EMVSwipe API

Integration Guide

iOS Version

August 2014

V2.4.0

Revision History

Rev	Date	Description
1.0.0	4 Feb 2013	Initial Draft
1.1.0	9 Apr 2013	Update **onReturnDeviceInfo** method to return hardwareVersion and supportedTrack Add **resetEmvSwipeController** method Modify the **setAmount** method to add parameter cashbackAmount
1.2.0	19 Apr 2013	Add support for FID 22
1.3.0	7 Jun 2013	Remove init and alloc. Remove releaseAudioResource. ENUM_ prefix removed in enumerations. Change currency in setAmount from int to string.
1.4.0	13 Jun 2013	Add Level 1 commands: powerOnIcc, powerOffIcc, sendApdu Add Level 1 delegates: onReturnPowerOnIccResult, onReturnPowerOffIccResult, onReturnApduResult
1.4.1	14 Jun 2013	Add getKsn and onReturnKsn Modify onReturnDeviceInfo to return NSDictionary.
1.5.0	21 Jun 2013	Add NFC functions - powerOnNfc, powerOffNfc, nfcDataExchange - onReturnPowerOnNfcResult, onReturnPowerOffNfcResult, onReturnNfcDataResult Add 4 new TransactionResult enums TransactionResult_CardNotSupported, TransactionResult_MissingMandatoryData, TransactionResult_CardBlockedOrNoEmvApps, TransactionResult_InvalidIccData Add enums CheckCardResult_NFC_Track2, CheckCardResult_UseIccCard Add enum ErrorType_CommandNotAvailable
1.6.0		Add cardEmulation and onCardEmulationResult
1.6.1	12 July 2013	Rename to getEmvCardData and onReturnEmvCardDataResult Add batteryPercentage to dictionary of onReturnDeviceInfo Fix currency bug and adjust decimal places according to currency code. Allow “,” as decimal point besides “.”.
1.6.2		Update validation rule of setAmount
1.6.3		Remove leading zero of service code
1.6.4		Add TransactionResult ConditionsOfUseNotSatisfied

Continue on next page…

Rev	Date	Description
1.6.5 beta 1	3 Aug 2013	Add new FID 60. Add isDeviceHere method and onDeviceHere callback.
1.6.5	19 Aug 2013	Add sendApduWithPkcs7Padding Add onReturnApduResultWIthPkcs7Padding Fix bug of incorrect apduLength of onReturnApduResult
1.6.6	29 Aug 2013	Add ErrorType AudioRecordingPermissionDenied for iOS7 Add requirement to import AVFoundation.framework Add section about background mode Add ErrorType_BackgroundTimeout
1.6.7	30 Aug 2013	Improve background mode behavior Update condition of triggering ErrorType_BackgroundTimeout
1.6.8	24 Sep 2013	Bug fix of format 60 decoding
1.6.9	09 Oct 2013	Bug fix of missing onWaitingForCard when low battery
1.7.0	24 Oct 2013	Fine tune command timing for iPad Mini Update return value of setAmount function Merge SDK library with some other products. Add new method getIntegratedApiVersion, getInegreatedApiBuildNumber Several functions are marked deprecated.
1.7.1	7 Nov 2013	Add encryptPin method Add onReturnEncryptPinResult callback Add key PAN to onReturnCheckCardResult dictionary Updated EMV Level 2 flow diagram Add tags to getEmvCardData output parameters
1.7.2	11 Nov 2013	Rename TransactionResult_TdkError to TransactionResult_DeviceError Rename TransactionResult_SelectApplicationFail to TransactionResult_CardBlocked. Rename TransactionResult_CardBlockedOrNoEmvApps to TransactionResult_NoEmvApps Add TransactionResult_ApplicationBlocked
1.7.3	19 Nov 2013	Fix decode card data bug of 1.7.1 and 1.7.2 Fix memory leak of 1.7.0, 1.7.1 and 1.7.2 Optimize threading and command timing
1.7.4	22 Nov 2013	Fix audio bug,
1.7.5	26 Nov 2013	Fix onRequestOnlineProcess empty string in 1.7.4
1.7.6	27 Nov 2013	Fix timeout problem of onRequestOnlineProcess

Continue on next page…

Rev	Date	Description
1.8.0	10 Dec 2013	Add encryptData and onReturnEncryptDataResult methods Add releaseEmvSwipeController method for ARC
1.8.1	10 Jan 2014	Add key trackEncoding to onReturnCheckCardResult dictionary
1.8.2	20 Jan 2014	Add cancelCheckCard
1.9.0-beta1	6 Feb 2014	Add NfcDataExchangeStatus enum NfcDataExchangeStatus_Success, NfcDataExchangeStatus_NotYetPowerOn, NfcDataExchangeStatus_NoResponse Change (void) onReturnNfcDataResult:(BOOL)isSuccess data:(NSString *)data dataLength:(int)dataLength; to (void)onReturnNfcDataResult:(NfcDataExchangeStatus)status data:(NSString *)data dataLength:(int)dataLength; Add isSupportedNfc to getDeviceInfo outputs
1.9.1	12 Feb 2014	Add ErrorType_AudioFailToStart_OtherAudioIsPlaying Change (void) startAudio to (BOOL) startAudio
1.9.2	10 Mar2014	Update C++ Standard Library setting Add onReturnCancelCheckCardResult callback
1.9.3	21 Mar2014	Add emvKsn, pinKsn, trackKsn, uid to onReturnDeviceInfo dictionary when using firmware 7.8a or above
2.0.0-beta1	23 Apr 2014	Add readTerminalSetting and updateTerminalSetting methods Add onReturnReadTerminalSettingsResult and onReturnUpdateTerminalSettingResult callbacks Add TerminalSettingStatus enums Add startEmv method with terminal time as parameter. Speed optimization
2.1.0	June 2014	Update the iOS version requirement to 5.0 or above Change the C++ Standard Library to LLVM C++ (Updated the session of Xcode Project Settings)
2.2.0	7 July 2014	Add checkCard overloaded method with parameters Add startEmvWithData method Add checkCard overloaded method with parameters Add finalMessage key to onReturnCheckCardResult Deprecated onReturnStartEmvResult method
2.3.0	21 July 2014	Add FID 46 Add exchangeApdu and onReturnExchangeApduResult
2.4.0	12 Aug 2014	Bug fix of Tag 9A when using startEmvWithData

		Add viposBatchExchangeApdu, onReturnViposBatchExchangeApduResult Rename exchangeApdu and onReturnExchangeApduResult to viposExchangeApdu and onReturnViposExchangeApduResult Add sendVerifyIDResult, onRequestVerifyID Add csn to getKsn and getDeviceInfo Update onReturnTransactionResult with data parameter.

Overview

EMVSwipe is a payment card reading device that works with mobile devices such as mobile phones, tablet computers and notebook computers. It has a magnetic card reader to read magnetic stripe cards and also an EMV card reader to read EMV cards. It communicates with the mobile device through the audio channel and USB.

This document provides the guideline on how to integrate EMVSwipe into their mobile payment application (or App) to accept either magnetic or EMV cards.

System Requirement

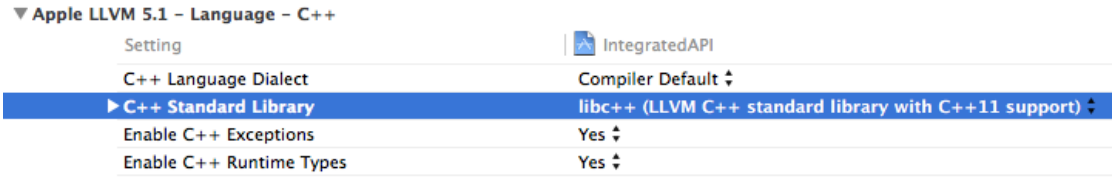
Target System:
iOS 5.0 or above (iOS 5.1 or above is recommended)
iPhone 3GS or above (iPhone 4 or above is recommended)
iPod Touch 3rd Generation or above
iPad 1 or above
iPad Mini 1st Generation or above

Xcode Project Settings

The library needs several frameworks to be included into the Xcode project for a successful compilation.
AudioToolbox.framework
CoreAudio.framework
MediaPlayer.framework
    AVFoundation.framework

The library contains C and C++ files, to let Xcode compiler recognize all source files, the object that contains **EmvSwipeController** have to use filename extension .mm. The AppDelegate.m file also needs to be renamed to AppDelegate.mm.
Link all the Objective-C classes in the static library and set the Other Linker Flags build setting to -ObjC.

In Xcode 5, the compiler options C++ Language Dialect must be set to Compile Default and C++ Standard Library must be set to libc++.



Microphone Access

In iOS7, microphone access by the app must be allowed by user in a per app base, similar to how location service is done. It must be enabled for the EMVSwipe to function. Otherwise, the ErrorType_AudioRecordingPermissionDenied will be triggered.

