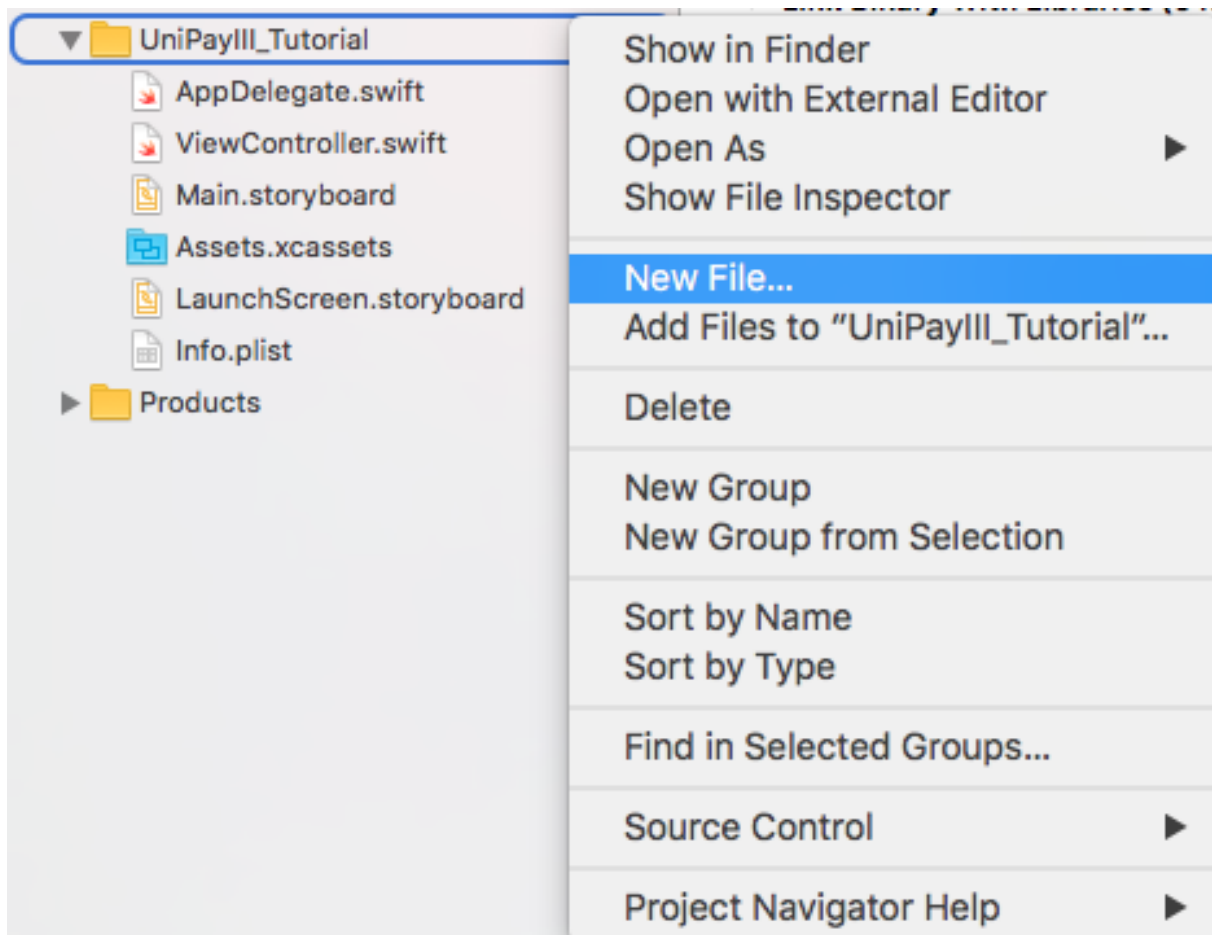
The Build Phases should now include the required frameworks/libraries for the VP3300



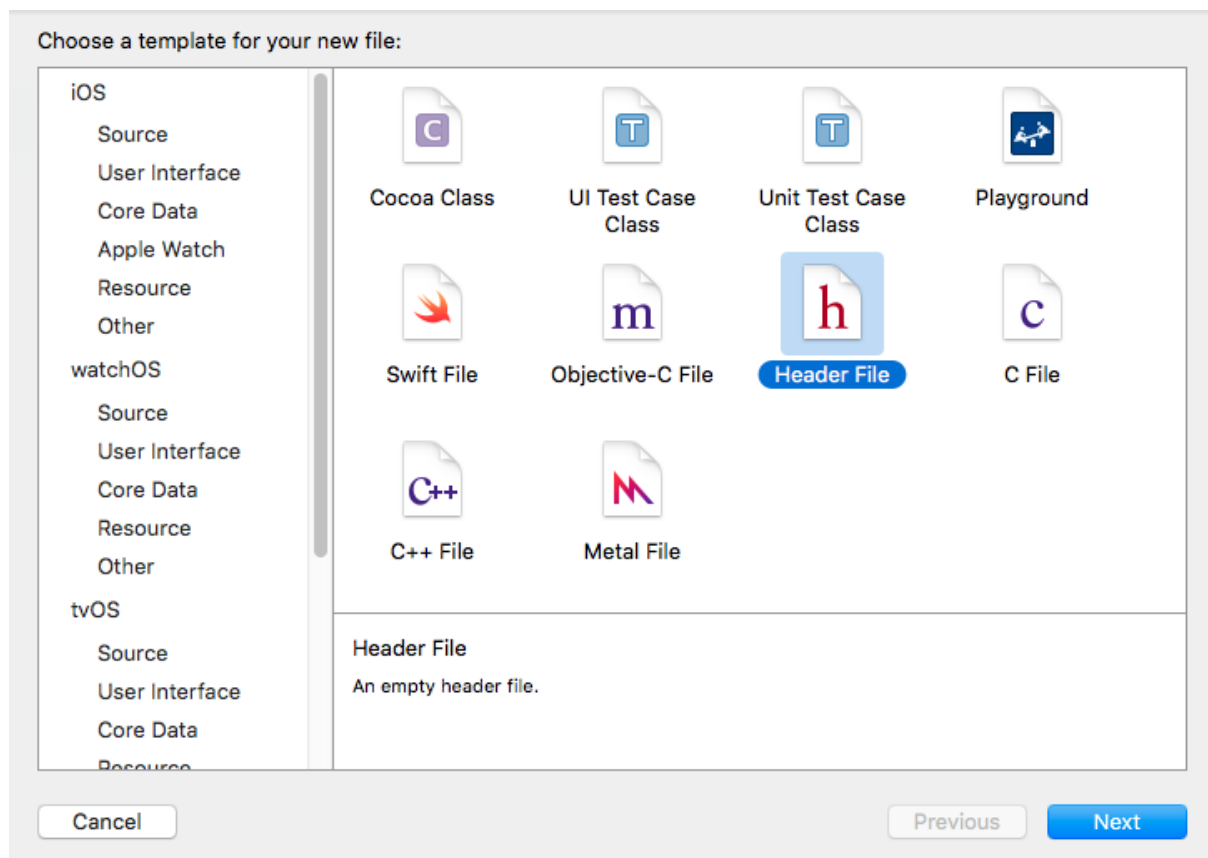
10.3 Create a Bridging Header File

Using the VP3300 SDK with Swift requires a bridging header to allow the application to have a mixed-language codebase.

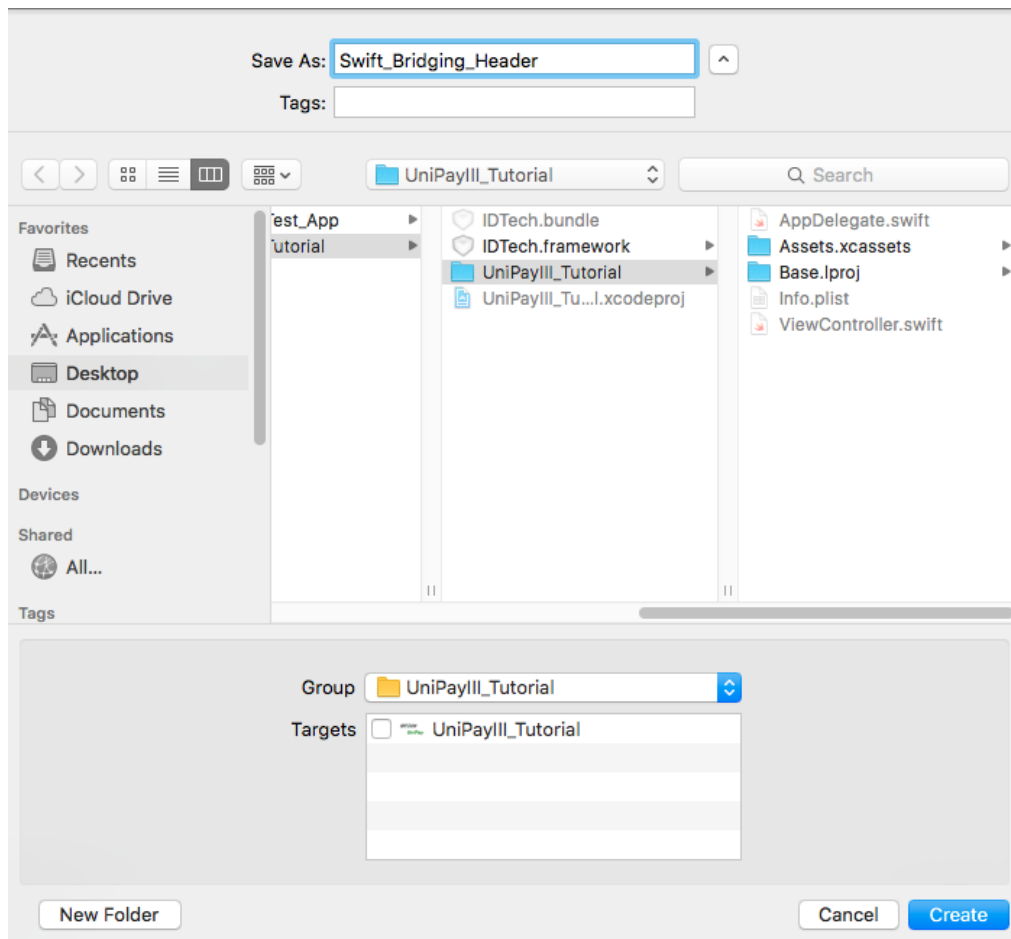
First, you will need to create a header file. Right click on your project directory and click "New File...".



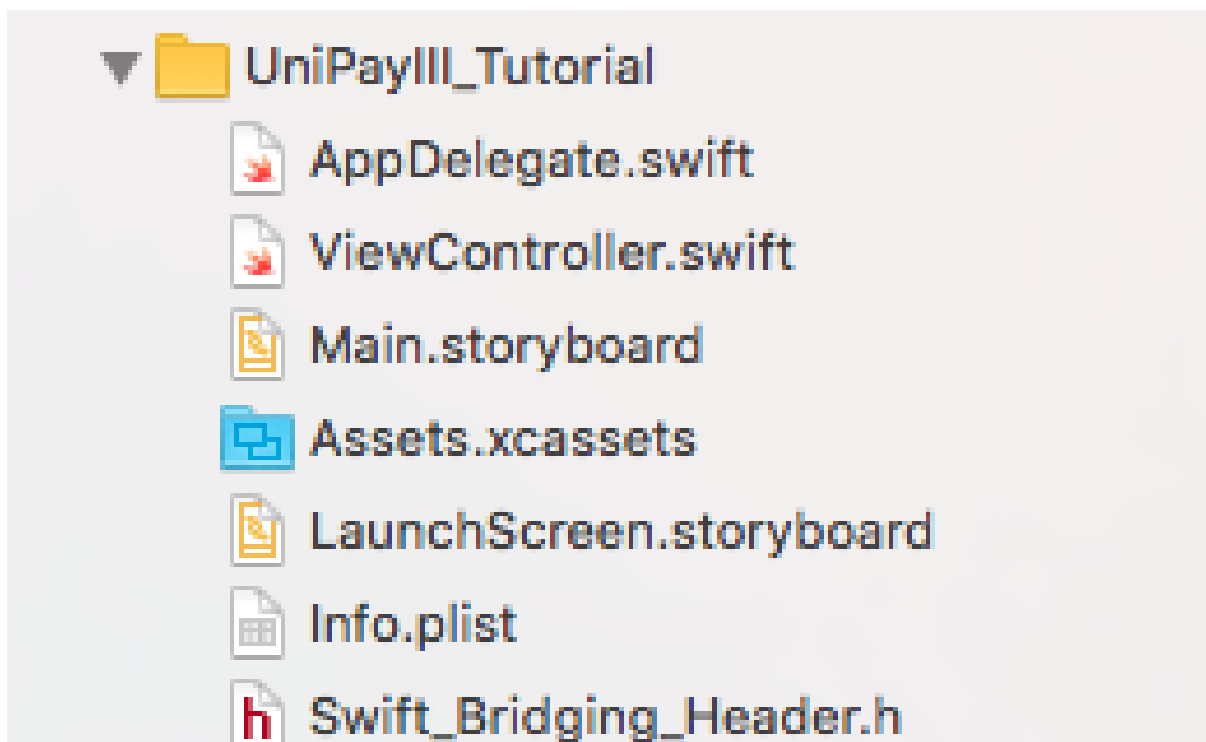
On the file creation screen, select Header File and click "Next"



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



The bridging header file is now created

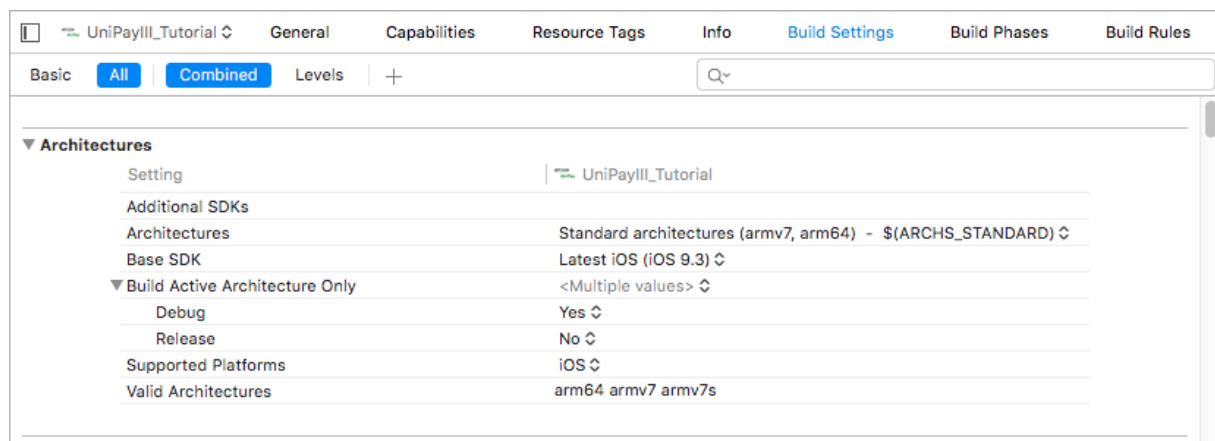


The header file should look similar to the following:

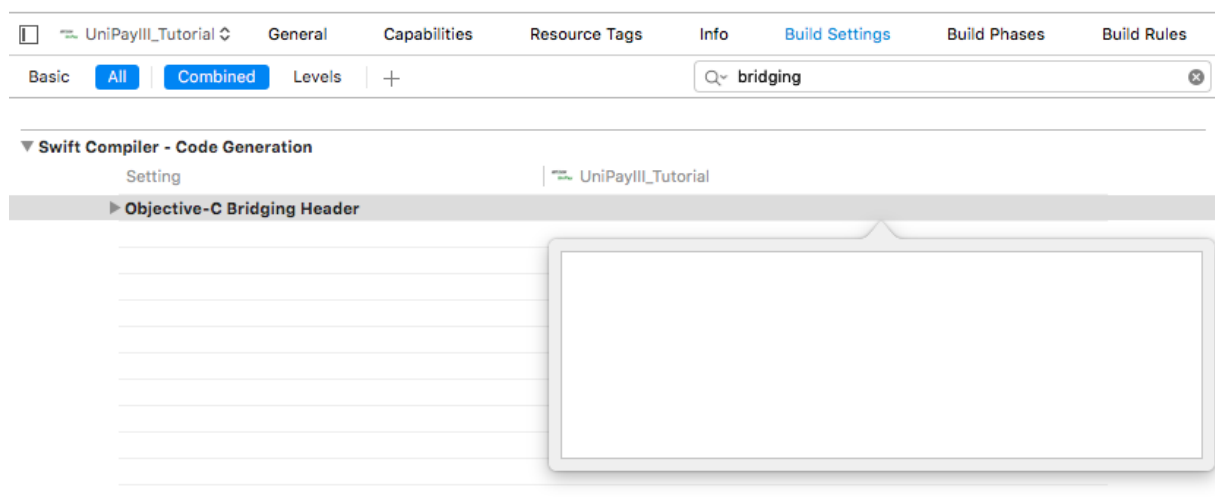
```
#ifndef Swift_Bridging_Header_h
#define Swift_Bridging_Header_h

#endif
```

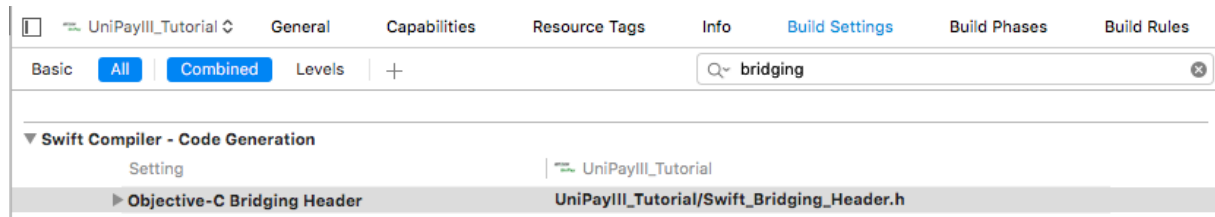
Next, you will need to link your header file to the bridging header setting. Navigate to Build Settings within Xcode



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



10.4 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
```

10.5 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_VP3300_Delegate
```

10.6 Implement Optional Delegate Protocols

Implement any/all of the optional delegate protocols used to receive data from [IDT_VP3300_Delegate](#):

```
func deviceConnected()
func deviceDisconnected()
func plugStatusChange(deviceInserted: Bool)
func dataInOutMonitor(data: NSData!, incoming isIncoming: Bool)
func swipeMSRData(cardData: IDTMSRData!)
```

```
func deviceMessage(message: NSString!)

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

func pinRequest(_ mode: EMV_PIN_MODE_Types, key: Data!, pan PAN: Data!, startTO: Int32, intervalTO: Int32,
    language: String!)

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

10.7 Allocate/Initialize IDT_VP3300 Object

A Singleton instance has been established in the [IDT_VP3300](#) 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_VP3300.sharedController().delegate = self
}
```

10.8 Sample Project Tutorial

Using Xcode 7.3.1, we will create a sample project that will interface with the VP3300 and will perform the following activities:

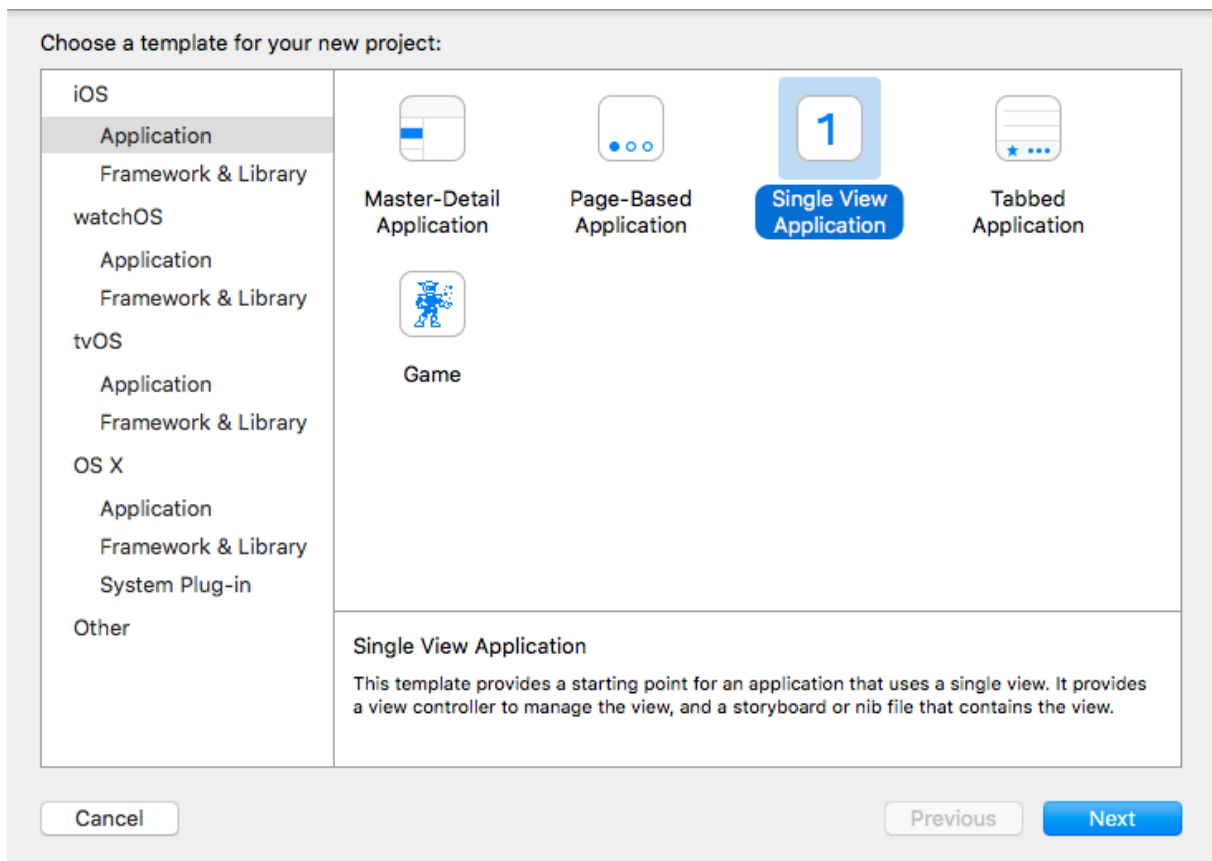
- Auto-Connect and display connection status
- Get Device Firmware
- Start/Stop Transaction Request for CTLS/MSR (tap/swipe)
- Start/Complete/Cancel EMV Transaction
- Show LCD Display for EMV transaction
- Automatically select first AID or first Language if prompted

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

10.8.1 Step 1: Create New Project

Create a new Single View Application in Xcode



Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

☐ Use Core Data

☐ Include Unit Tests

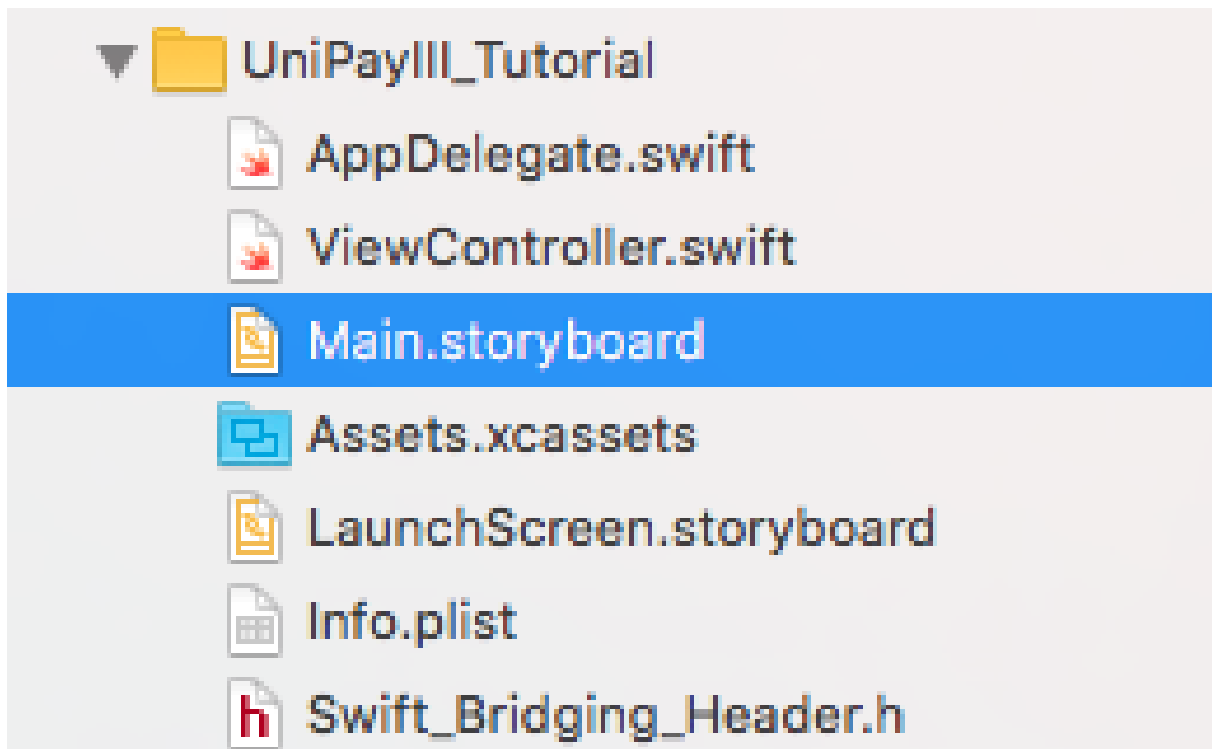
☐ Include UI Tests

10.8.2 Step 2: Import Frameworks

[Import the Necessary Framework/Libraries](#)

10.8.3 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 VP3300 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)

10.8.4 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_VP3300_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="VP3300_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="VP3300 Tutorial" id="gtg-5k-9F0"/>
                                </items>
                            </navigationBar>
                            <label opaque="NO" userInteractionEnabled="NO" contentMode="left"
                                horizontalHuggingPriority="251" verticalHuggingPriority="251" text="Disconnected" textAlignment="center" lineBreakMode="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="middleTruncation"
                                            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"/>
                                            </constraints>
                                        </button>
                                    </subviews>
                                </view>
                            </subviews>
                        </view>
                    </viewController>
                </objects>
            </scene>
        </scenes>
    </document>
```

```

                                <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="mid
translatesAutoresizingMaskIntoConstraints="NO" id="TGp-el-4b0">
                                    <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="9ml-BE-9sH"/>
                                    </connections>
                                    </button>
                                    <button opaque="NO" contentMode="scaleToFill"
contentHorizontalAlignment="center" contentVerticalAlignment="center" buttonType="roundedRect" lineBreakMode="mid
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="mid
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="mid
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-el-4b0" 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=
"ZF1-2h-07z" secondAttribute="leading" constant="8" id="xwM-oH-jQU"/>
        <constraint firstItem="tU5-uw-pz7" firstAttribute="top" secondItem="
ZF1-2h-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="0.6666666666999997" 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="0.6666666666999997" 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-pg-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="AlH-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"/>
    </constraints>

```

```

        <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-l6S"/>
    <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>

```

10.8.5 Step 5: Finalize the View Controller File

In the view controller file, perform the following:

- Set delegate and initialize `IDT_VP3300` singleton object in the `viewDidLoad` method. [Reference](#) : [Allocate/Initialize IDT_VP3300 Object](#)

```

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

    IDT_VP3300.sharedController().delegate = self
}

```

- Implement protocol delegate `IDT_VP3300Delegate::deviceDisconnected()` and `IDT_VP3300Delegate::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===== \n(message) \n(logTextView.text)"
    logTextView.scrollToVisible(NSRange(location: 0, length: 0))
}

func deviceConnected() {
    setConnectionStatus("Connected", backgroundColor: UIColor.greenColor())
    appendMessageToLog("VP3300 Connected\nFramework Version: \(IDT_Device.SDK_version())")
}

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

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

```
func swipeMSRData(cardData: IDTMSRData!) {
    NSLog("--MSR event received, type: \(cardData.event), data: \(cardData.encTrack1)")
    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: UNKNOWN")
        }

        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)")
        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)")
        appendMessageToLog("Length Track 2: \(cardData.track2Length)")
        appendMessageToLog("Length Track 3: \(cardData.track3Length)")
        appendMessageToLog("Encoded Track 1: \(cardData.encTrack1 == nil ? "N/A" : cardData.encTrack1.description)")
        appendMessageToLog("Encoded Track 2: \(cardData.encTrack2 == nil ? "N/A" : cardData.encTrack2.description)")
        appendMessageToLog("Encoded Track 3: \(cardData.encTrack3 == nil ? "N/A" : cardData.encTrack3.description)")
        appendMessageToLog("Hash Track 1: \(cardData.hashTrack1 == nil ? "N/A" : cardData.hashTrack1.description)")
        appendMessageToLog("Hash Track 2: \(cardData.hashTrack2 == nil ? "N/A" : cardData.hashTrack2.description)")
        appendMessageToLog("Hash Track 3: \(cardData.hashTrack3 == nil ? "N/A" : cardData.hashTrack3.description)")
    }
}
```

```

description) ")
    appendMessageToLog("KSN: \(cardData.KSN == nil ? "N/A" : cardData.KSN.description)")
    appendMessageToLog("\nSessionID: \(cardData.sessionID == nil ? "N/A" : cardData.sessionID.
description) ")
    appendMessageToLog("\nReader Serial Number: \(cardData.RSN == nil ? "N/A" : cardData.RSN)")
    appendMessageToLog("\nRead Status: \(cardData.readStatus)")

    if cardData.unencryptedTags != nil {
        appendMessageToLog("Unencrypted Tags: \(cardData.unencryptedTags.description)")
    }

    if cardData.encryptedTags != nil {
        appendMessageToLog("Encrypted Tags: \(cardData.encryptedTags.description)")
    }

    if cardData.maskedTags != nil {
        appendMessageToLog("Masked Tags: \(cardData.maskedTags.description)")
    }

    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.description)")
    NSLog("Encoded Track 2: \(cardData.encTrack2 == nil ? "N/A" : cardData.encTrack2.description)")
    NSLog("Encoded Track 3: \(cardData.encTrack3 == nil ? "N/A" : cardData.encTrack3.description)")
    NSLog("Hash Track 1: \(cardData.hashTrack1 == nil ? "N/A" : cardData.hashTrack1.description)")
    NSLog("Hash Track 2: \(cardData.hashTrack2 == nil ? "N/A" : cardData.hashTrack2.description)")
    NSLog("Hash Track 3: \(cardData.hashTrack3 == nil ? "N/A" : cardData.hashTrack3.description)")
    NSLog("SessionID: \(cardData.sessionID == nil ? "N/A" : cardData.sessionID.description)")
    NSLog("nReader Serial Number: \(cardData.RSN == nil ? "N/A" : cardData.RSN)")
    NSLog("Read Status: \(cardData.readStatus)")
    NSLog("KSN: \(cardData.KSN == nil ? "N/A" : cardData.KSN.description)")

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

case EVENT_MSR_BACKSPACE_KEY:
    appendMessageToLog("(Event) MSR Backspace Key received: \(cardData.encTrack1)")

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

case EVENT_MSR_UNKNOWN:
    appendMessageToLog("(Event) MSR unknown event, data: \(cardData.encTrack1)")
case EVENT_MSR_TIMEOUT:
    appendMessageToLog("MSR Timeout")
default:
    break
    }
}
}

```

- Implement protocol delegate `plugStatusChange()` to automatically attempt connection. Reference: [Implement Optional Delegate Protocols](#)

```

func plugStatusChange(deviceInserted: Bool) {
    if deviceInserted {
        appendMessageToLog("Device attached. Attempting to connect...")
        IDT_VP3300.sharedController().device_connectToAudioReader()
    } else {
        appendMessageToLog("Device removed.")
    }
}

```

- 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 = \(error)")

    appendMessageToLog("EMV transaction data response: \(IDT_VP3300.sharedController().