Background Mode

For better user experience, the app should enable background modes to allow the audio to continue to work when the app goes into background. This prevents interruption on the audio data when the Home button is hit accidentally.

For version below Xcode 5:
In plist, add 'Required background modes' key and 'App plays audio' value

▼ Required background modes	Array	(1 item)
Item 0	String	App plays audio

For Xcode 5 or above:
In plist, add 'Required background modes' key and 'App plays audio or streams audio/video using AirPlay' value

▼ Required background modes	Array	(1 item)
Item 0	String	App plays audio or streams audio/video using AirPlay

A red warning bar will appear when the app enters background mode while audio is in use.

When the app enters background mode, the SDK will still time-out after a certain period and stops the audio. The error ErrorType_BackgroundTimeout will be triggered. This red bar will disappear after that.

The EmvSwipeController Class

The **EmvSwipeController** class is the core of this API library. It has a number of general and utility methods that manage the class itself and a number of methods that communicate with the EMVSwipe device through the audio channel. Communications between the phone and EMVSwipe are bi-directional and a communication can be initiated by both sides. When commands are sent to the App from the EMVSwipe, the **EmvSwipeController** will receive them and delegate functions will be triggered asynchronously.
Command is initiated by the App, because of the nature of the communication channel and the operations, some methods will require a longer time to finish. To avoid blocking the App, all the methods are handled asynchronously. Delegate handler methods are used to obtain the results and they are all prefixed by "on", e.g. **onPowerDown**.

Public Members

Member Name	Description
NSObject <EmvSwipeControllerDelegate>* delegate	This delegate handles asynchronous events and must be implemented by the developer.
BOOL detectDeviceChange	A flag to enable or disable the events of plugging/unplugging an EMVSwipe, **onDevicePlugged**/**onDeviceUnplugged**.

General Methods

Method Name	Description
getApiVersion	Return the API version
getApiBuildNumber	Return the API Build Number
getIntegratedApiVersion	Return the integrated API version
getIntegratedApiBuildNumber	Return the integrated API Build Number
isDevicePresent	Check if the audio jack is plugged. Please note that a normal Headset will also return true.
getEmvSwipeState	Get the state of the EmvSwipeController. There are three states: Idle, AudioStopped and WaitingForResponse
resetEmvSwipeController	Reset and bring the device back to a known initial state.
releaseEmvSwipeController	Release the controller object

startAudio	Start the audio resource for playing and recording
stopAudio	Stop the audio resource for playing and recording

Methods to communicate with EMVSwipe

Method Name	Description
getKsn	Retrieve the KSNs of the EMVSwipe device used for encryption
isDeviceHere	Check if an EMVSwipe is plugged in
getDeviceInfo	Retrieve parameters about the EMVSwipe device. Results are returned by **onReturnDeviceInfo**.
readTerminalSetting	Get the EMV terminal setting
updateTerminalSetting	Update the EMV terminal setting
powerOnIcc	Turns on ICC for level 1 ADPU exchange
powerOffIcc	Turns off ICC after level 1 ADPU exchanges
sendApdu	Send APDU to ICC. Response from ICC is returned by onReturnAdpuResult
sendApduWithPkcs7Padding	Send APDU to ICC. Response from ICC is returned by onReturnAdpuResultWithPkcs7Padding
viposExchangeApdu	Response is returned by onReturnViposExchangeApduResult
viposBatchExchangeApdu	Send a batch of APDU commands to device. Response is returned by onReturnViposBatchExchangeApduResult
powerOnNfc	Turn on the NFC transceiver. Results are returned by **onReturnPowerOnNfcResult.**
powerOffNfc	Turn off the NFC transceiver. Results are returned by **onReturnPowerOffNfcResult.**
nfcDataExchange	Exchange data with NFC card
setAmount	Set the amount and transaction type required for EMV transaction. This can be done before the startEmv command or in response to **onRequestSetAmount**
cancelSetAmount	Cancel the process in response to **onRequestSetAmount**
checkCard (2 overloads)	Check if a card has been inserted or swiped. Results are returned by **onReturnCheckCardResult**
cancelCheckCard	Stop the checkCard process
encryptPin	Encrypt PIN for magnetic swipe operation with PIN Key
encryptData	Encrypt Data for other purposes with Data Key
getEmvCardData	Get card data in the EMV card. This is not part of the EMV payment process
startEmv (2 overloads)	Start an EMV transaction. After receiving this command, EMVSwipe will take control and execute the EMV operation flow.
startEmvWithData	Start an EMV transaction with input parameters including terminal time and various timeouts.
selectApplication	Select an application from a list of applications acceptable by the terminal and the

	EMV card in response to **onRequestSelectApplication**
cancelSelectApplication	Cancel the process in response to **onRequestSelectApplication**
sendPinEntryResult (deprecated)	Send PIN result to **onRequestPinEntry**
bypassPinEntry (deprecated)	Bypass the PIN entry step.
cancelPinEntry (deprecated)	Cancel the PIN entry and abort the transaction
sendFinalConfirmResult	Send confirmation to EMVSwipe in response to **onRequestFinalConfirm**.
sendReferProcessResult (deprecated)	Send referral results from the processor back to EMVSwipe in response to **onRequestReferProcess**.
cancelReferProcess (deprecated)	Cancel referral process and abort transaction
sendAdviceProcessResult(deprecated)f	Send advice results from the processor back to EMVSwipe in response to **onRequestAdviceProcess**.
sendOnlineProcessResult	Send transaction results from the processor back to EMVSwipe in response to **onRequestOnlineProcess**.
sendServerConnectivity	Send the connectivity status to EMVSwipe in response to **onRequestServerConnectivity**
sendTerminalTime	Send the terminal time in YYMMDDHHmmss format to EMVSwipe in response to **onRequestTerminalTime**
sendVerifyIDResult	Send verify cardholder ID result in response to onRequestVerifyID for PBOC

The controller has a public member **delegate** that must be set with an object that implements the protocol **EmvSwipeControllerDelegate**. This delegate handles different asynchronous events that occur during the operation of the EMVSwipe. The developer must design a class that implements the interface and assign it to the **delegate** member.

EmvSwipeControllerDelegate Protocol Methods

Method Name	Description
onWaitingForCard	EMVSwipe is ready and is waiting for a card swipe or a EMV card insert.
onReturnKsn	EMVSwipe has sent back the **getKsn** result.
onDeviceHere	EMVSwipe has responded to **isDeviceHere**
onReturnDeviceInfo	EMVSwipe has sent back the **getDeviceInfo** result.
onReturnReadTerminalSettingResult	EMVSwipe has sent back the **readTerminalSetting** result.
onUpdateReadTerminalSettingResult	EMVSwipe has sent back the **updateTerminalSetting** result.
onReturnPowerOnIccResult	EMVSwipe has responded to powerOnIcc
onReturnPowerOffIccResult	EMVSwipe has responded to powerOffIcc
onReturnApduResult	EMVSwipe has returned APDU in response to sendApdu
onReturnApduResultWithPkcs7Padding	EMVSwipe has returned APDU in response to sendApduWithPkcs7Padding

onReturnViposExchangeApduResult	EMVSwipe has returned APDU in response to viposExchangeApdu
onReturnPowerOnNfcResult	EMVSwipe has responsed to **powerOnNfc**
onReturnPowerOffNfcResult	EMVSwipe has responsed to **powerOffNfc**
onReturnNfcDataResult	EMVSwipe has returned APDU in response to **nfcDataExchange**
onReturnCheckCardResult	EMVSwipe has sent back the **checkCard** result.
onReturnCancelCheckCardResult	EMVSwipe has responsed to **cancelCheckCard**
onReturnEncryptPinResult	EMVSwipe has returned in response to **encryptPin**
onReturnEncryptDataResult	EMVSwipe has returned in response to **encryptData**
onReturnEmvCardDataResult	EMVSwipe has sent back **getEmvCardData** result.
onReturnStartEmvResult (deprecated)	EMVSwipe has responded to **startEmv**
onRequestSetAmount	EMVSwipe has requested to set amount.
onRequestSelectApplication	EMVSwipe has requested to select application.
onRequestPinEntry (deprecated)	EMVSwipe has requested for a PIN entry
onRequestCheckServerConnectivity	EMVSwipe has requested for a check of server connectivity
onRequestFinalConfirm	EMVSwipe has requested for a final confirm before calling the first generate AC command
onRequestOnlineProcess	EMVSwipe has requested for online processing
onRequestReferProcess (deprecated)	EMVSwipe has requested for referral processing
onRequestAdviceProcess (deprecated)	EMVSwipe has requested for advice processing
onRequestTerminalTime	EMVSwipe has requested for the terminal time.
onRequestDisplayText	EMVSwipe has requested to display some text.
onReturnClearDisplay	EMVSwipe has requested to clear the display
onRequestVerifyID	EMVSwipe has requested for checking cardholder ID for PBOC
onReturnTransactionResult	EMVSwipe has sent the final transaction result from the EMV card.
onReturnTransactionLog (deprecated)	EMVSwipe has sent back the transaction log.
onReturnBatchData	EMVSwipe has sent back the batch data
onReturnReversalData	EMVSwipe has sent back the reversal data
onDevicePlugged	EMVSwipe has been plugged in
onDeviceUnplugged	EMVSwipe has been unplugged
onNoDeviceDetected	No EMVSwipe is detected
onBatteryLow	The battery level EMVSwipe is low or critically low. It can occur after checkCard, startEmv or getDeviceInfo. When the level is low, the normal response will still happen. If the level is critically low, the normal response will not occur.
onError	An error has occurred.
onInterrupted	An interruption like incoming call, application switch has occurred.