```

```

        device_getResponseCodeString(error))\n")

    if emvData == nil {
        appendMessageToLog("EMV TRANSACTION ERROR. Refer to EMV_RESULT_CODE_V2_response = \(error)")
        return;
    }

    if emvData.resultCodeV2 != EMV_RESULT_CODE_V2_NO_RESPONSE {
        appendMessageToLog("EMV_RESULT_CODE_V2_RESPONSE: \(emvData.resultCodeV2.rawValue)")
    }

    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: \(emvData.unencryptedTags.description)")
    }

    if emvData.encryptedTags != nil {
        appendMessageToLog("Encrypted Tags: \(emvData.encryptedTags.description)")
    }

    if emvData.maskedTags != nil {
        appendMessageToLog("Masked Tags: \(emvData.maskedTags.description)")
    }

    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_VP3300.sharedController().emv_callbackResponseLCD(mode, selection
: 1)
    }
}

```

```

        default:
            break
    }
}

```

- Implement the button press methods

```

func displayReturnError(operation: String, rt: RETURN_CODE) {
    let message = "\ (operation) ERROR: ID-\ (rt.rawValue), Message: \ (
        IDT_VP3300.sharedController().device_getResponseCodeString(Int32(rt.rawValue))) "
    appendMessageToLog(message)
}

@IBAction func getFirmware(sender: UIButton) {
    var result: NSString?
    let rt = IDT_VP3300.sharedController().device_getFirmwareVersion(&result)

    logTextView.text = ""

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

@IBAction func startMSR_CTLs(sender: UIButton) {
    let rt = IDT_VP3300.sharedController().
        ctls_startTransaction()

    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_VP3300.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_VP3300.sharedController().emv_completeOnlineEMVTransaction(true,
        hostResponseTags: IDTUtility.hexToData("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_VP3300.sharedController().
        ctls_cancelTransaction()

    logTextView.text = ""

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

```


Chapter 11

VP3300 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 12

Enumeration Reference

IDTMSRData

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

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

IDTCommon

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

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

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

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

```
typedef enum{
    IDT_DEVICE_BTPAY_IOS = 0,
    IDT_DEVICE_BTPAY_OSX_BT,
    IDT_DEVICE_BTPAY_OSX_USB,
    IDT_DEVICE_UNIPAY_IOS,
    IDT_DEVICE_UNIPAY_OSX_USB,
    IDT_DEVICE_VP3300_IOS,
    IDT_DEVICE_VP3300_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_VP3300_EMV_NO_ICC_MSR_DATA,
    EVENT_VP3300_EMV_FALLBACK_DATA
}EVENT_MSR_Types;
```

```
typedef enum{
    EVENT_ACTIVE_TRANSACTION = 51
}EVENT_CTLT_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_CTLT,
    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_CRYPTOGAM_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_PRESENCE = 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_PRESENCE = 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 13

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 cumulative 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 DESFireViVOCComm 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 result again due to Output RAM is Not enough.
DFEF49	Outcome Parameter Set
DFE↔ F4A	User Interface Request Data
DFEF4B	MSR Equivalent Data Option
DFEF4C	MSR Equivalent Data Track Lengths
DFEF4D	MSR Equivalent Data
DFEF4E	ACT MSD Response DOL
DFEF4F	ACT Decline Response DOL
DFEF50	Terminal Interchange Profile (JCB)
DFEF51	Bypass EMV Completion Output
DFEF52	Re-FallBack times
DFEF53	Dynamic Reader Limits
DFEF54	SmartTap AID Index
DFEF55	Kernel Specific Features
DFEF56	Retry Limit
DFEF57	PPSE Terminate Flags
DFEF59	Terminal Data Setting - Default Amount
DFEF5A	Terminal Data Setting - Tags to Return
DFE↔ F5B	Mask for Tag5A

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

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

9F33 Terminal Capabilities

Byte 1

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

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

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

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

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

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

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

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 14

LCD Foreign Language Mapping Table

ID	Message ID	English	French	Spanish	Chinese
0	MSG_NULL	-	-	-	-
1	MSG_AMOUNT	AMOUNT	MONTANT	CANTIDAD	金
2	MSG_AMOUNT_↔ OK	AMOUNT OK?	MONTANT OK	MONTO CORRE↔ CTO?	确定金
3	MSG_APPROVED	APPROVED	APPROUVE	APROVADO	通
4	MSG_CALL_YO↔ UR_BANK	CALL YOUR BANK	APPE VOTRE B↔ ANQE	LLAME A SU BA↔ NCO	系您的行
5	MSG_CANCEL_↔ OR_ENTER	CANCEL OR EN↔ TER	ANNULE OU EN↔ TRER	CANCEL O ENT↔ RAR	取消或确定
6	MSG_CARD_ER↔ ROR	CARD ERROR	ERREUR CARTE	ERROR DE TAR↔ JETA	卡
7	MSG_DECLINED	DECLINED	REFUSE	DECLINADO	卡被拒
8	MSG_ENTER_A↔ MOUNT	ENTER AMOUNT	ENTRER MONT↔ ANT	INGRESE MONTO	入金
9	MSG_ENTER_PIN	ENTER PIN:	ENTRER PIN:	ENTRAR NPI:	入密
10	MSG_INCORRE↔ CT_PIN	INCORRECT PIN	NIP INCORRECT	NPI INCORRECTO	密
11	MSG_ICC_MSR1	SWIPE OR INSE↔ RT	PASSER OU INS↔ ERT	MOVER O INSERT	刷卡或插卡
12	MSG_ICC_MSR2	CARD	CARTE	TARJETA	卡
13	MSG_INSERT_↔ CARD	INSERT CARD	INSERT CARTE	INSERTAR TAR↔ JETA	插卡
14	MSG_USE_CHI↔ P_READER	USE CHIP READ↔ ER UTI	LECTEUR CHIP	USO CHIP LECT↔ OR	使用芯片卡
15	MSG_NOT_ACC↔ EPTED	NOT ACCEPTED	PAS ACCEPTE	DENEGADO	法接受
16	MSG_PIN_OK	GET PIN OK			密正确
17	MSG_PLEASE_↔ WAIT	PLEASE WAIT...	ATTENDRE...	POR FAVOR ES↔ PERE	等候中
18	MSG_PROCESS↔ ING_ERROR	PROCESSING E↔ RROR	ERREUR DE TR↔ AITE	ERROR PROCE↔ SANDO	理
19	MSG_USE_MAG↔ STRIPE	USE MAGSTRIPE	USAGE MAGST↔ RIPE	USO DE MAGST↔ RIPE	使用磁卡
20	MSG_TRY_AGAIN	TRY AGAIN	REESSAYER	VUELV INTENTA↔ RLO	重
21	MSG_ONLINE	GO ONLINE	GO LIGNE	GO LINEA	在

ID	Message ID	English	French	Spanish	Chinese
22	MSG_TRANSACTION_ERROR↔	TRANSACTION ERR	ERREUR DE TRANSACTIONS	ERROR DE TRANSAC↔	交易
23	MSG_TERMINATE	TERMINATE	RESILIER	TERMINAR	止
24	MSG_ADVICE	ADVICE	CONSEILS	CONSEJOS	建
25	MSG_TIMEOUT	TIME OUT	TIMEOUT	TIEMPO DE ESPERA↔	超
26	MSG_PROCESSING↔	PROCESSING...	PROCESSUS...	PROCESANDO...	理中。。。
27	MSG_PIN_TRY_EX↔	PIN TRY LIMIT EX	PIN TRY DEPASSE	TRY PIN SUPERADA↔	密次多
28	MSG_ISSUER_AUTH_FAIL↔	ISSUER AUTH FAIL	EMETTEUR FAIL	EMISOR FALLA	与卡机构
29	MSG_CONTINUE_PROCESS↔	CONTINUE PROCESS	CONTINUER LA	CONTINUAR PROCES↔	理
30	MSG_GET_PIN_ERROR↔	GET PIN ERROR	GET PIN ERROR	OBTENER PIN ERROR↔	密
31	MSG_GET_PIN_FAIL↔	GET PIN FAIL	GET PIN FAIL	OBTENER PIN FALL↔	取密
32	MSG_NOKEY_GET_PIN↔	NO KEY GET PIN	NO KEY GET PIN	NO CLAVE GET PIN	法入密
33	MSG_CANCELLED↔	CANCELLED	ANNULE	CANCELADO	取消
34	MSG_LAST_PIN_TRY↔	LAST PIN TRY	-	-	最后一次入密

Chapter 15

Hierarchical Index

15.1 Class Hierarchy

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

ADF_Info	87
AIDEntry	87
ApplicationID	89
CAKey	90
CRLEntry	91
ICCReaderStatus	91
<IDT_Device_Delegate>	
IDT_VP3300	92
IDTechEMV	146
IDTech_ApplicationID	144
IDTech_TerminalData	145
MaskAndEncryption	160
NSObject	
APDUResponse	87
IDT_VP3300	92
IDTechEMV	146
IDTEMVData	155
IDTMSRData	157
<NSObject>	
<IDT_VP3300_Delegate>	140
<IDTechEMV_Delegate>	153
PowerOnStructure	161
TerminalData	162

Chapter 16

Class Index

16.1 Class List

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

ADF_Info	87
AIDEntry	87
APDUResponse	87
ApplicationID	89
CAKey	90
CRLEntry	91
ICCReaderStatus	91
IDT_VP3300	92
<IDT_VP3300_Delegate>	140
IDTech_ApplicationID	144
IDTech_TerminalData	145
IDTechEMV	146
<IDTechEMV_Delegate>	153
IDTEMVData	155
IDTMSRData	157
MaskAndEncryption	160
PowerOnStructure	161
TerminalData	162

Chapter 17

Class Documentation

17.1 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

17.2 AIDEntry Struct Reference

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

Public Attributes

- unsigned char [aid](#) [16]
AID value as per payment networks.
- unsigned char [aidLen](#)
AID's length.

17.2.1 Detailed Description

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

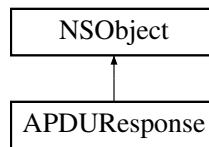
The documentation for this struct was generated from the following file:

- Source_iOS/IDTCommon.h

17.3 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.

17.3.1 Detailed Description

Used in `IDT_BTpay::icc_exchangeAPDU:encrypted:ksn:response:()` `IDT_UniPay::icc_exchangeAPDU:encrypted↵:response:()`

17.3.2 Method Documentation

17.3.2.1 - (void) clear

clears all [APDUResponse](#) properties

17.3.2.2 + ([APDUResponse *](#)) sharedController

Singleton instance of [APDUResponse](#)

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

- `Source_iOS/APDUResponse.h`

17.4 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 [floorLimitChecking](#)
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 [velocityChecking](#)
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.

17.4.1 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

17.5 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.

17.5.1 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

17.6 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.

17.6.1 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

17.7 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.

17.7.1 Detailed Description

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

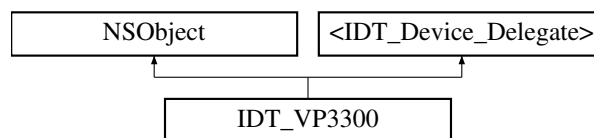
The documentation for this struct was generated from the following file:

- Source_iOS/IDTCommon.h

17.8 IDT_VP3300 Class Reference

```
#import <IDT_VP3300.h>
```

Inheritance diagram for IDT_VP3300:



Instance Methods

- (void) - [close](#)
- (void) - [device_disconnectBLE](#)
- (RETURN_CODE) - [ctls_cancelTransaction](#)
- (RETURN_CODE) - [emv_cancelTransaction](#)
- (RETURN_CODE) - [device_setTerminalData:](#)
- (RETURN_CODE) - [device_retrieveTerminalData:](#)
- (RETURN_CODE) - [device_addTLVToTerminalData:](#)
- (RETURN_CODE) - [device_cancelTransaction](#)
- (RETURN_CODE) - [device_getTransactionResults:](#)
- (RETURN_CODE) - [ctls_getConfigurationGroup:response:](#)
- (void) - [processBypassResponse:](#)
- (void) - [assignBypassDelegate:](#)
- (RETURN_CODE) - [ctls_removeAllCAPK](#)
- (RETURN_CODE) - [ctls_removeApplicationData:](#)
- (RETURN_CODE) - [ctls_removeCAPK:](#)
- (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) - [device_sendGen2Cmd:response:](#)
- (RETURN_CODE) - [ctls_setApplicationData:](#)
- (RETURN_CODE) - [ctls_setCAPK:](#)
- (RETURN_CODE) - [ctls_setConfigurationGroup:](#)

- (RETURN_CODE) - [ctls_setTerminalData:](#)
- (RETURN_CODE) - [ctls_startTransaction:type:timeout:tags:](#)
- (RETURN_CODE) - [ctls_startTransaction](#)
- (RETURN_CODE) - [device_cancelConnectToAudioReader](#)
- (RETURN_CODE) - [device_connectToAudioReader](#)
- (RETURN_CODE) - [device_getFirmwareVersion:](#)
- (bool) - [device_enableBLEDeviceSearch:](#)
- (void) - [setBLEDeviceTypeVP3300](#)
- (void) - [setServiceUUID:](#)
- (NSString *) - [device_getBLEFriendlyName](#)
- (void) - [device_setBLEFriendlyName:](#)
- (bool) - [device_disableBLEDeviceSearch](#)
- (NSUUID *) - [device_connectedBLEDevice](#)
- (BOOL) - [device_isAudioReaderConnected](#)
- (RETURN_CODE) - [device_getAutoPollTransactionResults:](#)
- (NSString *) - [device_getResponseCodeString:](#)
- (bool) - [device_isConnected:](#)
- (RETURN_CODE) - [device_sendIDGCommand:subCommand:data:response:](#)
- (RETURN_CODE) - [device_sendIDGCommandV3:subCommand:data:response:](#)
- (RETURN_CODE) - [device_setAudioVolume:](#)
- (RETURN_CODE) - [device_setPassThrough:](#)
- (RETURN_CODE) - [device_setBurstMode:](#)
- (RETURN_CODE) - [device_setPollMode:](#)
- (RETURN_CODE) - [device_startRKI](#)
- (RETURN_CODE) - [device_rebootDevice](#)
- (RETURN_CODE) - [device_getMerchantRecord:record:](#)
- (RETURN_CODE) - [device_setMerchantRecord:enabled:merchantID:merchantURL:](#)
- (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) - [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_retrieveCAPK:index:response:](#)
- (RETURN_CODE) - [emv_retrieveCAPKFile:index:response:](#)
- (RETURN_CODE) - [emv_retrieveCAPKList:](#)
- (RETURN_CODE) - [emv_retrieveCRLList:](#)
- (RETURN_CODE) - [emv_retrieveTerminalData:](#)
- (RETURN_CODE) - [emv_retrieveTransactionResult:retrievedTags:](#)
- (RETURN_CODE) - [emv_setApplicationData:configData:](#)
- (RETURN_CODE) - [emv_setCAPK:](#)
- (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_removeAllApplicationData](#)
- (RETURN_CODE) - [emv_getEMVKernelCheckValue:](#)

- (RETURN_CODE) - [emv_getEMVConfigurationCheckValue:](#)
- (RETURN_CODE) - [emv_callbackResponseMSR:](#)
- (RETURN_CODE) - [ctls_getAllConfigGroups:](#)
- (RETURN_CODE) - [ctls_getAllConfigurationGroups:](#)
- (RETURN_CODE) - [config_getSerialNumber:](#)
- (RETURN_CODE) - [icc_exchangeAPDU:response:](#)
- (RETURN_CODE) - [icc_getICCReaderStatus:](#)
- (RETURN_CODE) - [icc_powerOnICC:](#)
- (RETURN_CODE) - [icc_powerOffICC:](#)
- (RETURN_CODE) - [icc_getKeyFormatForICCDUKPT:](#)
- (RETURN_CODE) - [icc_setKeyFormatForICCDUKPT:](#)
- (RETURN_CODE) - [msr_cancelMSRSwipe](#)
- (RETURN_CODE) - [msr_startMSRSwipe](#)
- (bool) - [isConnected](#)
- (RETURN_CODE) - [device_startTransaction:amtOther:type:timeout:tags:forceOnline:fallback:](#)
- (RETURN_CODE) - [device_getPollMode:](#)
- (RETURN_CODE) - [device_setQuickChipHIDMode:](#)
- (RETURN_CODE) - [device_getProductType:](#)
- (RETURN_CODE) - [device_getProcessorType:](#)
- (RETURN_CODE) - [device_getHardwareInfo:](#)
- (RETURN_CODE) - [device_getUIDofMCU:](#)
- (RETURN_CODE) - [device_pingDevice](#)
- (RETURN_CODE) - [device_getDateTime:](#)
- (RETURN_CODE) - [device_setDateTime](#)
- (RETURN_CODE) - [device_controlLED:control:](#)
- (RETURN_CODE) - [device_controlUserInterface:](#)
- (RETURN_CODE) - [device_getCashTranRiskPara:](#)
- (RETURN_CODE) - [device_getDriReaderRiskPara:tlv:](#)
- (RETURN_CODE) - [device_getModuleVer:](#)
- (RETURN_CODE) - [emv_getEMVKernelVersion:](#)
- (RETURN_CODE) - [emv_getEMVKernelVersionExt:](#)
- (void) - [setReaderAttached:](#)
- (void) - [scanForBLEDevices:serviceUUIDs:options:](#)
- (void) - [scanForBLEDeviceNames:serviceUUIDs:options:](#)
- (RETURN_CODE) - [activateTransaction:timeout:](#)
- (RETURN_CODE) - [device_syncClock](#)
- (RETURN_CODE) - [device_syncTime](#)

Class Methods

- (void) + [disableAudioDetection](#)
- (NSString *) + [SDK_version](#)
- (IDT_VP3300 *) + [sharedController](#)
- (NSString *) + [createFastEMVData:](#)

Properties

- id< [IDT_VP3300_Delegate](#) > [delegate](#)
- id< [IDT_VP3300_Delegate](#) > [bypassDelegate](#)

17.8.1 Detailed Description

Class to drive the [IDT_VP3300](#) device

17.8.2 Method Documentation

17.8.2.1 - (RETURN_CODE) activateTransaction: (NSMutableDictionary< NSString *, NSString * > *_Nullable) tags
timeout:(int) timeout

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

<i>tags</i>	Activate TLV tags
<i>timeout</i>	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;
}
```

17.8.2.2 - (void) assignBypassDelegate: (id< IDT_VP3300_Delegate >) del

Set Bypass Delegate

Sets the bypass delegate. .

Parameters

<i>del</i>	IDT_Device_Delegate delegate.
------------	-------------------------------

17.8.2.3 - (void) close

Close Device

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

Polls device for Serial Number

Parameters

<i>response</i>	Returns Serial Number
-----------------	-----------------------

Returns

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

17.8.2.5 + (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

<i>emvData</i>	The IDTEMVData object populated with card data.
----------------	---

Returns

Fast EMV String data

17.8.2.6 - (RETURN_CODE) ctls_cancelTransaction

Cancels Transaction request (swipe or CTLS).

Returns

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

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

Get All Configuration Groups

Retrieves all Configuration Groups.

Parameters

<i>response</i>	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:()

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

Get All Configuration Groups

Retrieves all Configuration Groups.

Parameters

<i>response</i>	
<i>response</i>	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:()

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

Get Configuration Group

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

Parameters

<i>group</i>	Configuration Group
<i>response</i>	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:](#)

17.8.2.10 - (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:](#)

17.8.2.11 - (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

<i>AID</i>	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](#):

17.8.2.12 - (RETURN_CODE) ctls_removeCAPK: (NSData *) *capk*

Remove Certificate Authority Public Key

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

Parameters

<i>capk</i>	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](#):

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

Remove Configuration Group

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

Parameters

<i>group</i>	Configuration Group
--------------	---------------------

Returns

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

17.8.2.14 - (RETURN_CODE) ctls_retrieveAIDList: (NSArray **) *response*

Retrieve AID list

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

Parameters

<i>response</i>	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](#):

17.8.2.15 - (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

<i>AID</i>	Name of ApplicationID as Hex String (example "A0000000031010"). Must be between 5 and 16 bytes
<i>response</i>	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](#):

17.8.2.16 - (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

<i>capk</i>	6 bytes CAPK = 5 bytes RID + 1 byte Index
<i>key</i>	<p>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:</p> <ul style="list-style-type: none"> • 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](#):

17.8.2.17 - (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

<i>keys</i>	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](#):

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

Retrieve Terminal Data

Retrieves the Terminal Data for CTLS. This is configuration group 0 (Tag FFEE - > FFEE0100). The terminal data can also be retrieved by `ctls_getConfiguraitonGroup(0)`.

Parameters

<i>tlv</i>	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](#):

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

Set Application Data by AID

Sets the Application Data for CTLS as specified by TLV data

Parameters

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

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

Returns

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

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

Set Certificate Authority Public Key

Sets the CAPK for CTLS as specified by the [CAKey](#) structure

Parameters

key	<p>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:</p> <ul style="list-style-type: none"> • 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](#):

17.8.2.21 - (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 (FFE4). 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](#):

17.8.2.22 - (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:()`

17.8.2.23 - (RETURN_CODE) ctls_startTransaction

Enable Transaction Request

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](#):

17.8.2.24 - (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

<i>amount</i>	Transaction amount value (tag value 9F02)
<i>exponent</i>	Number of characters after decimile point
<i>type</i>	Transaction type (tag value 9C).
<i>timeout</i>	Timeout value in seconds.
<i>tags</i>	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 APPLEPAY VAS: To enable ApplePay VAS, first a merchant record must be defined in one of the six available index positions (1-6) using `device_setMerchantRecord`, then container tag FFEE06 must be sent as part of the additional tags parameter of `ctls_startTransaction`. Tag FFEE06 must contain tag 9F26 and 9F22, and can optionally contain tags 9F2B and DFO1. Example FFEE06189F220201009F2604000000009F2B050100000000↔ DF010101 9F22 = two bytes = ApplePay Terminal Application Version Number. Hard defined as 0100 for now. (required) 9F26 = four bytes = ApplePay Terminal Capabilities Information (required)

- Byte 1 = RFU
- Byte 2 = Terminal Type
- - Bit 8 = VAS Support (1=on, 0 = off)
- - Bit 7 = Touch ID Required (1=on, 0 = off)
- - Bit 6 = RFU
- - Bit 5 = RFU
- - Bit 1,2,3,4
- - - 0 = Payment Terminal
- - - 1 = Transit Terminal
- - - 2 = Access Terminal

- - - 3 = Wireless Handoff Terminal
- - - 4 = App Handoff Terminal
- - - 15 = Other Terminal
- Byte 3 = RFU
- Byte 4 = Terminal Mode
- - 0 = ApplePay VAS OR ApplePay
- - 1 = ApplePay VAS AND ApplePay
- - 2 = ApplePay VAS ONLY
- - 3 = ApplePay ONLY 9F2B = 5 bytes = ApplePay VAS Filter. Each byte filters for that specific merchant index (optional) DF01 = 1 byte = ApplePay VAS Protocol. (optional)
- - Bit 1 : 1 = URL VAS, 0 = Full VAS
- - Bit 2 : 1 = VAS Beeps, 0 = No VAS Beeps
- - Bit 3 : 1 = Silent Comm Error, 2 = EMEA Comm Error
- - Bit 4-8 : RFU

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

Add Terminal Data

Adds the specified TLV to the current terminal data .

Parameters

<i>tlv</i>	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:](#)

17.8.2.26 - (RETURN_CODE) device_cancelConnectToAudioReader

Cancel Connect To Audio Reader

Returns

RETURN_CODE

Cancels a connection attempt to an IDTech MSR device connected via the audio port.

17.8.2.27 - (RETURN_CODE) device_cancelTransaction

Cancels Transaction request (EMV, swipe or CTLS).

Returns

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

17.8.2.28 - (NSUUID*) device_connectedBLEDevice

Returns the UUID of the connected BLE device

- VP3300

Returns

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

17.8.2.29 - (RETURN_CODE) device_connectToAudioReader

Connect To Audio Reader

Returns

RETURN_CODE

Attempts to recognize and connect to an IDTech MSR device connected via the audio port.

17.8.2.30 - (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

<i>indexLED</i>	The LED to control starting from the left <ul style="list-style-type: none">• 00: LED 0• 01: LED 1• 02: LED 2• 03: LED 3• FF: All LEDs
<i>control</i>	Turns chosen LED(s) OFF (00) or ON (01)

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.31 - (RETURN_CODE) device_controlUserInterface: (NSData *) values

Control User Interface

Controls the Display, Beep, and LED

Parameters

<i>values</i>	<p>Four bytes to control the user interface elements Byte[0] = LCD Message</p> <ul style="list-style-type: none"> • 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](#):

17.8.2.32 - (bool) device_disableBLEDeviceSearch

Stops searching for Bluetooth Low Energy devices in range

- VP3300

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.

17.8.2.33 - (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.

17.8.2.34 - (bool) device_enableBLEDeviceSearch: (NSUUID *) identifier

Begins searching for Bluetooth Low Energy devices in range

- VP3300

Parameters

<i>identifier</i>	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_VP3300::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 combination 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.

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

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

Parameters

<i>result</i>	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

17.8.2.36 - (NSString*) device_getBLEFriendlyName

Get BLE Friendly Name

Returns

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

17.8.2.37 - (RETURN_CODE) device_getCashTranRiskPara: (NSData **) tlv

Get Cash Transaction Reader Risk Parameters

Returns the TTQ and reader risk parameters that will be used for cash transactions, if enabled

Parameters

<i>tlv</i>	TLV data objects
------------	------------------

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString:](#)

17.8.2.38 - (RETURN_CODE) device_getDateTime: (NSData **) dateTime

Device Get DateTime

Gets the current system date and time of the device for VP3300 and UNIPAYI_V

Parameters

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

Returns

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

17.8.2.39 - (RETURN_CODE) `device_getDrlReaderRiskPara: (Byte) index tlv:(NSData **) tlv`

Get DRL Reader Risk Parameters

Gets the index, application program ID, and reader risk parameters for the DRL settings

Parameters

<i>index</i>	DRL index (01-04)
<i>tlv</i>	TLV data objects

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString:](#)

17.8.2.40 - (RETURN_CODE) `device_getFirmwareVersion: (NSString **) response`

Polls device for Firmware Version

Parameters

<i>response</i>	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:()

17.8.2.41 - (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 [device_getResponseCodeString](#):

17.8.2.42 - (RETURN_CODE) device_getMerchantRecord: (int) *index* record:(NSData **) *record*

Get Merchant Record

Gets the merchant record for ApplePay VAS

Parameters

<i>index</i>	Merchant record index, valid values 1-6
<i>record</i>	Data returned containing 99 bytes <ul style="list-style-type: none"> • Byte 0: Merchant index • Byte 1: Merchant enabled (1 = enabled, 0 = disabled) • Bytes 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](#):

17.8.2.43 - (RETURN_CODE) device_getModuleVer: (NSString **) *moduleVer*

Get Module Version Information

Gets the 16 byte UID of the MCU

Parameters

<i>uid</i>	The string representation of the UID
------------	--------------------------------------

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

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

Get Poll Mode

Gets the current poll mode of the device

Parameters

<i>mode</i>	Response from the device of the current poll mode
-------------	---

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.45 - (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 [device_getResponseCodeString](#):

17.8.2.46 - (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 [device_getResponseCodeString](#):

17.8.2.47 - (NSString *) device_getResponseCodeString: (int) errorCode

Get Response Code String

Interpret a response code and return string description.

Parameters

<i>errorCode</i>	Error code, range 0x0000 - 0xFFFF, example 0x0300
------------------	---

Returns

Verbose error description

17.8.2.48 - (RETURN_CODE) device_getTransactionResults: (NSData **) results**Get Transaction Results**

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

Parameters

<i>results</i>	The transaction results
----------------	-------------------------

Returns

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

17.8.2.49 - (RETURN_CODE) device_getUIDofMCU: (NSString **) response**Get UID of MCU**

Returns the UID of the device

Parameters

<i>response</i>	The module UID information
-----------------	----------------------------

Returns

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

17.8.2.50 - (BOOL) device_isAudioReaderConnected**Is Audio Reader Connected**

Returns value on device connection status when device is an audio-type connected to headphone plug.

Returns

BOOL True = Connected, False = Disconnected

17.8.2.51 - (bool) device_isConnected: (IDT_DEVICE_Types) device**Is Device Connected**

Returns the connection status of the requested device

Parameters

<i>device</i>	Check connectivity of device type
---------------	-----------------------------------

```
typedef enum{
    IDT_DEVICE_VP3300_IOS = 8
    IDT_DEVICE_VP3300_OSX_USB = 9
}IDT_DEVICE_Types;
```

17.8.2.52 - (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 [device_getResponseCodeString](#):

17.8.2.53 - (RETURN_CODE) device_rebootDevice

Reboot Device

- NEO2
- VP3300 -VP8800 -UniPayl_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](#):

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

Retrieve Terminal Data

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

Parameters

<i>responseData</i>	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](#):

17.8.2.55 - (RETURN_CODE) device_sendGen2Cmd: (NSData *) *tlv* response:(NSData **) *response*

Send 2nd Gen Command

Informs SDK to format command for EMV Gen2 Device Communication

Parameters

<i>tlv</i>	TLV Command
<i>response</i>	TLV Response

Returns

RETURN_CODE: Return codes listed as typedef enum in IDTCommon:RETURN_CODE. Values can be parsed with [device_getResponseCodeString](#);

17.8.2.56 - (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

<i>command</i>	One byte command as per NEO IDG Reference Guide
<i>subCommand</i>	One byte sub-command as per NEO IDG Reference Guide
<i>data</i>	Command data (if applicable)
<i>response</i>	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:()

17.8.2.57 - (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

<i>command</i>	One byte command as per NEO IDG Reference Guide
----------------	---

Parameters

<i>subCommand</i>	One byte sub-command as per NEO IDG Reference Guide
<i>data</i>	Command data (if applicable)
<i>response</i>	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():

17.8.2.58 - (RETURN_CODE) device_setAudioVolume: (float) *val*

Set Volume To Audio Reader

Set the iPhone's volume for command communication with audio-based readers. The the range of iPhone's volume is from 0.1 to 1.0.