Instantiation

An instance of an **EmvSwipeController** must be created first by using the sharedController singleton. There should not be more than one instance at any moment.

```
[EmvSwipeController sharedController].delegate = self;
[EmvSwipeController sharedController].detectDeviceChange = YES;
If ([[EmvSwipeController sharedController] startAudio]){
    // audio started successfully
} else { // onError will be triggered.}
```

When the EmvSwipeController instance is no longer used, it should be disposed properly.

```
[[EmvSwipeController sharedController] stopAudio];
 [[EmvSwipeContoller sharedController] setDelegate:nil];
[[EmvSwipeController sharedController] releaseEmvSwipeController];
```

Power On/Off

The EMVSwipe is normally powered down to minimize power consumption and it will be turned on upon any commands received from the host device. After turning on, it is ready to accept commands. To conduct Level 1 transactions, the power on/off of the ICC must be managed by the application. So a typical Level 1 operation sequence will be like: **powerOnIcc**, **sendApdu** (one or more), **powerOffIcc**. To conduct Level 2 transactions, calling **checkCard** and **startEmv** is sufficient.

Reset

At any time, the EMVSwipe can be reset and brought back to a known initial state by using the method **resetEmvSwipeController**.

Get Information about the Device

The **isDeviceHere** can be called to check if an EMVSwipe device is connected. This command can be used to differentiate from out audio interface device.

Information of the device can be obtained by **getKsn** and **getDeviceInfo**.
In the **getKsn** method, a unique identification of the crypto-processor and three sets of KSN used for data encryption are returned.

Key	Description
uid	Unique identifier of the crypto-processor
trackKsn	KSN used for encryption of track data
emvKsn	KSN used for encryption of EMV data, including ATR response, APDU data and other online messages.
pinKsn	KSN used for encryption of online PIN

In the **getDeviceInfo** method, the hardware configuration and status is returned.

Key	Description
bootLoaderVersion	Bootloader Version
firmwareVersion	Firmware Version
hardwareVersion	HardwareVersion
batteryLevel	4-digit voltage level reading
batteryPercentage	Integer of battery percentage e.g. “100”, “99”
isCharging	Indicator of the charging status
isUsbConnected	Indicator of the USB connection status
isSupportedTrack1	Indicator of whether Track 1 is supported
isSupportedTrack2	Indicator of whether Track 2 is supported
isSupportedTrack3	Indicator of whether Track 3 is supported
isSupportedNfc	Indicator of whether NFC is supported

EMV Level 1 APDU exchange

EMV Level 1 APDU exchange can be achieved by using the **sendApdu** or **sendApduWithPkcs7Padding** method and response from the EMV card is returned by the **onReturnApduResult** and **onReturnApduResultWithPkcs7Padding** method respectively.

Before the data exchange the EMV card has to be turned on first by the **powerOnIcc method**. When the data exchange is finished, the EMV card can be turned off by **powerOffIcc**. After **powerOnIcc**, if there are no data communications within 15 minutes, the EMVSwipe will timeout and power off.

The **powerOnIcc**/**powerOffIcc** methods are only required for this EMV Level1 data exchange. There is no need to call them in the **checkCard** and **startEmv** methods below.

NFC Data Exchange

Similarly, NFC data exchange can be achieved by using the **nfcDataExchange** method and response from the EMV card will be returned by the **onReturnNfcDataResult** method.

Before the data exchange the NFC transceiver has to be turned on first by the **powerOnNfc** method. When the data exchange is finished, the NFC transceiver can be turned off by **powerOffNfc** method.

Check Card

EMVSwipe is capable of reading both magnetic stripe cards and EMV cards.
The payment application should first use the **checkCard** method to determine whether the cardholder is using magnetic stripe cards or EMV cards. There is no need to call **powerOnIcc** before this step and the power on/off is done transparently.
If the EMVSwipe supports NFC, then **checkCard** can return encTrack2 that contains the Track 2 equivalent stored on an EMV contactless card.
Note: There is another version of **checkCard** with an NSDictionary as an input parameter for device using FID61.
The **onReturnCheckCardResult** delegate method returns the relevant information.

```
- (void)onReturnCheckCardResult:(CheckCardResult)result
cardDataDict:(NSDictionary *)cardDataDict;
```

result can be one of the following values:

```
typedef enum {
    CheckCardResult_NoCard,
    CheckCardResult_InsertedCard,
    CheckCardResult_NotIccCard,
    CheckCardResult_BadSwipe,
    CheckCardResult_SwipedCard,
    CheckCardResult_MagHeadFail,
    CheckCardResult_NoResponse,
    CheckCardResult_OnlyTrack2
    CheckCardResult_NFC_Track2
    CheckCardResult_UseIccCard
} CheckCardResult;
```

If a magnetic stripe card has been swiped, the encrypted track data will be returned along with the KSN (key serial number) in an NSDictionary.

The NSDictionary contain keys for the values:

Key	**Description**
formatID	The format of the output track data. It supports 22, 36, 54 and 60
Ksn	KSN of the device
PAN	Full PAN (optional)
maskedPAN	Masked card number showing at most the first 6 and last 4 digits with in-between digits masked by “X”
cardholderName	The cardholder name as seen on the card. This can be up to 26 characters.
expiryDate	4-digit in the form of YYMM in the track data
serviceCode	3-digit service code in the track data
encTracks	Encrypted track1 and track2
encTrack1	Encrypted track 1 data with encryption key derived from KSN
encTrack2	Encrypted track 2 data with encryption key derived from KSN
encTrack3	Encrypted track 3 data with encryption key derived from KSN
track1Length	Length of Track 1 data
track2Length	Length of Track 2 data
track3Length	Length of Track 3 data
partialTrack	The 26 characters found in track1 after the first ^ symbol. This part is necessary for the reconstruction of track1.
trackEncoding	Tell if track2 and Track3 data are packed in “ASCII” or “BCD”

finalMessage	(Optional) FID61 device only

The checkCard process can be stopped by the **cancelCheckCard** method.

Start EMV

If an EMV card has been inserted, the **startEmv** command can be called to start the EMV payment process. The EMVSwipe will take control and go through the EMV steps.

```
- (void)startEmv:(EmvOption)EmvOption;
```

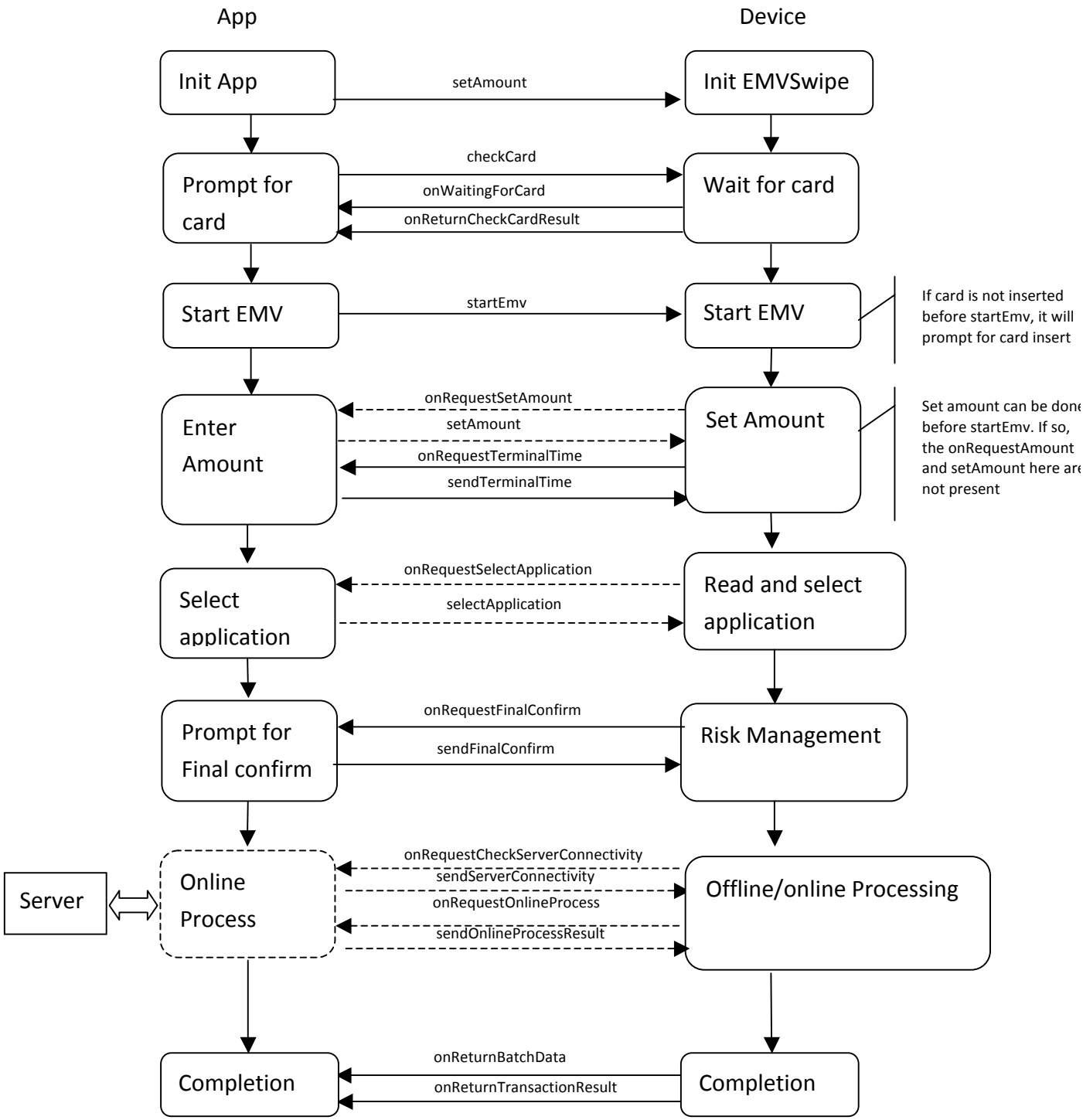
where EmvOption can be used to force an online transaction

```
typedef enum {
    EmvOption_Start,
    EmvOption_StartWithForceOnline
} EmvOption;
```

Another **startEmv** method has the terminalTime parameter in the input. When this overloaded version of **startEmv** is used, the onRequestTerminalTime and onRequestCheckServerConnectivity callbacks will not be triggered and this helps to reduce the transaction time.

The method **startEmvWithData** method can be used to set various timeouts in the EMV transaction.

EMV Level 2Transaction Flow



EMV Operation

EMV Step 0. Start EMV

The **setAmount** command should be called to capture the amount in the transaction. After that, the EMV payment process is started by calling **startEmv**.
The EMVSwipe asks for the terminal time through **onRequestTerminalTime**.
The terminal time in YYMMDDHHmmss format should be sent in response by **sendTerminalTime**.

EMV Step 1. Select Application

An EMV card may support multiple payment applications. The EMVSwipe reads the list of applications supported by the EMV card and asks the customer/operator to select the desired application.

The delegate method **onRequestSelectApplication** is triggered to return an array of application IDs. The app should prompt the user to select one application and then call the **selectApplication** method. The user can also select to abort the transaction by **cancelSelectApplication**.

EMV Step 2. Read Application Data

In this step, EMVSwipe will read the necessary data from the EMV card.

EMV Step 3. Card Authentication

This step is only done between the EMVSwipe and EMV card. If this step fails, **onReturnTransactionResult** will be returned and the EMV process stops.

EMV Step 4. Processing Restrictions

This step is only done between the EMVSwipe and EMV card. If this step fails, **onReturnTransactionResult** will be returned and the EMV process stops.

EMV Step 5. Cardholder Verification

Depending on the requirement of the card, either signature or no verification is required.

EMV Step 6. Terminal Risk Management

This step is only done between the EMVSwipe and EMV card. If this step fails, **onReturnTransactionResult** will be returned and the EMV process stops.

EMV Step 7. Terminal Action Analysis

This step is done between the EMVSwipe and EMV card. If this step fails, **onReturnTransactionResult** will be returned and the EMV process stops.

At the end of this step, a final confirmation is required and **onRequestFinalConfirm** is triggered. The app should prompt the user for a confirmation to proceed. This gives the user a chance to review the amount, the payment method, etc. A final confirmation is sent to EMVSwipe by calling **sendFinalConfirmResult**.

EMV Step 8. Card Risk Management

This step is only done between the EMVSwipe and EMV card. If this step fails, **onReturnTransactionResult** will be returned and the EMV process stops.

EMV Step 9. Online Processing

An EMV transaction can either be online or offline.

If online processing is required, then the **onRequestOnlineProcess** delegate method is triggered. The parameter tlv contains the tag-length-value data structure returned by the EMV kernel.

After reformatting, the client app should send the data to the processor. When the processing results are returned from the processor, it should send the results back to EMVSwipe by **sendOnlineProcessResult.**

The tags and data elements that are required are processor and issuer dependent. Check the processor integration guide for details. See also the EVM Book 3 Annex A for the full list of tags and the TLV structure.

The following tags are usually returned to the ICC.

Tag	Parameter
008A	Authorization Response Code
0091	Issuer Authentication Data
0071	Issuer Script Template 1 (needed for Step 10)
0072	Issuer Script Template 2 (needed for Step 10)

Note: for development, the **sendOnlineProcessResult** parameter can be simulated by using “8A023030” for approval and “8A023035” for decline, corresponding to server response bit 39 having values “00” and “05” respectively.

This step is skipped in offline processing.

EMV Step 10. Issuer Scripts Processing

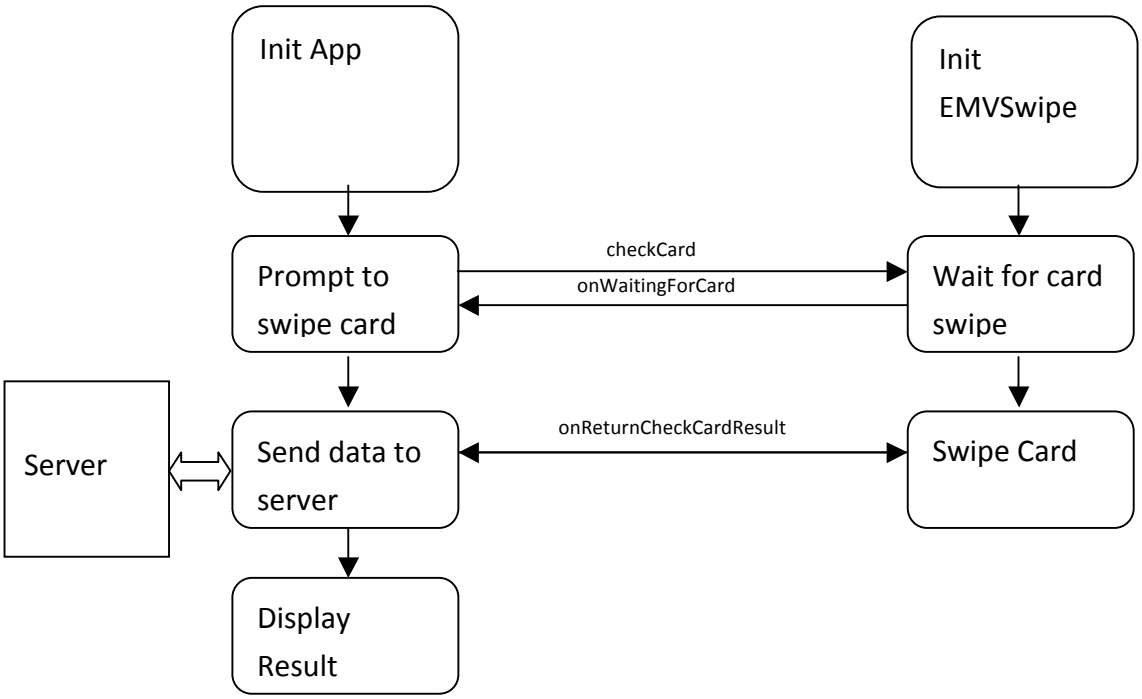
This step is handled transparently between the EMVSwipe and EMV card if issue scripts are present in the online processing results or skipped otherwise.
This step is skipped in offline processing.

EMV Step 11. Completion

In this step, EMVSwipe sends back the final transaction result from the EMV card by **onReturnBatchData**.

In this step, the client app should store the results, display the results, print receipts, and prompt the user to remove the card from the EMVSwipe device. Later, the batch data should be updated to the server for settlement.