Parameters

<i>val</i>	Volume level from 0.1 to 1.0
------------	------------------------------

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():

17.8.2.59 - (void) device_setBLEFriendlyName: (NSString *) *friendlyName*

Set BLE Friendly Name

Parameters

<i>friendlyName</i>	Sets the default friendly name to be used when discovering any BLE devices
---------------------	--

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

Send Burst Mode

Sets the burst mode for the device.

Parameters

<i>mode</i>	0 = OFF, 1 = Always On, 2 = Auto Exit
-------------	---------------------------------------

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.61 - (RETURN_CODE) device_setDateTime

Set Date Time

Sets the date and time of the device to the current date and time of the iOS device

Returns

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

17.8.2.62 - (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

<i>index</i>	Merchant Record Index, valid values: 1-6
<i>enabled</i>	Merchant Enabled / valid flag
<i>merchantID</i>	Merchant unique identifier registered with Apple. Example: ID TECH
<i>merchantURL</i>	Merchant URL, when applicable Example: com.idtechproducts.applePay

Returns

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

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

Set Pass Through

Sets Pass-Through mode on VP3300

Parameters

<i>enablePassThrough</i>	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](#):

17.8.2.64 - (RETURN_CODE) device_setPollMode: (int) mode

Send Poll Mode

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

Parameters

<i>mode</i>	0 = Auto Poll, 1 = Poll On Demand
-------------	-----------------------------------

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.65 - (RETURN_CODE) device_setQuickChipHIDMode: (Byte) mode

Set QuickChip HID Mode

Puts the device into QuickChip KB Output Mode while in HID

Parameters

<i>mode</i>	<ul style="list-style-type: none"> • 0 = Disable • 2 = CT + MSR + CL • 3 = CT + MSR • 4 = CL + MSR
-------------	--

Returns

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

17.8.2.66 - (RETURN_CODE) device_setTerminalData: (NSData *) tags

Set Terminal Data

Sets the Terminal Data .

Parameters

<i>responseData</i>	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](#):

17.8.2.67 - (RETURN_CODE) device_startRKI

Start Remote Key Injection

Attempts to perform a Remote Key Injection with IDTech's RKI servers.

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
- 0x0100 through 0xFFFF refer to [device_getResponseCodeString](#):

17.8.2.68 - (RETURN_CODE) device_startTransaction: (double) amount amtOther:(double) amtOther type:(int) type timeout:(int) timeout tags:(NSData *) tags forceOnline:(BOOL) forceOnline fallback:(BOOL) fallback

Start A Transaction Request

Authorizes the EMV, CTLS, or MSR transaction

For a contact ICC EMV transaction, 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 `emvTransaction->Data` delegate protocol, and then `IDT_UniPayII::emv_authenticateTransaction()` must be executed. If auto authorize is ENABLED (default), `IDT_UniPayII::emv_authenticateTransaction()` will automatically be executed after receiving the result EMV_RESULT_CODE_START_TRANSACTION_SUCCESS. The auto authorize can be enabled/disabled with [emv_disableAutoAuthenticateTransaction](#):

Parameters

<i>amount</i>	Transaction amount value (tag value 9F02)
<i>amtOther</i>	Other amount value, if any (tag value 9F03)
<i>type</i>	Transaction type (tag value 9C).
<i>timeout</i>	Timeout value in seconds.
<i>tags</i>	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
<i>forceOnline</i>	TRUE = do not allow offline approval, FALSE = allow ICC to approve offline if terminal capable
<i>autoAuthenticate</i>	Will automatically execute Authenticate Transaction after start transaction returns successful
<i>fallback</i>	Indicate if it supports fallback to MSR

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.69 - (RETURN_CODE) device_syncClock

Sync Clock Syncs the VP3300 clock with the iOS device clock

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):. When no data is available, return code = RETURN_CODE_NO_DATA_AVAILABLE

17.8.2.70 - (RETURN_CODE) device_syncTime

Sync Time Syncs the VP3300 time with the iOS device clock

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):. When no data is available, return code = RETURN_CODE_NO_DATA_AVAILABLE

17.8.2.71 + (void) disableAudioDetection

Disable Audio Detection. For BLE implementations. Removes monitoring headphone jack for audio devices.

17.8.2.72 - (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

<i>tags</i>	Any tags to modify original tags submitted with <code>startTransaction</code> . Passed as <code>NSData</code> . 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:](#)

17.8.2.73 - (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

<i>mode</i>	The choices are as follows <ul style="list-style-type: none"> • 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
<i>selection</i>	Line number in hex (0x01, 0x02), or 'C'/'E' of function key

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString:](#)

17.8.2.74 - (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

<i>MSR</i>	Swiped track data
------------	-------------------

Returns

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

17.8.2.75 - (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

<i>mode</i>	PIN Mode: <ul style="list-style-type: none"> • EMV_PIN_MODE_CANCEL = 0X00, • EMV_PIN_MODE_ONLINE_PIN_DUKPT = 0X01, • EMV_PIN_MODE_ONLINE_PIN_MSK = 0X02, • EMV_PIN_MODE_OFFLINE_PIN = 0X03
<i>KSN</i>	Key Serial Number. If no pairing and PIN is plaintext, value is nil
<i>PIN</i>	PIN data, encrypted. If no pairing, PIN will be sent plaintext

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.76 - (RETURN_CODE) emv_cancelTransaction

Cancels Transaction request (EMV).

Returns

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

17.8.2.77 - (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

<i>isSuccess</i>	Determines if connection to host was successful: <ul style="list-style-type: none"> • 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.
<i>tags</i>	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

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

Disable Auto Authenticate Transaction

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

Parameters

<i>disable</i>	FALSE = auto authenticate ENABLED, TRUE = auto authenticate DISABLED
----------------	--

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

Get EMV Configuration Check Value

Polls device for the EMV Configuration Check Value

Parameters

<i>response</i>	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 [device_getResponseCodeString](#):

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

Get Kernel Check Value

Polls the device for the Kernel Check Value

Parameters

<i>response</i>	Response returned of the Check Value of the Kernel
-----------------	--

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

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

Get EMV Kernel Version

Polls the device for the EMV Kernel Version

Parameters

<i>response</i>	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 [device_getResponseCodeString](#):

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

Get Extended EMV Kernel Version

Polls the device for the extended EMV kernel version

Parameters

<i>response</i>	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 [device_getResponseCodeString](#):

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

Polls device for EMV L2 Version

Parameters

<i>response</i>	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](#):

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

Gets the terminal major configuration in ICS .

Parameters

<i>configuration</i>	A configuration value, range 1-5 <ul style="list-style-type: none"> • 1 = 1C • 2 = 2C • 3 = 3C • 4 = 4C • 5 = 5C
----------------------	---

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.85 - (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 [device_getResponseCodeString](#):

17.8.2.86 - (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

<i>AID</i>	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters
------------	---

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](#):

17.8.2.87 - (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

<i>rid</i>	RID of the key to remove
<i>index</i>	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 [device_getResponseCodeString](#):

17.8.2.88 - (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 [device_getResponseCodeString](#):

17.8.2.89 - (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 [device_getResponseCodeString](#):

17.8.2.90 - (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

<i>response</i>	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 [device_getResponseCodeString](#):

17.8.2.91 - (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

<i>AID</i>	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters
<i>responseAID</i>	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](#):

17.8.2.92 - (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

<i>rid</i>	The RID of the key to retrieve
<i>index</i>	The Index of the key to retrieve
<i>response</i>	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 [device_getResponseCodeString](#):

17.8.2.93 - (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

<i>rid</i>	The RID of the key to retrieve
<i>index</i>	The Index of the key to retrieve
<i>response</i>	Response returned as a NSData object with the following data: <ul style="list-style-type: none"> • 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](#):

17.8.2.94 - (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

<i>response</i>	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 [device_getResponseCodeString](#):

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

Retrieve the Certificate Revocation List

Returns all the RID in the CRL.

Parameters

<i>response</i>	Response returned as an NSArray of NSData objects 9-bytes each: <ul style="list-style-type: none"> • 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](#):

17.8.2.96 - (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](#):

17.8.2.97 - (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](#):

17.8.2.98 - (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 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-existent 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*) DICTotLV:(NSDictionary*)tags;
```

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

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

Parameters

<i>aidName</i>	aidName AID name. Example "a0000000031010"
<i>data</i>	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](#):

17.8.2.99 - (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 [device_getResponseCodeString](#):

17.8.2.100 - (RETURN_CODE) emv_setCAPKFile: (NSData *) file

Set Certificate Authority Public Key

Sets the CAPK as specified by the [CAKey](#) raw format

Parameters

key	<p>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:</p> <ul style="list-style-type: none"> • 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](#):

17.8.2.101 - (RETURN_CODE) emv_setCRLentries: (NSData *) data

Set Certificate Revocation List

Sets the CRL list

Parameters

data	<p>CRLentries as a repeating occurrence of CRL: CRL1 CRL2 ... CRLn. CRL format is</p> <ul style="list-style-type: none"> • 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 [device_getResponseCodeString](#):

17.8.2.102 - (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 NSString 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 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-existent 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

<i>data</i>	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](#):

17.8.2.103 - (RETURN_CODE) emv_setTerminalMajorConfiguration: (int) configuration

Sets the terminal major configuration in ICS .

Parameters

<i>configuration</i>	A configuration value, range 1-5 <ul style="list-style-type: none"> • 1 = 1C • 2 = 2C • 3 = 3C • 4 = 4C • 5 = 5C
----------------------	---

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.104 - (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_VP3300::*emv_authenticateTransaction()* must be executed. If auto authorize is ENABLED (default), IDT_VP3300::*emv_authenticateTransaction()* will automatically be executed after receiving the result EMV_RESULT_CODE_START_TRANSACTION_SUCCESS. The auto authorize can be enabled/disabled with [emv_disableAutoAuthenticateTransaction](#):

Parameters

<i>amount</i>	Transaction amount value (tag value 9F02)
<i>amtOther</i>	Other amount value, if any (tag value 9F03)
<i>type</i>	Transaction type (tag value 9C).
<i>timeout</i>	Timeout value in seconds.
<i>tags</i>	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
<i>forceOnline</i>	TRUE = do not allow offline approval, FALSE = allow ICC to approve offline if terminal capable
<i>autoAuthenticate</i>	Will automatically execute Authenticate Transaction after start transaction returns successful
<i>fallback</i>	Indicate if it supports fallback to MSR

Returns

RETURN_CODE: Values can be parsed with [device_getResponseCodeString](#):

17.8.2.105 - (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

<i>dataAPDU</i>	APDU data packet
<i>response</i>	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](#):

17.8.2.106 - (RETURN_CODE) icc_getICReaderStatus: (ICReaderStatus **) readerStatus

Get Reader Status

Returns the reader status

Parameters

<i>readerStatus</i>	Pointer that will return with the ICReaderStatus results.
---------------------	---

Returns

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

```
ICReaderStatus* readerStatus;
RETURN_CODE rt = [[IDT_Device sharedController] getICReaderStatus:&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];
}
```

17.8.2.107 - (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

<i>format</i>	Response return from method: -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](#):

17.8.2.108 - (RETURN_CODE) icc_powerOffICC: (NSString **) *error*

Power Off ICC

Powers down the ICC

Parameters

<i>error</i>	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

17.8.2.109 - (RETURN_CODE) icc_powerOnICC: (NSData **) *response*

Power On ICC

Power up the currently selected microprocessor card in the ICC reader

Parameters

<i>response</i>	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.

17.8.2.110 - (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

<i>encryption</i>	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](#):

17.8.2.111 - (bool) isConnected

Check if device is connected

17.8.2.112 - (RETURN_CODE) msr_cancelMSRSwipe

Disable MSR Swipe

Cancels MSR swipe request.

Returns

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

17.8.2.113 - (RETURN_CODE) msr_startMSRSwipe

Enable MSR Swipe

Enables CLTS and MSR, waiting for swipe or tap to occur. Returns [IDTEMVData](#) to deviceDelegate::emvTransactionData:()

Returns

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

17.8.2.114 - (void) processBypassResponse: (NSData *) data

Process Bypass Response.

When output is being bypassed, and a command is received in the delegate redirectedOutput and sent to the device, the device response must be sent to this method so the SDK can finishing processing the results;

Parameters

<i>data</i>	The data received from the device that the SDK must process to complete the executing command
-------------	---

17.8.2.115 - (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

<i>scanTime</i>	Amount of time to search for devices before returning alert view.
<i>serviceUUIDs</i>	A null terminated array of CBUUID to filter for specific device type. Sending nil will report all ble devices in range
<i>options</i>	scan options

Note: Please refer to CBCentralManager scanForPeripheralsWithServices for information about serviceUUIDs and options parameters

17.8.2.116 - (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

<i>scanTime</i>	Amount of time to search for devices before returning alert view.
<i>serviceUUIDs</i>	A null terminated array of CBUUID to filter for specific device type. Sending nil will report all ble devices in range
<i>options</i>	scan options

Note: Please refer to CBCentralManager scanForPeripheralsWithServices for information about serviceUUIDs and options parameters

17.8.2.117 + (NSString*) SDK_version

SDK Version

Returns the current version of IDTech.framework

Returns

Framework version

17.8.2.118 - (void) setBLEDeviceTypeVP3300

Sets BLE search filter for VP3300 family

17.8.2.119 - (void) setReaderAttached: (BOOL) *attached*

Set Reader Attached

Forces the device to reader attached state.

Parameters

<i>attached</i>	TRUE = attached, FALSE = not attached
-----------------	---------------------------------------

17.8.2.120 - (void) setServiceUUID: (nullable NSArray< CBUUID * > *) *serviceUUIDs*

Set BLE to filter for specified service UUID

Parameters

<i>setServiceUUID</i>	Sets the UUID filter to be used when discovering BLE devices
-----------------------	--

17.8.2.121 + (IDT_VP3300*) sharedController

Singleton Instance

Establishes an singleton instance of [IDT_VP3300](#) class.

Returns

Instance of [IDT_VP3300](#)

17.8.3 Property Documentation

17.8.3.1 - (id< IDT_VP3300_Delegate >) bypassDelegate [read],[write],[atomic],[strong]

- Reference to [IDT_VP3300_Delegate](#).

17.8.3.2 - (id< IDT_VP3300_Delegate >) delegate [read],[write],[atomic],[strong]

- Reference to [IDT_VP3300_Delegate](#).

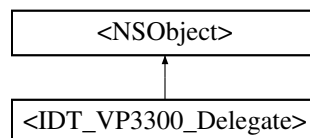
The documentation for this class was generated from the following file:

- Source_iOS/IDT_VP3300.h

17.9 <IDT_VP3300_Delegate> Protocol Reference

```
#import <IDT_VP3300.h>
```

Inheritance diagram for <IDT_VP3300_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) - [plugStatusChange:](#)
- (void) - [dataInOutMonitor:incoming:](#)
- (void) - [swipeMSRData:](#)
- (void) - [deviceMessage:](#)
- (void) - [bypassData:](#)
- (void) - [ctlsEvent:scheme:data:](#)
- (void) - [lcdDisplay:lines:](#)
- (void) - [bluetoothPickerAlert:](#)
- (void) - [bluetoothDeviceNames:](#)
- (void) - [gen2Data:](#)
- (void) - [pinRequest:key:PAN:startTO:intervalTO:language:](#)
- (void) - [emvTransactionData:errorCode:](#)

17.9.1 Detailed Description

Protocol methods established for [IDT_VP3300](#) class

17.9.2 Method Documentation

17.9.2.1 - (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

<i>names</i>	NSArray of device names:
--------------	--------------------------

17.9.2.2 - (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

<i>view</i>	UIAlertView↔ View:
-------------	-----------------------

17.9.2.3 - (void) bypassData: (NSData *) *data* [optional]

When bypass output is enabled, all data intended for the current device will be sent here .

Parameters

<i>data</i>	The data intended for the device
-------------	----------------------------------

17.9.2.4 - (void) ctlsEvent: (Byte) *event* scheme:(Byte) *scheme* data:(Byte) *data* [optional]

Contactless Event During a Contactless transaction, if events are enabled, they will be sent to this protocol,

Parameters

<i>event</i>	Event Type: <ul style="list-style-type: none">• 01 = LED Event• 02 = Buzzer Event• 03 = LCD Message
<i>scheme</i>	LCD Message Scheme

Parameters

<i>data</i>	Data <ul style="list-style-type: none"> • When Event Type 01: – 0x00 = LED0 off – 0x10 = LED1 off – 0x20 = LED2 off – 0x30 = LED3 off – 0xF0 = ALL off – 0x01 = LED0 on – 0x11 = LED1 on – 0x21 = LED2 on – 0x31 = LED3 on – 0xF1 = ALL on – When Event Type 02: – 0x10 = Short Beep No Change – 0x11 = Short Beep No Change – 0x12 = Double Short Beep – 0x13 = Triple Short Beep – 0x20 = 200ms Beep – 0x21 = 400ms Beep – 0x22 = 600ms Beep – When Event Type 03: – Message ID (please refer to table in NEO Reference Guide)
-------------	---

17.9.2.5 - (void) dataInOutMonitor: (NSData *) *data* incoming:(BOOL) *isIncoming* [optional]

All incoming/outgoing data going to the device can be monitored through this delegate.

Parameters

<i>data</i>	The serial data represented as a NSData object
<i>isIncoming</i>	The direction of the data <ul style="list-style-type: none"> • TRUE specifies data being received from the device, • FALSE indicates data being sent to the device.

17.9.2.6 - (void) deviceMessage: (NSString *) *message* [optional]

Receives messages from the framework

Parameters

<i>message</i>	String message transmitted by framework
----------------	---

17.9.2.7 - (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

<i>emvData</i>	EMV Results Data. Result code, card type, encryption type, masked tags, encrypted tags, unencrypted tags and KSN
<i>error</i>	The error code as defined in the errors.h file

17.9.2.8 - (void) gen2Data: (NSData *) *tlv* [optional]

Receives Gen2 TLV data.

Parameters

<i>tlv</i>	TLV data from gen2 event
------------	--------------------------

17.9.2.9 - (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

<i>mode</i>	LCD Display Mode: <ul style="list-style-type: none"> • 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
-------------	---

17.9.2.10 - (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

<i>mode</i>	PIN Mode: <ul style="list-style-type: none"> • 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
<i>key</i>	Either DUKPT or SESSION, depending on mode. If offline plaintext, value is nil
<i>PAN</i>	PAN for calculating PINBlock
<i>startTO</i>	Timeout value to start PIN entry
<i>intervalTO</i>	Timeout value between key presses
<i>language</i>	"EN"=English, "ES"=Spanish, "ZH"=Chinese, "FR"=French

17.9.2.11 - (void) plugStatusChange: (BOOL) *deviceInserted* [optional]

Monitors the headphone jack for device insertion/removal.

Parameters

<i>deviceInserted</i>	TRUE = device inserted, FALSE = device removed
-----------------------	--

17.9.2.12 - (void) swipeMSRData: (IDTMSRData *) cardData [optional]

Receives card data from MSR swipe.

Parameters

cardData	Captured card data from MSR swipe
----------	-----------------------------------

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

- Source_iOS/IDT_VP3300.h

17.10 IDTech_ApplicationID Struct Reference

```
#include <IDTechEMV.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 [applicationVersionNumber](#) [2]
EMV application version number. Tag 9F09.
- unsigned char [tacDefault](#) [5]
Terminal Action Code Denial. Tag DF13.
- unsigned char [tacDenial](#) [5]
Terminal Action Code Denial. Tag DF14.
- unsigned char [tacOnline](#) [5]
Terminal Action Code Denial. Tag DF15.
- unsigned char [transactionCurrencyCode](#) [2]
AID's currency. Example: For US, {0x08,0x40}. Tag 5F2A.
- unsigned char [transactionCurrencyExponent](#)
Transaction Currency Exponent. Example: Amount \$4.53 is managed as 453. Tag 5F36.
- unsigned char [useTACDefault](#)
Indicates if tacDefault value should be used.
- unsigned char [useTACDenial](#)
Indicates if tacDefault value should be used.
- unsigned char [useTACOnline](#)
Indicates if tacDefault value should be used.
- unsigned char [applicationSelectionIndicator](#)
Indicates if partial AID matching is allowed. 0x01 = allowed DF62.

17.10.1 Detailed Description

device AID File - 47 bytes

Used as parameter in [setApplicationData: \(IDTechEMV\)](#)

Used as return value of aidResponse in [retrieveApplicationData: \(IDTechEMV\)](#)

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

- Source_iOS/IDTechEMV.h

17.11 IDTech_TerminalData Struct Reference

```
#include <IDTechEMV.h>
```

Public Attributes

- unsigned char [terminalCountryCode](#) [2]
Terminal's location. Tag 9F1A {0x08,0x40}.
- unsigned char [merchantCategoryCode](#) [2]
Classifies the type of business being done by the merchant, represented according to ISO 8583:1993.
- unsigned char [merchantID](#) [15]
When concatenated with the Acquirer Identifier, uniquely identifies a given merchant.
- unsigned char [terminalID](#) [8]
Designates the unique location of a terminal at a merchant.
- unsigned char [defaultTACDefault](#) [5]
Default Terminal Action Code Denial. Tag DF13.
- unsigned char [defaultTACDenial](#) [5]
Default Terminal Action Code Denial. Tag DF14.
- unsigned char [defaultTACOnline](#) [5]
Default Terminal Action Code Denial. Tag DF15.
- unsigned char [useDefaultTACDefault](#)
Indicates if tacDefault value should be used as terminal default value.
- unsigned char [useDefaultTACDenial](#)
Indicates if tacDenial value should be used as terminal default value.
- unsigned char [useDefaultTACOnline](#)
Indicates if tacOnline value should be used as terminal default value.
- unsigned char [terminalLocation](#) [36]
Terminal Location tag 9F4E.

17.11.1 Detailed Description

device Terminal Configuration File - 45 bytes

Used as parameter in [setTerminalData: \(IDTechEMV\)](#)

Used as return value in [setTerminalData: \(IDTechEMV\)](#)

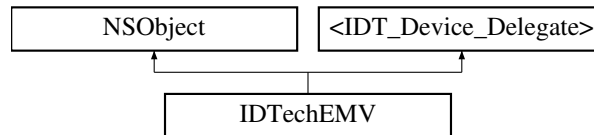
The documentation for this struct was generated from the following file:

- Source_iOS/IDTechEMV.h

17.12 IDTechEMV Class Reference

```
#import <IDTechEMV.h>
```

Inheritance diagram for IDTechEMV:



Instance Methods

- (void) - [completeOnlineEMVTransaction:resultCode:issuerAuthenticationData:issuerScripts:](#)
- (void) - [confirmApplication:](#)
- (void) - [confirmApplicationCancel](#)
- (NSDictionary *) - [getAllTags](#)
- (NSString *) - [getEMVKernelVersion](#)
- (NSString *) - [getBuildVersion](#)
- (NSString *) - [getEMVKernelChecksum](#)
- (NSString *) - [getEMVConfigurationChecksum](#)
- (NSData *) - [getTag:](#)
- (void) - [removeApplicationData:](#)
- (NSArray *) - [retrieveAIDList](#)
- (IDTech_ApplicationID) - [retrieveApplicationData:](#)
- (IDTech_ApplicationID) - [retrieveApplicationData:additionalTags:](#)
- (IDTech_TerminalData) - [retrieveTerminalData](#)
- (void) - [setApplicationData:](#)
- (void) - [setApplicationData:additionalTags:](#)
- (void) - [setTag:tagData:](#)
- (void) - [preloadEMVCardData](#)
- (void) - [setTerminalData:](#)
- (void) - [cancelTransaction](#)
- (void) - [startEMVTransaction:timeout:transactionType:additionalTags:](#)
- (void) - [accelerateRead:](#)
- (void) - [sendSystemError](#)

Class Methods

- (IDTechEMV *) + [sharedController](#)

Properties

- id< [IDTechEMV_Delegate](#) > [delegate](#)

17.12.1 Detailed Description

Class to drive the [IDTechEMV](#) device

17.12.2 Method Documentation

17.12.2.1 - (void) accelerateRead: (BOOL) *enabled*

Accelerate Read Data

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

Parameters

<i>enabled</i>	TRUE = use accelerated reading, FALSE = use standard reading
----------------	--

17.12.2.2 - (void) cancelTransaction

Cancel Transaction

Cancels the current transaction

17.12.2.3 - (void) completeOnlineEMVTransaction: (EMV_COMPLETION_RESULT) *result* resultCode:(NSString *) *rc* issuerAuthenticationData:(NSString *) *iad* issuerScripts:(NSString *) *scripts*

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

<i>result</i>	EMV_COMPLETION_RESULT: Used to specify if contacting online host was successful or other problem occurred
<i>rc</i>	Result Code from host. Mandatory. 2 characters ASCII value. Example "00"
<i>iad</i>	Issuer Authentication Data. Optional. 10 bytes, 20 Hex Characters representing data. Example "11223344556677883030"
<i>scripts</i>	Issuer Scripts. Optional. Data represented by Hex Characters. TLV Format. Must start with 71 or 72, followed by length, followed by data. Example "711000112233445566778899AABBCCDDEEFF". Can string multiple scripts, both 71 and 72.

Results are returned on delegate protocol emvTransactionData.

17.12.2.4 - (void) confirmApplication: (int) *index*

Set Application

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

Parameters

<i>index</i>	The index of the selected app from the application array passed back from confirmApplicationSelection:()
--------------	--

17.12.2.5 - (void) confirmApplicationCancel

Cancel Set Application

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

17.12.2.6 - (NSDictionary*) getAllTags

Get All Tags

Retrieves all EMV tags from the inserted card. Only available after the card has been processed after executing IDTechEMV::startEMVTransaction:timeout:additionalTags:()

Returns

data NSDictionary with all tag values. Key is NSString, Object is NSData.

17.12.2.7 - (NSString*) getBuildVersion

Returns SDK EMV Build Version

Returns

response Response returned of Kernel Version

17.12.2.8 - (NSString*) getEMVConfigurationChecksum

Returns SDK EMV Configuration Version

Returns

response Response returned of Kernel Configuration Version

17.12.2.9 - (NSString*) getEMVKernelChecksum

Returns SDK EMV Kernel Checksum

Returns

response Response returned of Kernel Version

17.12.2.10 - (NSString*) getEMVKernelVersion

Returns SDK EMV Kernel Version

Returns

response Response returned of Kernel Version

17.12.2.11 - (NSData*) getTag: (NSString *) *tagName*

Get Tag

Retrieves an EMV tag from the inserted card. Only available after the card has been processed after executing IDTechEMV::startEMVTransaction:timeout:additionalTags:()

Parameters

<i>tagName</i>	Name fo tag to retrieve
----------------	-------------------------

Returns

data Tag data

17.12.2.12 - (void) preloadEMVCardData**Preload EMV Card Data**

This function will attempt to cache preliminary EMV card data. It is to be used in an environment where there will be a few seconds between inserting the card and executing the startEMV command. Example would be the idle time in collecting an amount before startEVM is executed.

17.12.2.13 - (void) removeApplicationData: (NSString *) AID**Remove Application Data by AID**

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

Parameters

<i>AID</i>	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters
------------	---

17.12.2.14 - (NSArray*) retrieveAIDList**Retrieve AID list**

Returns all the AID name/length on the inserted ICC. Populates response parameter with an dictionary with Keys of AID Names (NSData*), and values of AID Lengths (NSData*)

Returns

response Returns a NSArray of NSString of AID Names

17.12.2.15 - (IDTech_ApplicationID) retrieveApplicationData: (NSString *) AID**Retrieve Application Data by AID**

Retrieves the Application Data as specified by the AID name passed as a parameter. The AID will be in the response parameter responseAID. If aidLen = 0, then requested AID was not found.

Parameters

<i>AID</i>	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters
------------	---

Returns

responseAID The AID returned from the method

17.12.2.16 - (IDTech_ApplicationID) retrieveApplicationData: (NSString *) AID additionalTags:(NSData **) additionalTags

Retrieve Application Data by AID

Retrieves the Application Data as specified by the AID name passed as a parameter. The AID will be in the response parameter responseAID. If aidLen = 0, then requested AID was not found.

Parameters

<i>AID</i>	Name of ApplicationID in ASCII, example "A0000000031020". Must be between 5 and 16 characters Any additional tags specified not included in the IDTech_ApplicationID structure
------------	--

Returns

responseAID The AID returned from the method

17.12.2.17 - (IDTech_TerminalData) retrieveTerminalData

Retrieve Terminal Data

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

Returns

response Response returned as a [TerminalData](#)

17.12.2.18 - (void) sendSystemError

Send Sytem Error

For Internal Use Only

17.12.2.19 - (void) setApplicationData: (IDTech_ApplicationID) data

Set Application Data by AID

Sets the Application Data as specified by the [ApplicationID](#) structure passed as a parameter

Parameters

<i>data</i>	ApplicationID configuration file
-------------	--

17.12.2.20 - (void) setApplicationData: (IDTech_ApplicationID) data additionalTags:(NSData *) additionalTags

Set Application Data with Additional Tags

Sets the Application Data as specified by the [ApplicationID](#) structure passed as a parameter, and optional Additional Tags

Parameters

<i>data</i>	ApplicationID configuration file
-------------	--

17.12.2.21 - (void) setTag: (NSString *) *tagName* tagData:(NSData *) *data*

Set Tag

Sets an EMV tag from the tags read into the terminal. Only available after the card has been processed after executing IDTechEMV::startEMVTransaction:timeout:additionalTags:()

Parameters

<i>tagName</i>	Name of tag to set
<i>data</i>	Tag data

17.12.2.22 - (void) setTerminalData: (IDTech_TerminalData) *data*

Set Terminal Data

Sets the Terminal Data as specified by the [IDTech_TerminalData](#) structure passed as a parameter

Parameters

<i>data</i>	TerminalData configuration file
-------------	---

17.12.2.23 + (IDTechEMV*) sharedController

Singleton Instance

Establishes an singleton instance of [IDTechEMV](#) class.

Returns

Instance of [IDTechEMV](#)

17.12.2.24 - (void) startEMVTransaction: (float) *amount* timeout:(int) *timeout* transactionType:(int) *type*
additionalTags:(NSMutableDictionary *) *tags*

Start EMV Transaction Request

Authorizes the EMV transaction amounts for an ICC card

The tags will be returned in the emvTransactionData delegate protocol.

Parameters

<i>amount</i>	Transaction amount value (tag value 9F02)
<i>timeout</i>	Timeout value in seconds.
<i>type</i>	Transaction Type.
<i>tags</i>	Any other optional tags to be included in the request. Passed as a mutable dictionary.

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
95	05	0000008000
9B	02	E800
9F27	01	01
8A	02	5A33
9F26	08	0C552B9364D55CE5

An example how to create an NSMutableDictionary with these values follows.

```
-(NSMutableDictionary*) createTLVDict{
NSMutableDictionary *emvTags = [[NSMutableDictionary alloc] initWithCapacity:0];

[emvTags setObject:@"0000008000" forKey:@"95"];
[emvTags setObject:@"E800" forKey:@"9B"];
[emvTags setObject:@"01" forKey:@"9F27"];
[emvTags setObject:@"5A33" forKey:@"8A"];
[emvTags setObject:@"0C552B9364D55CE5" forKey:@"9F26"];

return emvTags;
}
```

17.12.3 Property Documentation

17.12.3.1 `-(id< IDTechEMV_Delegate >) delegate` [read],[write],[atomic],[strong]

- Reference to [IDTechEMV_Delegate](#).