Magnetic Card Flow



The operation for magnetic stripe operation is much simpler.
After calling the **checkCard** method, if a card has been swiped successfully, the data are returned in a table containing values encTrack1, encTrack2, encTrack3 and ksn, among others. With these parameters, the track data can be decrypted back in the server.

- (void)checkCard;

- (void)onReturnCheckCardResult:(CheckCardResult)result cardDataDict:(NSDictionary *)cardDataDict;

API Methods Reference

getApiVersion

Signature	- (NSString *)getApiVersion
Inputs	None
Outputs	API version
Description	Return the API version
See also	getApiBuildNumber

getApiBuildNumber;

Signature	- (NSString *)getApiBuildNumber
Inputs	None
Outputs	API Build Number
Description	Return the API Build Number of associated components.
See also	getApiVersion

getIntegratedApiVersion

Signature	- (NSDictionary *)getIntegratedApiVersion
Inputs	None
Outputs	API versions
Description	Return the integrated API versions The keys are “CSwiper”, “EmvSwipe”, “WisePad”
See also	getIntegratedApiBuildNumber

getIntegratedApiBuildNumber;

Signature	- (NSDictionary *)getIntegratedApiBuildNumber
Inputs	None
Outputs	API Build Number
Description	Return the API Build Number of associated components. The keys are “CSwiper”, “EmvSwipe”, “WisePad”
See also	getIntegratedApiVersion

getEmvSwipeState

Signature	- (EmvSwipeControllerState)getEmvSwipeState
Inputs	None
Outputs	Enum of the controller state Possible values: EmvSwipeController_Idle EmvSwipeController _AudioStopped EmvSwipeController _WaitingForResponse
Description	Return the state of the controller object
See also	

resetEmvSwipeController

Signature	- (void)resetEmvSwipeController
Inputs	None
Outputs	None
Description	Reset the state of the EmvSwipe controller
See also	

startAudio

Signature	- (void)startAudio
Inputs	None
Outputs	None
Description	Start the audio resource for playing and recording
See also	stopAudio, onAudioStarted

stopAudio

Signature	- (void)stopAudio
Inputs	None
Outputs	None
Description	Stop and release the audio resource for playing and recording
See also	startAudio

isDevicePresent

Signature	- (BOOL) isDevicePresent
Inputs	None
Outputs	Yes - Presence of an audio device. NO – no audio device is plugged
Description	Check if the audio jack is plugged in.
See also	isDeviceHere

isDeviceHere

Signature	- (void) isDeviceHere
Inputs	None
Outputs	None
Description	Check if an EmvSwipe is connected.
See also	isDevicePresent, onDeviceHere

getKsn

Signature	- (void) getKsn
Inputs	None
Outputs	None
Description	Retrieve the KSN of the EmvSwipe device. Results are returned by onReturnKsn.
See also	onReturnKsn

getDeviceInfo

Signature	- (void) getDeviceInfo
Inputs	None
Outputs	None
Description	Retrieve parameters about the EmvSwipe device. Results are returned by onReturnDeviceInfo which includes: firmware version, bootloader version, USB connection and charging status, battery level, and hardware version
See also	onReturnDeviceInfo

readTerminalSetting

Signature	- (void) readTerminalSetting: (NSString *) tag
Inputs	tag: EMV tag of the terminal configuration to be read, e.g. “9F1A”
Outputs	None
Description	Retrieve an EMV terminal configuration parameter. Common configurable tags include: 9F01: Acquirer Identifier (format n6..11) 9F16: Merchant Identifier (format ans15) 9F1A: Terminal Country Code (format n3) 9F4E: Merchant Name and Location (variable length)
See also	onReturnReadTerminalSettingResult

updateTerminalSetting

Signature	- (void) updateTerminalSetting: (NSString *) tlv
Inputs	tlv: EMV tag of the terminal configuration to be updated
Outputs	None
Description	Update an EMV terminal configuration parameter, provided that it is there. Common configurable tags include: 9F01: Acquirer Identifier (format n6..11) 9F16: Merchant Identifier (format ans15) 9F1A: Terminal Country Code (format n3) 9F4E: Merchant Name and Location (variable length) 9F1B: Terminal floor limit. (format: binary 32) e.g. tlv=“9F1A020840” sets the terminal country code to USA. **Note**: for tag “9F1B”, it will change the terminal floor limit of all applications (AIDs) in the terminal setting. It is not possible to update the floor limit of only one AID at this moment. WARNING: This method should not be called frequently because there is a limited write-life-cycle of the flash memory used to store the parameters. It should only be called once for each deployment/redeployment.
See also	onReturnUpdateTerminalSettingResult

Notes:

The next four methods **powerOnIcc**, **powerOffIcc**, **sendApdu,** **sendApduWithPkcs7Padding** are EMV Level 1 Commands. For EMV Level 2 operation, calling the methods powerOnIcc and powerOffIcc are not required and it is handled by the EMV level 2 operation flow automatically.

powerOnIcc

Signature	- (void)powerOnIcc
Inputs	None
Outputs	None
Description	Provide power to ICC for level 1 APDU exchange
See also	onReturnPowerOnIccResult

powerOffIcc

Signature	- (void)powerOffIcc
Inputs	None
Outputs	None
Description	Cut off power to ICC after level 1 APDU exchange
See also	onReturnPowerOffIccResult

sendApdu

Signature	- (void)sendApdu: (NSString *) apdu apduLength: (int) apduLength
Inputs	apdu: the apdu data to be sent apduLength: length of apdu to be sent
Outputs	None
Description	Send APDU exchange to ICC. The data are padded by zeros padding and then encrypted. The original length of the data is indicated by apduLength. Response data are returned by onReturnApduResult.
See also	onReturnApduResult

sendApduWithPkcs7Padding

Signature	- (void)sendApduWithPkcs7Padding: (NSString *) apdu
Inputs	apdu: the apdu data to be sent
Outputs	None
Description	Send APDU exchange to ICC. The data are padded by the PKCS7 padding and then encrypted. Response data are returned by onReturnApduResultWithPkcs7Padding.
See also	onReturnApduResultWithPkcs7Padding

viposExchangeApdu

Signature	- (void)viposExchangeApdu: (NSString *) apdu
Inputs	apdu: the apdu data to be sent
Outputs	None
Description	Exchange APDU
See also	onReturnViposExchangeApduResult

The next three methods **powerOnNfc**, **powerOffNcc**, **nfcDataExchange** are NFC Commands.

powerOnNfc

Signature	- (void)powerOnNfc:(NSString *)data
Inputs	data: a protocol dependent string to initiate the NFC reader.
Outputs	None
Description	Provide power to NFC transceiver for data exchange
See also	onReturnPowerOnNfcResult

powerOffIcc

Signature	- (void)powerOffNfc
Inputs	None
Outputs	None
Description	Cut off power to NFC transceiver after data exchange
See also	onReturnPowerOffNfcResult

nfcDataExchange

Signature	- (void)nfcDataExchange:(NSString *)data dataLength:(int)dataLength
Inputs	data: the data to be sent dataLength: length of data to be sent
Outputs	None
Description	Send data to NFC card. Response data are returned by onReturnNfcDataResult.
See also	onReturnNfcDataResult

setAmount

Signature	- (BOOL)setAmount:(NSString *)amount cashbackAmount: (NSString *) cashbackAmount currencyCode: (NSString *) currencyCode transactionType: (TransactionType) transactionType
Inputs	amount: the amount for a transaction. cashbackAmount: the amount for a transaction. If this is non-zero, amount cannot be zero. currencyCode: three digits of the currency code, e.g. “840” for USD transactionType: enum of the transaction type.
Outputs	YES: setAmount is successful, NO: setAmount fails
Description	Set the amount, currency and type of a transaction. The amount can have at most one decimal point “.” or “,” and the EMV data field will be adjusted according to the currencyCode. For example “100.00” USD will be represented as 10000 while “100” JPY will be represented as 100. The total of amount and cashback must be at most 12 digits. This method can be called at the beginning of a transaction or in response to an onRequestSetAmount call requested by the EMV engine. If the return result is NO, the application should wait for the onError to handle the error before proceeding further and calling startEmv. Please set **TransactionType** as TransactionType_Goods.
See also	onRequestSetAmount, cancelSetAmount

cancelSetAmount

Signature	- (void)cancelSetAmount
Inputs	None
Outputs	None
Description	Cancel setting the amount of a transaction. This method can be called to abort a transaction in response to onRequestSetAmount
See also	onRequestSetAmount

checkCard

Signature	- (void)checkCard

Inputs	None
Outputs	None
Description	Check the status of the Magnetic Card Reader or the EMV Card reader. It checks if a card has been swiped or an EMV card is inserted. The result is returned by the onReturnCheckCardResult delegate method.
See also	cancelCheckCheck, onReturnCheckCardResult

checkCard

Signature	- (void)checkCard: (NSDictionary*) data
Inputs	data: parameter tables
Outputs	None
Description	Check the status of the Magnetic Card Reader or the EMV Card reader. It checks if a card has been swiped or an EMV card is inserted. The result is returned by the onReturnCheckCardResult delegate method. The keys include: checkCardTimeout: NSNumber of the timeout in second orderID: (Optional) 32 hexadecimal digit NSString used in FID61 devices randomNumber: (Optional) 6 hexadecimal digit NSSString used in FID61 devices
See also	cancelCheckCheck, onReturnCheckCardResult

cancelCheckCard

Signature	- (void)cancelCheckCard
Inputs	None
Outputs	None
Description	Stop the checkCard process
See also	checkCard, onReturnCancelCheckCardResult

encryptPin

Signature	- (void)encryptPin:(NSString *)pin pan:(NSString *)pan
Inputs	pin: 4-12 digits clearText PIN pan: 13-19 digits the full PAN of the card

Outputs	None
Description	Encrypt PIN with PIN key for magnetic stripe operation. The result is returned by the onReturnEncryptPinResult delegate method.
See also	onReturnEncryptPinResult

encryptData

Signature	-(void) encryptData: (NSString *) data
Inputs	data: the data to be encrypt in Hexstring, the data must be packed to multiple of 16 hex characters.
Outputs	None
Description	A general encrypt data operation with Data key. DUKPT key, TDES encryption, CBC mode with zero IV and no padding is used. The result is returned by the onReturnEncryptDataResult delegate method.
See also	onReturnEncryptDataResult

getEmvCardData

Signature	-(void) getEmvCardData
Inputs	None
Outputs	None
Description	Get the card data from an EMV card. This is not part of the EMV payment process.
See also	onReturnEmvCardDataResult

startEmv

Signature	-(void) startEmv: (EmvOption) emvOption
Inputs	emvOption: enum to indicate select online or offline emv.
Outputs	None
Description	Start the EMV process. The EmvSwipe will take control and go through the EMV steps. When started normally, whether the transaction will go online will be a joint decision of EmvSwipe and the EMV card. The Enum **EmvOption** can be EmvOption_Start, EmvOption_StartWithForceOnline

See also	

startEmv

Signature	-(void) startEmv: (EmvOption) emvOption terminalTime:(NSString *)terminalTime
Inputs	emvOption: enum to indicate select online or offline emv. terminalTime: Current local time in the format YYMMDDHHmmss
Outputs	None
Description	Start the EMV process. The EmvSwipe will take control and go through the EMV steps. When started normally, whether the transaction will go online will be a joint decision of EmvSwipe and the EMV card. The Enum **EmvOption** can be EmvOption_Start, EmvOption_StartWithForceOnline When this overloaded version is called, the onRequestTerminalTime and onRequestCheckServerConnectivity will be skipped.
See also	

startEmvWithData

Signature	-(void) startEmvWithData: (NSDictionary*)data
Inputs	data: table of input parameters
Outputs	None
Description	Start the EMV process. The EmvSwipe will take control and go through the EMV steps. When started normally, whether the transaction will go online will be a joint decision of EmvSwipe and the EMV card. The keys include: terminalTime: 12-digit NSString in YYMMDDHHmmss format checkCardTimeout: NSNumber of the timeout in second setAmountTimeout: NSNumber of the timeout in second selectApplicationTimeout: NSNumber of the timeout in second onlineProcessTimeout: NSNumber of the timeout in second finalConfirmTimeout: NSNumber of the timeout in second orderID: (Optional) 32 hexadecimal digit NSString used in FID61

	devices randomNumber: (Optional) 6 hexadecimal digit NSSString used in FID61 devices
See also	

selectApplication

Signature	- (void)selectApplication:(int)applicationIndex
Inputs	applicationIndex: index to the selected application.
Outputs	None
Description	An EMV card may support multiple payment applications. The EmvSwipe reads the list of applications IDs supported by the EMV card and triggers the onRequestSelectApplication delegate method. The app should prompt the user to select one application and then call this method.
See also	onRequestSelectApplication, cancelSelectApplication

cancelSelectApplication

Signature	- (void) cancelSelectApplication
Inputs	None
Outputs	None
Description	An EMV card may support multiple payment applications. The EmvSwipe reads the list of applications IDs supported by the EMV card and triggers the onRequestSelectApplication delegate method and expect the operator to select a payment application. The EMV transaction can be aborted at this step by calling this method.
See also	onRequestSelectApplication, selectApplication

sendPinEntryResult (deprecated)

Signature	- (void) sendPinEntryResult (NSString *)pin;
Inputs	pin: clear text PIN
Outputs	None
Description	This method can be called to send the plaintext Pin in response to onRequestPinEntry.
See also	onRequestPinEntry, cancelPinEntry, bypassPinEntry

bypassPinEntry (deprecated)

Signature	- (void) bypassPinEntry
Inputs	None
Outputs	None
Description	This method can be called to try to bypass the Pin Entry step in response to onRequestPinEntry. If the card does not accept bypassing, the transaction will be aborted.
See also	onRequestPinEntry, cancelPinEntry

cancelPinEntry (deprecated)

Signature	- (void) cancelPinEntry
Inputs	None
Outputs	None
Description	This method can be called to try to abort the transaction in response to onRequestPinEntry.
See also	onRequestPinEntry, bypassPinEntry

sendFinalConfirmResult

Signature	- (void)sendFinalConfirmResult:(BOOL)isConfirmed
Inputs	isConfirmed: YES – proceed, NO - cancel transaction
Outputs	None
Description	The EMV process requested for a final confirm before calling the first generate AC command and the onRequestFinalConfirm method is triggered. The operator can choose to proceed or cancel with this method.
See also	onRequestFinalConfirm

sendTerminalTime

Signature	- (void)sendTerminalTime:(NSString *)terminalTime
Inputs	terminalTime: Current local time in the format YYMMDDHHmmss
Outputs	None
Description	Send terminal time in response to onRequestTerminalTime where the EMV process has requested to get the current time.
See also	onRequestCheckServerConnectivity

sendServerConnectivity

Signature	- (void)sendServerConnectivity:(BOOL)isConnected
Inputs	isConnected: YES – connected, NO – no connection
Outputs	None
Description	Send server connectivity result in response to onRequestCheckServerConnectivity where the EMV process has requested to get the connectivity status before online operation.
See also	onRequestCheckServerConnectivity

sendReferProcessResult (deprecated)

Signature	- (void)sendReferProcessResult:(ReferProcessResult)result
Inputs	result: – ReferProcessResult_Approved: the transaction is approved ReferProcessResult_Declined: the transaction is declined
Outputs	None
Description	A manual voice referral process may be initiated by the card or by the issuer and the onRequestReferProcess method is triggered. The operator should call the bank to ask for the referral approval. The attendant may manually override the referral process and may accept or decline the transaction without performing a referral.
See also	onRequestReferProcess

cancelReferProcess (deprecated)

Signature	- (void)cancelReferProcess
Inputs	None
Outputs	None
Description	Abort the EMV transaction
See also	onRequestReferProcess

sendOnlineProcessResult

Signature	- (void)sendOnlineProcessResult: (NSString *)tlv
Inputs	tlv: The TLV message data of online processing result.
Outputs	None
Description	Send back the online process result to the ICC.
See also	onRequestOnlineProcess

Delegate Methods Reference

onBatteryLow

Signature	- (void)onBatteryLow: (BatteryStatus) batteryStatus
Inputs	batteryStatus: enum of the battery status
Outputs	None
Description	EMVSwipe is waken up by an audio command and is powered up and is ready for operations. Enum BatteryStatus can be BatteryStatus_Low BatteryStatus_CriticallyLow When battery is low, the operation can still be completed. When battery is critically low, the operation cannot be completed.
See also	

onDeviceHere

Signature	-(void) onDeviceHere: (BOOL) isHere
Inputs	isHere: YES - Presence of the EmvSwipe. NO - There is no device plugged in or another audio device is plugged in
Outputs	None
Description	Response to isDeviceHere query
See also	

onDevicePlugged

Signature	- (void)onDevicePlugged
Inputs	None
Outputs	None
Description	A device is plugged in.
See also	onDeviceUnplugged

onDeviceUnplugged

Signature	- (void)onDeviceUnplugged
Inputs	None
Outputs	None
Description	A device is unplugged.
See also	onDevicePlugged

onNoDeviceDetected

Signature	- (void)onNoDeviceDetected;
Inputs	None
Outputs	None
Description	No EmvSwipe is detected.
See also	

onWaitingForCard

Signature	- (void)onWaitingForCard
Inputs	None
Outputs	None
Description	Triggered in response to checkCard
See also	