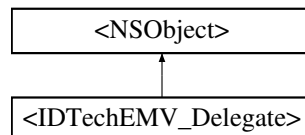
The documentation for this class was generated from the following file:

- Source_iOS/IDTechEMV.h

17.13 <IDTechEMV_Delegate> Protocol Reference

```
#import <IDTechEMV.h>
```

Inheritance diagram for <IDTechEMV_Delegate>:



Instance Methods

- (void) - [confirmApplicationSelection:retry:](#)
- (void) - [languagePreference:](#)
- (void) - [emvTransactionData:errorCode:performReversal:](#)
- (void) - [emvTransactionMessage:](#)
- (void) - [swipeMSRDataEMV:emv:](#)

Receives card data from MSR swipe.

17.13.1 Detailed Description

Protocol methods established for [IDTechEMV](#) class

17.13.2 Method Documentation

17.13.2.1 - (void) **confirmApplicationSelection:** (NSArray *) *labelArray* **retry:**(BOOL) *tryAgain* [optional]

Receives available applications to select from during and EMV transactions when terminal is set for cardholder confirmation or a decision must be made between multiple applications available for selection. Array index of selected app must be passed back to [confirmApplication:](#) ([IDTechEMV](#)) to continue the EMV transaction flow.

Parameters

<i>labelArray</i>	Array of available application labels. Use this array item index to pass back to confirmApplication: (IDTechEMV)
<i>tryAgain</i>	Flag to indicate requirement to display "Try Again" along with application selection.

17.13.2.2 - (void) **emvTransactionData:** (IDTEMVData *) *emvData* **errorCode:**(int) *error* **performReversal:**(BOOL) *reversal* [optional]

EMV Transaction Data

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

Parameters

<i>emvData</i>	EMV Results Data. Result code, card type, encryption type, masked tags, encrypted tags, unencrypted tags and KSN
<i>error</i>	The error code as defined in the errors.h file
<i>reversal</i>	If this flag is TRUE, this means the transaction was APPROVED online, but a TC was not returned by the kernel and the transaction was declined. In most cases, the online host needs to be contacted to reverse the approval.

17.13.2.3 - (void) **emvTransactionMessage:** (MESSAGE_Types) *message* [optional]

EMV Messages

During an EMV Transaction flow , this delegate will pass messages that to request further interaction, or post information

Parameters

<i>message</i>	EMV Message Type- <pre>typedef enum{ MESSAGE_INSERT_CARD = 0, MESSAGE_REMOVE_CARD }EMV_MESSAGE_Types;</pre>
----------------	--

17.13.2.4 - (void) **languagePreference:** (NSData *) *lang* [optional]

Language Preference

After the kernel loads the Application, it will send back the Language Preference, tag 5F2D. This can be used to determine what language to use when displaying prompts and messages for the remainder of the transaction flow.

Parameters

<i>lang</i>	1-4 languages stored in order of preference, each represented by 2 alphabetical characters according to ISO 639.
-------------	--

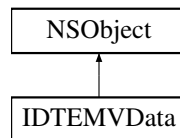
The documentation for this protocol was generated from the following file:

- Source_iOS/IDTechEMV.h

17.14 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.

17.14.1 Detailed Description

Encapsulating data class for EMV data capture

17.14.2 Method Documentation

17.14.2.1 - (void) clear

clears all `IDTEMVData` properties

17.14.2.2 + (IDTEMVData *) sharedController

Singleton instance of `IDTEMVData`

17.14.3 Property Documentation

17.14.3.1 - (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;
```

17.14.3.2 - (EMV_RESULT_CODE_V2_Types) resultCodeV2 [read], [write], [atomic]

Result Code. Uses enumeration `EMV_RESULT_CODE_V2_Types`

```
typedef enum{
    EMV_RESULT_CODE_V2_APPROVED_OFFLINE = 0x0000,
    EMV_RESULT_CODE_V2_DECLINED_OFFLINE = 0x0001,
    EMV_RESULT_CODE_V2_APPROVED = 0x0002,
    EMV_RESULT_CODE_V2_DECLINED = 0x0003,
    EMV_RESULT_CODE_V2_GO_ONLINE = 0x0004,
    EMV_RESULT_CODE_V2_CALL_YOUR_BANK = 0x0005,
    EMV_RESULT_CODE_V2_NOT_ACCEPTED = 0x0006,
    EMV_RESULT_CODE_V2_USE_MAGSTRIPE = 0x0007,
    EMV_RESULT_CODE_V2_TIME_OUT = 0x0008,
```

```

EMV_RESULT_CODE_V2_START_TRANS_SUCCESS = 0x0010,
EMV_RESULT_CODE_V2_MSR_SUCCESS = 0x0011,
EMV_RESULT_CODE_V2_FILE_ARG_INVALID = 0x1001,
EMV_RESULT_CODE_V2_FILE_OPEN_FAILED = 0x1002,
EMV_RESULT_CODE_V2_FILE_OPERATION_FAILED = 0x1003,
EMV_RESULT_CODE_V2_MEMORY_NOT_ENOUGH = 0x2001,
EMV_RESULT_CODE_V2_SMARTCARD_FAIL = 0x3001,
EMV_RESULT_CODE_V2_SMARTCARD_INIT_FAILED = 0x3003,
EMV_RESULT_CODE_V2_FALLBACK_SITUATION = 0x3004,
EMV_RESULT_CODE_V2_SMARTCARD_ABSENT = 0x3005,
EMV_RESULT_CODE_V2_SMARTCARD_TIMEOUT = 0x3006,
EMV_RESULT_CODE_V2_MSR_CARD_ERROR = 0x3007,
EMV_RESULT_CODE_V2_PARSING_TAGS_FAILED = 0x5001,
EMV_RESULT_CODE_V2_CARD_DATA_ELEMENT_DUPLICATE = 0x5002,
EMV_RESULT_CODE_V2_DATA_FORMAT_INCORRECT = 0x5003,
EMV_RESULT_CODE_V2_APP_NO_TERM = 0x5004,
EMV_RESULT_CODE_V2_APP_NO_MATCHING = 0x5005,
EMV_RESULT_CODE_V2_AMANDATORY_OBJECT_MISSING = 0x5006,
EMV_RESULT_CODE_V2_APP_SELECTION_RETRY = 0x5007,
EMV_RESULT_CODE_V2_AMOUNT_ERROR_GET = 0x5008,
EMV_RESULT_CODE_V2_CARD_REJECTED = 0x5009,
EMV_RESULT_CODE_V2_AIP_NOT_RECEIVED = 0x5010,
EMV_RESULT_CODE_V2_AFL_NOT_RECEIVEDE = 0x5011,
EMV_RESULT_CODE_V2_AFL_LEN_OUT_OF_RANGE = 0x5012,
EMV_RESULT_CODE_V2_SFI_OUT_OF_RANGE = 0x5013,
EMV_RESULT_CODE_V2_AFL_INCORRECT = 0x5014,
EMV_RESULT_CODE_V2_EXP_DATE_INCORRECT = 0x5015,
EMV_RESULT_CODE_V2_EFF_DATE_INCORRECT = 0x5016,
EMV_RESULT_CODE_V2_ISS_COD_TBL_OUT_OF_RANGE = 0x5017,
EMV_RESULT_CODE_V2_CRYPTOGAM_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_PRESENCEDE = 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_PRESENCEDE = 0x5035,
EMV_RESULT_CODE_V2_CVM_TYPE_UNKNOWN = 0x6001,
EMV_RESULT_CODE_V2_CVM_AIP_NOT_SUPPORTED = 0x6002,
EMV_RESULT_CODE_V2_CVM_TAG_8E_MISSING = 0x6003,
EMV_RESULT_CODE_V2_CVM_TAG_8E_FORMAT_ERROR = 0x6004,
EMV_RESULT_CODE_V2_CVM_CODE_IS_NOT_SUPPORTED = 0x6005,
EMV_RESULT_CODE_V2_CVM_COND_CODE_IS_NOT_SUPPORTED = 0x6006,
EMV_RESULT_CODE_V2_CVM_NO_MORE = 0x6007,
EMV_RESULT_CODE_V2_PIN_BYPASSED_BEFORE = 0x6008
} EMV_RESULT_CODE_V2_Types;

```

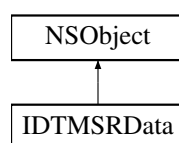
The documentation for this class was generated from the following file:

- Source_iOS/IDTEMVData.h

17.15 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.

17.15.1 Detailed Description

Encapsulating data class for MSR data capture

17.15.2 Method Documentation

17.15.2.1 - (void) clear

clears all [IDTMSRData](#) properties

17.15.2.2 + (IDTMSRData *) sharedController

Singleton instance of [IDTMSRData](#)

17.15.3 Property Documentation

17.15.3.1 - (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;
```

17.15.3.2 - (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;
```

17.15.3.3 - (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_BT_PAY_CARD_DATA,
    EVENT_UNIPAYII_EMV_NO_ICC_MSR_DATA,
    EVENT_UNIPAYII_EMV_FALLBACK_DATA,
    EVENT_UNIPAY_KEYLOADING,
    EVENT_MSR_TIMEOUT
}EVENT_MSR_Types;
```

17.15.3.4 - (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

17.16 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](#)

17.16.1 Detailed Description

Mask and Encryption - Used to Set/Retrieve mask and encryption values IDT_BTPay::emv_retrieveAIDList() IDT↔_UniPay::emv_retrieveAIDList().

17.16.2 Member Data Documentation

17.16.2.1 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

17.16.2.2 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

17.17 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](#)
 - length of pps data

17.17.1 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

17.18 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 [referralManaged](#)
Indicates whether or not Referral 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.

17.18.1 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

- <IDT_VP3300_Delegate>, [140](#)
- <IDTechEMV_Delegate>, [153](#)
- ADF_Info, [87](#)
- AIDEntry, [87](#)
- APDUResponse, [87](#)
 - clear, [88](#)
 - sharedController, [88](#)
- accelerateRead:
 - IDTechEMV, [147](#)
- activateTransaction:timeout:
 - IDT_VP3300, [95](#)
- ApplicationID, [89](#)
- assignBypassDelegate:
 - IDT_VP3300, [96](#)
- bluetoothDeviceNames:
 - IDT_VP3300_Delegate-p, [141](#)
- bluetoothPickerAlert:
 - IDT_VP3300_Delegate-p, [141](#)
- bypassData:
 - IDT_VP3300_Delegate-p, [141](#)
- bypassDelegate
 - IDT_VP3300, [140](#)
- CAKey, [90](#)
- CRLEntry, [91](#)
- cancelTransaction
 - IDTechEMV, [147](#)
- captureEncodeType
 - IDTMSRData, [159](#)
- captureEncryptType
 - IDTMSRData, [159](#)
- clear
 - APDUResponse, [88](#)
 - IDTEMVData, [156](#)
 - IDTMSRData, [159](#)
- close
 - IDT_VP3300, [96](#)
- completeOnlineEMVTransaction:resultCode:issuer↔
 - AuthenticationData:issuerScripts:
 - IDTechEMV, [147](#)
- config_getSerialNumber:
 - IDT_VP3300, [97](#)
- confirmApplication:
 - IDTechEMV, [147](#)
- confirmApplicationCancel
 - IDTechEMV, [147](#)
- confirmApplicationSelection:retry:
 - IDTechEMV_Delegate-p, [154](#)
- createFastEMVData:
 - IDT_VP3300, [97](#)
- ctls_cancelTransaction
 - IDT_VP3300, [97](#)
- ctls_getAllConfigGroups:
 - IDT_VP3300, [97](#)
- ctls_getAllConfigurationGroups:
 - IDT_VP3300, [98](#)
- ctls_getConfigurationGroup:response:
 - IDT_VP3300, [98](#)
- ctls_removeAllCAPK
 - IDT_VP3300, [98](#)
- ctls_removeApplicationData:
 - IDT_VP3300, [98](#)
- ctls_removeCAPK:
 - IDT_VP3300, [99](#)
- ctls_removeConfigurationGroup:
 - IDT_VP3300, [99](#)
- ctls_retrieveAIDList:
 - IDT_VP3300, [99](#)
- ctls_retrieveApplicationData:response:
 - IDT_VP3300, [100](#)
- ctls_retrieveCAPK:key:
 - IDT_VP3300, [100](#)
- ctls_retrieveCAPKList:
 - IDT_VP3300, [100](#)
- ctls_retrieveTerminalData:
 - IDT_VP3300, [101](#)
- ctls_setApplicationData:
 - IDT_VP3300, [101](#)
- ctls_setCAPK:
 - IDT_VP3300, [101](#)
- ctls_setConfigurationGroup:
 - IDT_VP3300, [102](#)
- ctls_setTerminalData:
 - IDT_VP3300, [102](#)
- ctls_startTransaction
 - IDT_VP3300, [102](#)
- ctls_startTransaction:type:timeout:tags:
 - IDT_VP3300, [103](#)
- ctlsEvent:scheme:data:
 - IDT_VP3300_Delegate-p, [141](#)
- dataInOutMonitor:incoming:
 - IDT_VP3300_Delegate-p, [142](#)
- delegate
 - IDT_VP3300, [140](#)
 - IDTechEMV, [153](#)
- device_addTLVToTerminalData:
 - IDT_VP3300, [104](#)

- device_cancelConnectToAudioReader
 - IDT_VP3300, [104](#)
- device_cancelTransaction
 - IDT_VP3300, [104](#)
- device_connectToAudioReader
 - IDT_VP3300, [105](#)
- device_connectedBLEDevice
 - IDT_VP3300, [105](#)
- device_controlLED:control:
 - IDT_VP3300, [105](#)
- device_controlUserInterface:
 - IDT_VP3300, [105](#)
- device_disableBLEDeviceSearch
 - IDT_VP3300, [107](#)
- device_disconnectBLE
 - IDT_VP3300, [107](#)
- device_enableBLEDeviceSearch:
 - IDT_VP3300, [107](#)
- device_getAutoPollTransactionResults:
 - IDT_VP3300, [107](#)
- device_getBLEFriendlyName
 - IDT_VP3300, [108](#)
- device_getCashTranRiskPara:
 - IDT_VP3300, [108](#)
- device_getDateTime:
 - IDT_VP3300, [108](#)
- device_getDrIReaderRiskPara:tlv:
 - IDT_VP3300, [108](#)
- device_getFirmwareVersion:
 - IDT_VP3300, [109](#)
- device_getHardwareInfo:
 - IDT_VP3300, [109](#)
- device_getMerchantRecord:record:
 - IDT_VP3300, [110](#)
- device_getModuleVer:
 - IDT_VP3300, [110](#)
- device_getPollMode:
 - IDT_VP3300, [110](#)
- device_getProcessorType:
 - IDT_VP3300, [111](#)
- device_getProductType:
 - IDT_VP3300, [111](#)
- device_getResponseCodeString:
 - IDT_VP3300, [111](#)
- device_getTransactionResults:
 - IDT_VP3300, [112](#)
- device_getUIDofMCU:
 - IDT_VP3300, [112](#)
- device_isAudioReaderConnected
 - IDT_VP3300, [112](#)
- device_isConnected:
 - IDT_VP3300, [112](#)
- device_pingDevice
 - IDT_VP3300, [113](#)
- device_rebootDevice
 - IDT_VP3300, [113](#)
- device_retrieveTerminalData:
 - IDT_VP3300, [113](#)
- device_sendGen2Cmd:response:
 - IDT_VP3300, [114](#)
- device_sendIDGCommand:subCommand:data↔
 - :response:
 - IDT_VP3300, [114](#)
- device_sendIDGCommandV3:subCommand:data↔
 - :response:
 - IDT_VP3300, [114](#)
- device_setAudioVolume:
 - IDT_VP3300, [115](#)
- device_setBLEFriendlyName:
 - IDT_VP3300, [115](#)
- device_setBurstMode:
 - IDT_VP3300, [115](#)
- device_setDateTime
 - IDT_VP3300, [116](#)
- device_setMerchantRecord:enabled:merchantID↔
 - :merchantURL:
 - IDT_VP3300, [116](#)
- device_setPassThrough:
 - IDT_VP3300, [116](#)
- device_setPollMode:
 - IDT_VP3300, [117](#)
- device_setQuickChipHIDMode:
 - IDT_VP3300, [117](#)
- device_setTerminalData:
 - IDT_VP3300, [117](#)
- device_startRKI
 - IDT_VP3300, [118](#)
- device_startTransaction:amtOther:type:timeout:tags↔
 - :forceOnline:fallback:
 - IDT_VP3300, [118](#)
- device_syncClock
 - IDT_VP3300, [119](#)
- device_syncTime
 - IDT_VP3300, [119](#)
- deviceMessage:
 - IDT_VP3300_Delegate-p, [142](#)
- disableAudioDetection
 - IDT_VP3300, [119](#)
- emv_authenticateTransaction:
 - IDT_VP3300, [119](#)
- emv_callbackResponseLCD:selection:
 - IDT_VP3300, [120](#)
- emv_callbackResponseMSR:
 - IDT_VP3300, [120](#)
- emv_callbackResponsePIN:KSN:PIN:
 - IDT_VP3300, [121](#)
- emv_cancelTransaction
 - IDT_VP3300, [121](#)
- emv_completeOnlineEMVTransaction:hostResponse↔
 - Tags:
 - IDT_VP3300, [121](#)
- emv_disableAutoAuthenticateTransaction:
 - IDT_VP3300, [122](#)
- emv_getEMVConfigurationCheckValue:
 - IDT_VP3300, [122](#)
- emv_getEMVKernelCheckValue:

- IDT_VP3300, [123](#)
- emv_getEMVKernelVersion:
 - IDT_VP3300, [123](#)
- emv_getEMVKernelVersionExt:
 - IDT_VP3300, [123](#)
- emv_getEMVL2Version:
 - IDT_VP3300, [124](#)
- emv_getTerminalMajorConfiguration:
 - IDT_VP3300, [124](#)
- emv_removeAllApplicationData
 - IDT_VP3300, [124](#)
- emv_removeApplicationData:
 - IDT_VP3300, [125](#)
- emv_removeCAPK:index:
 - IDT_VP3300, [125](#)
- emv_removeCRLList
 - IDT_VP3300, [126](#)
- emv_removeTerminalData
 - IDT_VP3300, [126](#)
- emv_retrieveAIDList:
 - IDT_VP3300, [126](#)
- emv_retrieveApplicationData:response:
 - IDT_VP3300, [127](#)
- emv_retrieveCAPK:index:response:
 - IDT_VP3300, [127](#)
- emv_retrieveCAPKFile:index:response:
 - IDT_VP3300, [128](#)
- emv_retrieveCAPKList:
 - IDT_VP3300, [129](#)
- emv_retrieveCRLList:
 - IDT_VP3300, [129](#)
- emv_retrieveTerminalData:
 - IDT_VP3300, [130](#)
- emv_retrieveTransactionResult:retrievedTags:
 - IDT_VP3300, [130](#)
- emv_setApplicationData:configData:
 - IDT_VP3300, [131](#)
- emv_setCAPK:
 - IDT_VP3300, [132](#)
- emv_setCAPKFile:
 - IDT_VP3300, [132](#)
- emv_setCRLEntries:
 - IDT_VP3300, [133](#)
- emv_setTerminalData:
 - IDT_VP3300, [133](#)
- emv_setTerminalMajorConfiguration:
 - IDT_VP3300, [134](#)
- emv_startTransaction:amtOther:type:timeout:tags↔:forceOnline:fallback:
 - IDT_VP3300, [135](#)
- emvTransactionData:errorCode:
 - IDT_VP3300_Delegate-p, [142](#)
- emvTransactionData:errorCode:performReversal:
 - IDTechEMV_Delegate-p, [154](#)
- emvTransactionMessage:
 - IDTechEMV_Delegate-p, [154](#)
- encryptionOption
 - MaskAndEncryption, [161](#)
- event
 - IDTMSRData, [160](#)
- gen2Data:
 - IDT_VP3300_Delegate-p, [142](#)
- getAllTags
 - IDTechEMV, [148](#)
- getBuildVersion
 - IDTechEMV, [148](#)
- getEMVConfigurationChecksum
 - IDTechEMV, [148](#)
- getEMVKernelChecksum
 - IDTechEMV, [148](#)
- getEMVKernelVersion
 - IDTechEMV, [148](#)
- getTag:
 - IDTechEMV, [148](#)
- ICCRReaderStatus, [91](#)
- IDT_VP3300, [92](#)
 - activateTransaction:timeout:, [95](#)
 - assignBypassDelegate:, [96](#)
 - bypassDelegate, [140](#)
 - close, [96](#)
 - config_getSerialNumber:, [97](#)
 - createFastEMVData:, [97](#)
 - ctls_cancelTransaction, [97](#)
 - ctls_getAllConfigGroups:, [97](#)
 - ctls_getAllConfigurationGroups:, [98](#)
 - ctls_getConfigurationGroup:response:, [98](#)
 - ctls_removeAllCAPK, [98](#)
 - ctls_removeApplicationData:, [98](#)
 - ctls_removeCAPK:, [99](#)
 - ctls_removeConfigurationGroup:, [99](#)
 - ctls_retrieveAIDList:, [99](#)
 - ctls_retrieveApplicationData:response:, [100](#)
 - ctls_retrieveCAPK:key:, [100](#)
 - ctls_retrieveCAPKList:, [100](#)
 - ctls_retrieveTerminalData:, [101](#)
 - ctls_setApplicationData:, [101](#)
 - ctls_setCAPK:, [101](#)
 - ctls_setConfigurationGroup:, [102](#)
 - ctls_setTerminalData:, [102](#)
 - ctls_startTransaction, [102](#)
 - ctls_startTransaction:type:timeout:tags:, [103](#)
 - delegate, [140](#)
 - device_addTLVToTerminalData:, [104](#)
 - device_cancelConnectToAudioReader, [104](#)
 - device_cancelTransaction, [104](#)
 - device_connectToAudioReader, [105](#)
 - device_connectedBLEDevice, [105](#)
 - device_controlLED:control:, [105](#)
 - device_controlUserInterface:, [105](#)
 - device_disableBLEDeviceSearch, [107](#)
 - device_disconnectBLE, [107](#)
 - device_enableBLEDeviceSearch:, [107](#)
 - device_getAutoPollTransactionResults:, [107](#)
 - device_getBLEFriendlyName, [108](#)
 - device_getCashTranRiskPara:, [108](#)

- device_getDateTime:, 108
- device_getDriReaderRiskPara:tlv:, 108
- device_getFirmwareVersion:, 109
- device_getHardwareInfo:, 109
- device_getMerchantRecord:record:, 110
- device_getModuleVer:, 110
- device_getPollMode:, 110
- device_getProcessorType:, 111
- device_getProductType:, 111
- device_getResponseCodeString:, 111
- device_getTransactionResults:, 112
- device_getUIDofMCU:, 112
- device_isAudioReaderConnected, 112
- device_isConnected:, 112
- device_pingDevice, 113
- device_rebootDevice, 113
- device_retrieveTerminalData:, 113
- device_sendGen2Cmd:response:, 114
- device_sendIDGCommand:subCommand:data↵
:response:, 114
- device_sendIDGCommandV3:subCommand↵
:data:response:, 114
- device_setAudioVolume:, 115
- device_setBLEFriendlyName:, 115
- device_setBurstMode:, 115
- device_setDateTime, 116
- device_setMerchantRecord:enabled:merchantID↵
:merchantURL:, 116
- device_setPassThrough:, 116
- device_setPollMode:, 117
- device_setQuickChipHIDMode:, 117
- device_setTerminalData:, 117
- device_startRKI, 118
- device_startTransaction:amtOther:type:timeout↵
:tags:forceOnline:fallback:, 118
- device_syncClock, 119
- device_syncTime, 119
- disableAudioDetection, 119
- emv_authenticateTransaction:, 119
- emv_callbackResponseLCD:selection:, 120
- emv_callbackResponseMSR:, 120
- emv_callbackResponsePIN:KSN:PIN:, 121
- emv_cancelTransaction, 121
- emv_completeOnlineEMVTransaction:host↵
ResponseTags:, 121
- emv_disableAutoAuthenticateTransaction:, 122
- emv_getEMVConfigurationCheckValue:, 122
- emv_getEMVKernelCheckValue:, 123
- emv_getEMVKernelVersion:, 123
- emv_getEMVKernelVersionExt:, 123
- emv_getEMVL2Version:, 124
- emv_getTerminalMajorConfiguration:, 124
- emv_removeAllApplicationData, 124
- emv_removeApplicationData:, 125
- emv_removeCAPK:index:, 125
- emv_removeCRLList, 126
- emv_removeTerminalData, 126
- emv_retrieveAIDList:, 126
- emv_retrieveApplicationData:response:, 127
- emv_retrieveCAPK:index:response:, 127
- emv_retrieveCAPKFile:index:response:, 128
- emv_retrieveCAPKList:, 129
- emv_retrieveCRLList:, 129
- emv_retrieveTerminalData:, 130
- emv_retrieveTransactionResult:retrievedTags:, 130
- emv_setApplicationData:configData:, 131
- emv_setCAPK:, 132
- emv_setCAPKFile:, 132
- emv_setCRLEntries:, 133
- emv_setTerminalData:, 133
- emv_setTerminalMajorConfiguration:, 134
- emv_startTransaction:amtOther:type:timeout↵
:tags:forceOnline:fallback:, 135
- icc_exchangeAPDU:response:, 135
- icc_getICCRReaderStatus:, 136
- icc_getKeyFormatForICCDUKPT:, 136
- icc_powerOffICC:, 137
- icc_powerOnICC:, 137
- icc_setKeyFormatForICCDUKPT:, 137
- isConnected, 137
- msr_cancelMSRSwipe, 137
- msr_startMSRSwipe, 138
- processBypassResponse:, 138
- SDK_version, 139
- scanForBLEDeviceNames:serviceUUIDs:options:, 138
- scanForBLEDevices:serviceUUIDs:options:, 138
- setBLEDeviceTypeVP3300, 139
- setReaderAttached:, 139
- setServiceUUID:, 139
- sharedController, 139
- IDT_VP3300_Delegate-p
 - bluetoothDeviceNames:, 141
 - bluetoothPickerAlert:, 141
 - bypassData:, 141
 - ctlsEvent:scheme:data:, 141
 - dataInOutMonitor:incoming:, 142
 - deviceMessage:, 142
 - emvTransactionData:errorCode:, 142
 - gen2Data:, 142
 - lcdDisplay:lines:, 143
 - pinRequest:key:PAN:startTO:intervalTO:language↵
:, 143
 - plugStatusChange:, 143
 - swipeMSRData:, 144
- IDTEMVData, 155
 - clear, 156
 - resultCode, 156
 - resultCodeV2, 156
 - sharedController, 156
- IDTMSRData, 157
 - captureEncodeType, 159
 - captureEncryptType, 159
 - clear, 159
 - event, 160
 - readStatus, 160

- sharedController, 159
- IDTech_ApplicationID, 144
- IDTech_TerminalData, 145
- IDTechEMV_Delegate-p
 - confirmApplicationSelection:retry:, 154
 - emvTransactionData:errorCode:performReversal:, 154
 - emvTransactionMessage:, 154
 - languagePreference:, 154
- IDTechEMV, 146
 - accelerateRead:, 147
 - cancelTransaction, 147
 - completeOnlineEMVTransaction:resultCode↔:issuerAuthenticationData:issuerScripts:, 147
 - confirmApplication:, 147
 - confirmApplicationCancel, 147
 - delegate, 153
 - getAllTags, 148
 - getBuildVersion, 148
 - getEMVConfigurationChecksum, 148
 - getEMVKernelChecksum, 148
 - getEMVKernelVersion, 148
 - getTag:, 148
 - preloadEMVCardData, 150
 - removeApplicationData:, 150
 - retrieveAIDList, 150
 - retrieveApplicationData:, 150
 - retrieveApplicationData:additionalTags:, 150
 - retrieveTerminalData, 151
 - sendSystemError, 151
 - setApplicationData:, 151
 - setApplicationData:additionalTags:, 151
 - setTag:tagData:, 151
 - setTerminalData:, 152
 - sharedController, 152
 - startEMVTransaction:timeout:transactionType↔:additionalTags:, 152
- icc_exchangeAPDU:response:
 - IDT_VP3300, 135
- icc_getICCReaderStatus:
 - IDT_VP3300, 136
- icc_getKeyFormatForICCDUKPT:
 - IDT_VP3300, 136
- icc_powerOffICC:
 - IDT_VP3300, 137
- icc_powerOnICC:
 - IDT_VP3300, 137
- icc_setKeyFormatForICCDUKPT:
 - IDT_VP3300, 137
- isConnected
 - IDT_VP3300, 137
- languagePreference:
 - IDTechEMV_Delegate-p, 154
- lcdDisplay:lines:
 - IDT_VP3300_Delegate-p, 143
- MaskAndEncryption, 160
 - encryptionOption, 161
- maskOption
 - MaskAndEncryption, 161
- msr_cancelMSRSwipe
 - IDT_VP3300, 137
- msr_startMSRSwipe
 - IDT_VP3300, 138
- pinRequest:key:PAN:startTO:intervalTO:language:
 - IDT_VP3300_Delegate-p, 143
- plugStatusChange:
 - IDT_VP3300_Delegate-p, 143
- PowerOnStructure, 161
- preloadEMVCardData
 - IDTechEMV, 150
- processBypassResponse:
 - IDT_VP3300, 138
- readStatus
 - IDTMSRData, 160
- removeApplicationData:
 - IDTechEMV, 150
- resultCode
 - IDTEMVData, 156
- resultCodeV2
 - IDTEMVData, 156
- retrieveAIDList
 - IDTechEMV, 150
- retrieveApplicationData:
 - IDTechEMV, 150
- retrieveApplicationData:additionalTags:
 - IDTechEMV, 150
- retrieveTerminalData
 - IDTechEMV, 151
- SDK_version
 - IDT_VP3300, 139
- scanForBLEDeviceNames:serviceUUIDs:options:
 - IDT_VP3300, 138
- scanForBLEDevices:serviceUUIDs:options:
 - IDT_VP3300, 138
- sendSystemError
 - IDTechEMV, 151
- setApplicationData:
 - IDTechEMV, 151
- setApplicationData:additionalTags:
 - IDTechEMV, 151
- setBLEDeviceTypeVP3300
 - IDT_VP3300, 139
- setReaderAttached:
 - IDT_VP3300, 139
- setServiceUUID:
 - IDT_VP3300, 139
- setTag:tagData:
 - IDTechEMV, 151
- setTerminalData:
 - IDTechEMV, 152
- sharedController
 - APDUResponse, 88

IDT_VP3300, [139](#)
 IDTEMVData, [156](#)
 IDTMSRData, [159](#)
 IDTechEMV, [152](#)
startEMVTransaction:timeout:transactionType:additional↔
 Tags:
 IDTechEMV, [152](#)
swipeMSRData:
 IDT_VP3300_Delegate-p, [144](#)
TerminalData, [162](#)