onReturnKsn

Signature	- (void)onReturnKsn: (NSString *) ksnDict
Inputs	ksnDict: IDs and KSNs of the device in hexadecimal format
Outputs	None
Description	Return KSN in response to getKsn **NSDicitonary Keys** uid: Unique identifier of the crypto-processor trackKsn: KSN used for encryption of track data emvKsn: KSN used for encryption of EMV data, including ATR response, APDU data and other online messages. pinKsn: KSN used for encryption of online PIN
See also	getKsn

onReturnDeviceInfo

Signature	- (void) onReturnDeviceInfo(NSDictionary *) deviceInfoDict
Inputs	deviceInfoDict: device data table
Outputs	None
Description	Return device info in response to getDeviceInfo **NSDicitonary Keys** firmwareVersion: version of the firmware bootLoaderVersion: the version of the bootloader hardwareVersion: version of the hardware batteryLevel: 4 digits value of the voltage of the battery batteryPercentage: Integer of battery percentage e.g. “100”, “99” isCharging: 1 – is charging, 0 – no charging isUsbConnected: 1 – Connected, 0 – Not connected. isSupportedTrack1: 1 – Supported, 0 – not supported isSupportedTrack2: 1 – Supported, 0 – not supported isSupportedTrack3: 1 – Supported, 0 – not supported isSupportedNfc: 1 – Supported, 0 – not supported
See also	getDeviceInfo

onReturnReadTerminalSettingResult

Signature	-(void)onReturnReadTerminalSettingResult: (TerminalSettingStatus)status tagValue:(NSString *)tagValue
Inputs	status: Indicates status of operation tagValue: the value of the corresponding tag
Outputs	None
Description	Return EMV terminal setting in response to readTerminalSetting The enum TerminalSettingStatus is defined as below: typedef enum { TerminalSettingStatus_Success, TerminalSettingStatus_InvalidTlvFormat, TerminalSettingStatus_TagNotFound, TerminalSettingStatus_IncorrectLength } TerminalSettingStatus;
See also	readTerminalSetting

onReturnUpdateTerminalSettingResult

Signature	-(void)onReturnUpdateTerminalSettingResult: (TerminalSettingStatus)status
Inputs	status: Indicates status of operation
Outputs	None
Description	Return result in response to updateTerminalSetting The enum TerminalSettingStatus is defined as below: typedef enum { TerminalSettingStatus_Success, TerminalSettingStatus_InvalidTlvFormat, TerminalSettingStatus_TagNotFound, TerminalSettingStatus_IncorrectLength } TerminalSettingStatus;
See also	updateTerminalSetting

onReturnCheckCardResult

Signature	- (void)onReturnCheckCardResult:(CheckCardResult)result cardDataDict:(NSDictionary *)cardDataDict;
Inputs	result: Enum to show the card status cardDataDict: card data read by magnetic card reader, if a card swipe is captured successfully **NSDicitonary Keys:** *formatID* – Format ID PAN – Full PAN (optional) *maskedPAN* – Masked PAN *cardholderName* – cardholder name *expiryDate* – Expiry Date of the card *serviceCode* – 3-digit service code *ksn* - Key Serial Number for track data encryption. *encTrack1* - Encrypted Track 1 in HEX string. *encTrack2* - Encrypted Track 2in HEX string. *encTrack3*- Encrypted Track 3in HEX string. *encTracks* - Encrypted tracks in HEX string. *track1Length* – length of track 1 *track2Length* – length of track 2 *track3Length* – length of track 3 *partialTrack* – part of track 1 *trackEncoding*- Tell if track2 and Track3 data are packed in “ASCII” or “BCD” *finalMessage – proprietary data in FID61 devices*
Outputs	None
Description	Return the status of the checkCard command and also the card swipe result, if present. Enum **CheckCard** can be CheckCardResult_NoCard, CheckCardResult_InsertedCard, CheckCardResult_NotIccCard, CheckCardResult_BadSwipe, CheckCardResult_SwipedCard, CheckCardResult_MagHeadFail, CheckCardResult_NoResponse, CheckCardResult_OnlyTrack2

	CheckCardResult_NFC_Track2 CheckCardResult_UseIccCard
See also	checkCard

onReturnCancelCheckCardResult

Signature	- (void)onReturnCancelCheckCardResult:(BOOL) isSuccess
Inputs	isSuccess
Outputs	None
Description	Respond to cancelCheckCard. If isSuccess is YES, the checkCard process has been stopped successfully.
See also	cancelCheckCard

onReturnEncryptPinResult

Signature	- (void)onReturnEncryptPinResult: (NSString *) epn ksn: (NSString *) ksn
Inputs	epb: encrypted PIN block ksn: KSN used for encryption of PIN Data
Outputs	None
Description	Respond to encryptPin.
See also	encryptPin

onReturnEncryptDataResult

Signature	- (void)onReturnEncryptDataResult: (NSString *) encryptedData ksn: (NSString *) ksn
Inputs	encrypted: encrypted Data ksn: KSN used for encryption of PIN Data
Outputs	None
Description	Respond to encryptData.
See also	encryptData

onReturnPowerOnIccResult

Signature	- (void)onReturnPowerOnIccResult:(BOOL) isSuccess ksn: (NSString *) ksn atr: (NSString *) atr atrLength: (int) atrLength
Inputs	isSuccess: YES – success, No – failure ksn: EMV KSN used for encryption of ATR data and APDU data. If ksn is all FF, then ATR and APDU data are not encrypted. atr: data returned in ATR atrLength: length of the ATR data.
Outputs	None
Description	Respond to powerOnIcc.
See also	powerOnIcc

onReturnPowerOffIccResult

Signature	- (void)onReturnPowerOffIccResult:(BOOL) isSuccess
Inputs	isSuccess: YES – success, No - failure
Outputs	None
Description	Respond to powerOffIcc.
See also	powerOffIcc

onReturnApduResult

Signature	- (void)onReturnApduResult:(BOOL) isSuccess apdu: (NSString *) apdu apduLength: (int) apduLength
Inputs	isSuccess: YES – success, No – failure apdu: data returned apduLength: length of the apdu data
Outputs	None
Description	Return data in response to the level 1 EMV method sendApdu. If the apdu data are encrypted, the KSN returned after the powerOnIcc command is used for encryption.
See also	sendApdu

onReturnApduResultWithPkcs7Padding

Signature	- (void)onReturnApduResult:(BOOL) isSuccess apdu: (NSString *) apdu
Inputs	isSuccess: YES – success, No – failure apdu: data returned
Outputs	None
Description	Return data in response to the level 1 EMV method sendApduWithPkcs7. If the apdu data are encrypted, the KSN returned after the powerOnIcc command is used for encryption.
See also	sendApduWithPkcs7Padding

onReturnViposExchangeApduResult

Signature	- (void) onReturnViposExchangeApduResult: (NSString *)apdu
Inputs	apdu: data returned
Outputs	None
Description	Return APDU data.
See also	viposExchangeApdu

onReturnViposBatchExchangeApduResult

Signature	- (void) onReturnViposBatchExchangeApduResult: (NSString *)apdu
Inputs	**NSDictionary Keys** NSNumber Tag number. Refer to Q/CUP 037.2.4-2011 Section 11.8.3 **NSDictionary Values** NSArray of String In the order of 0: encryption key number for INIT_FOR_DESCRYPT 1: submitTime for INIT_FOR_DESCRYPT 2: DES_CRYPT APDU command
Outputs	None
Description	APDU exchange in batch for VI-POS. Response data are returned by onReturnViposBatchExchangeApduResult.
See also	viposBatchExchangeApdu

onReturnPowerOnNfcResult

Signature	- (void)onReturnPowerOnNfcResult☹BOOL)isSuccess response☹NSString *)response responseLength☹int)responseLength
Inputs	isSuccess: YES – success, NO – failure response: protocol dependent data returned upon power on. responseLength: length of the response data
Outputs	None
Description	Respond to powerOnNfc.
See also	powerOnNfc

onReturnPowerOffNfcResult

Signature	- (void)onReturnPowerOffNfcResult☹BOOL)isSuccess
Inputs	isSuccess: YES – success, NO – failure
Outputs	None
Description	Respond to powerOffNfc
See also	powerOffNfc

onReturnNfcDataResult

Signature	- (void)onReturnNfcDataResult☹NfcDataExchangeStatus)status data☹NSString *)data dataLength☹int)dataLength;
Inputs	NfcDataExchangeStatus: NfcDataExchangeStatus_Success, NfcDataExchangeStatus_NotYetPowerOn, NfcDataExchangeStatus_NoResponse data: data returned dataLength: length of the apdu data
Outputs	None
Description	Return data in response to the NFC data exchange.
See also	nfcDataExchange

onReturnEmvCardDataResult

Signature	- (void)onReturnEmvCardDataResult☹NSString *)tlv
Inputs	tlv: a list of card data in TLV format.
Outputs	None
Description	Respond to getEmvCardData. Information about the card is returned. This includes cardholder name (5F20), expiration date (5F24), masked PAN (custom tag C4), encrypted PAN (custom tag C5), KSN for encryption key used in encrypted PAN (custom tag C3)
See also	getEmvCardData.

onReturnStartEmvResult (Deprecated)

Signature	- (void)onReturnStartEmvResult☹StartEmvResult) result ksn: (NSString *) ksn
Inputs	result: enum of the startEmv result. Ksn: KSN of EMVSwipe
Outputs	None
Description	Return data in response to startEmv
See also	selectApplication

onRequestSelectApplication

Signature	- (void)onRequestSelectApplication☹NSArray *)applicationArray
Inputs	applicationArray: Array of applications found in EMV card
Outputs	None
Description	EmvSwipe is requesting user to select an application in the EMV process and a list of applications supported is returned.
See also	selectApplication

onRequestSetAmount

Signature	- (void) onRequestSetAmount
Inputs	None
Outputs	None
Description	EmvSwipe is requesting the amount. The app should prompt the user of the transaction amount.
See also	setAmount, cancelSetAmount

onRequestPinEntry (deprecated)

Signature	- (void) onRequestPinEntry
Inputs	None
Outputs	None
Description	EmvSwipe is requesting PIN entry in the EMV process.
See also	sendPinEntryResult, cancelPinEntry, bypassPinEntry

onRequestCheckServerConnectivity

Signature	- (void) onRequestCheckServerConnectivity
Inputs	None
Outputs	None
Description	EmvSwipe is requesting checking of the server connectivity in the EMV process.
See also	sendServerConnectivity

onRequestTerminalTime

Signature	- (void) onRequestTerminalTime
Inputs	None
Outputs	None
Description	EmvSwipe is requesting the current local time in the EMV process.
See also	sendTerminalTime

onRequestFinalConfirm

Signature	- (void)onRequestFinalConfirm
Inputs	None
Outputs	None
Description	A final confirm request for the operator to check the transaction information such as amount before processing the EMV transaction and Generate AC.
See also	sendFinalConfirmResult

onReturnTransactionResult

Signature	- (void)onReturnTransactionResult: (TransactionResult) result data:(NSDictionary *)data
Inputs	result: enum of transaction result data: data for the transaction **NSDictionary Keys and Values of data:** receiptData - receipt data in TLV format
Outputs	None
Description	Return the transaction result on an EMV transaction TransactionResult_Approved, TransactionResult_Terminated, TransactionResult_Declined, TransactionResult_SetAmountCancelOrTimeout, TransactionResult_CapkFail, TransactionResult_NotIcc, TransactionResult_CardBlocked, //Updated in 1.7.2 TransactionResult_DeviceError, //Updated in 1.7.2 TransactionResult_CardNotSupported, TransactionResult_MissingMandatoryData, TransactionResult_NoEmvApps, //Updated in 1.7.2 TransactionResult_InvalidIccData, TransactionResult_ConditionsOfUseNotSatisfied, TransactionResult_ApplicationBlocked //Added in 1.7.2
See also	

onReturnTransactionLog (deprecated)

Signature	- (void)onReturnTransationLog: (NSString *) tlv
Inputs	tlv: tag value results to be sent back to server for processing
Outputs	None
Description	Return the transaction data after completion of an EMV transaction.
See also	

onReturnReversalData

Signature	- (void)onReturnReversalData: (NSString *) tlv
Inputs	tlv: tag value results to be sent back to server for processing
Outputs	None
Description	Return the reversal data after completion of an EMV transaction. ***Note***: the data in tlv is in proprietary format for FID61 devices.
See also	

onReturnBatchData

Signature	- (void)onReturnBatchData: (NSString *) tlv
Inputs	tlv: tag value results to be sent back to server for processing
Outputs	None
Description	Return the batch data after completion of an EMV transaction. ***Note***: the data in tlv is in proprietary format for FID61 devices.
See also	

onRequestOnlineProcess

Signature	- (void)onRequestOnlineProcess: (NSString *) tlv
Inputs	tlv: tag value results to be sent back to server for processing
Outputs	None
Description	Return data for online processing. ***Note***: the data in tlv is in proprietary format for FID61 devices.
See also	sendOnlineProcessResult

onRequestAdviceProcess (deprecated)

Signature	- (void)onRequestAdviceProcess: (NSString *) tlv
Inputs	tlv: tag value results to be sent back to server for processing
Outputs	None
Description	Return data for advice processing.
See also	sendOnlineProcessResult

onRequestReferProcess (deprecated)

Signature	- (void)onRequestReferProcess: (NSString *) pan
Inputs	pan: the card number to be communicated to the voice authentication operator
Outputs	None
Description	A manual voice referral process may be initiated by the card or by the issuer and the onRequestReferProcess method is triggered. The operator should call the bank to ask for the referral approval. The attendant may manually override the referral process and may accept or decline the transaction without performing a referral.
See also	sendReferProcessResult

onRequestDisplayText

Signature	- (void)onRequestDisplayText: (DisplayText *)displayMessage
Inputs	displayMessage: enum of a display message
Outputs	None
Description	EmvSwipe has requested to display a message.
See also	onRequestClearDisplay

onRequestClearDisplay

Signature	- (void)onRequestClearDisplay
Inputs	None
Outputs	None
Description	EmvSwipe has requested to clear the display
See also	onRequestDisplayText

onRequestVerifyID

Signature	- (void)onRequestVerifyID:(NSString *)tlv;
Inputs	tlv: information about cardholder name in TLV format
Outputs	None
Description	An ID checking process for PBOC may be initiated by the card or by the terminal and the onRequestVerifyID method is triggered. The operator should check for the cardholder ID and respond by using sendVerifyIDResult function.
See also	sendVerifyIDResult

onError

Signature	- (void)onError:(ErrorType) errorType errorMessage:(NSString *)errorMessage;
Inputs	errorType: Enum of the error that occurs errorMessage: a message about the error
Outputs	None
Description	A generic method to report error
See also	See ErrorType Enumerations section

onInterrupted

Signature	- (void)onInterrupted
Inputs	None
Outputs	None
Description	The application is interrupted by the following: Incoming call, alarm, SMS and application switch to background
See also	

Enumeration

ErrorType

```
typedef enum {
        ErrorType_InvalidInput,
        ErrorType_InvalidInput_NotNumeric,
        ErrorType_InvalidInput_InputValueOutOfRange,
        ErrorType_InvalidInput_InvalidDataFormat,
        ErrorType_InvalidInput_NoAcceptAmountForThisTransactionType,
        ErrorType_InvalidInput_NotAcceptCashbackForThisTransactionType,

        ErrorType_DeviceReset,
        ErrorType_CommError,
        ErrorType_Unknown,

        ErrorType_AudioFailToStart,
        ErrorType_AudioNotYetStarted,
        ErrorType_IllegalStateException,
        ErrorType_CommandNotAvailable,                  //Added in 1.5.0
        ErrorType_AudioRecordingPermissionDenied,       //Added in 1.6.6
        ErrorType_BackgroundTimeout,                    //Added in 1.6.6
        ErrorType_AudioFailToStart_OtherAudioIsPlaying  //Added in 1.9.1
} ErrorType;
```

DisplayText

```
typedef enum {
        DisplayText_AMOUNT,              //Deprecated since 1.3.0
        DisplayText_AMOUNT_OK_OR_NOT,
        DisplayText_APPROVED,
        DisplayText_CALL_YOUR_BANK,
        DisplayText_CANCEL_OR_ENTER,
        DisplayText_CARD_ERROR,
        DisplayText_DECLINED,
        DisplayText_ENTER_AMOUNT,    //Deprecated since 1.3.0
        DisplayText_ENTER_PIN,         //Deprecated since 1.3.0
        DisplayText_INCORRECT_PIN,
        DisplayText_INSERT_CARD,
        DisplayText_NOT_ACCEPTED,
        DisplayText_PIN_OK,
        DisplayText_PLEASE_WAIT,
        DisplayText_PROCESSING_ERROR,
        DisplayText_REMOVE_CARD,
        DisplayText_USE_CHIP_READER,
        DisplayText_USE_MAG_STRIPE,
        DisplayText_TRY_AGAIN,
        DisplayText_REFER_TO_YOUR_PAYMENT_DEVICE,
        DisplayText_TRANSACTION_TERMINATED,
        DisplayText_TRY_ANOTHER_INTERFACE,
        DisplayText_ONLINE_REQUIRED,
        DisplayText_PROCESSING,
        DisplayText_WELCOME,
        DisplayText_PRESENT_ONLY_ONE_CARD,
        DisplayText_LAST_PIN_TRY,
        DisplayText_CAPK_LOADING_FAILED
} DisplayText;
```