CurrencyPay - Mobile

Software Development Kit (SDK)

iOS Developer Guide

Version 2.0

Revision Date: 10.01.2019

TABLE OF CONTENTS

Overview

The CurrencyPay SDK for iOS consists of embedded frameworks that allow a user to access credit cards readers and perform payment processing requests. Using this SDK one can create a full mobile App that handles card swipes and the process those transactions against a designated payment gateway.

The CurrencyPay SDK is distributed using an industry standard product CocoaPods, which is a dependency and distribution manager. This product fully automates the distribution of software, and eases the task of release management, and product integration.

Requirements

The CurrencyPay SDK was designed and developed to leverage the following requirements:

- iOS 12 or Later
- XCode IDE
- Bluetooth and Audio Card Readers (or Manual Card Entry)
- Network accessibility (WIFI or Cellular)

Setting Up the SDK

Experienced using CocoaPods with XCode

If you are experienced with using CocoaPods then the following steps will be very familiar.

- Add the following to a podfile for your App.

  target : 'YourProjectApp' do
  pod 'CurrencyPaySDK'
  end

- Then execute from a command line within Terminal:

  pod install

  After running this command line request, you will see the following messages:

```
Pod installation complete! There are x dependencies from the Podfile and x total pods
installed.
Analyzing dependencies
Downloading dependencies
Installing CurrencyPaySDK (2.0.0)
```

That's it. The SDK has been added to your project and you can now access the services of the SDK.

New to CocoaPods with XCode

You will find using CocoaPods much simpler to use than cloning libraries manually integrating the products, especially since integrating frameworks of an SDK can be tricky and complex.

The CocoaPods website provides step-by-step instructions on how to setup CocoaPods on your computer, and then how to integrate into your App.

- First you need to set up CocoaPods on your computer. The following link explains CocoaPods, and has a short video to help in your understanding. You will see that CocoaPods is easy to install, and use.

  https://guides.cocoapods.org/using/getting-started.html

- After CocoaPods is installed you next need to connect CocoaPods to your iOS project. This connection is made through a podfile. Here is a sample of a podfile as well as a link explaining how they work. Basically, you just have a plain text file named: **podfile** in the directory where your app resides. The contents of that file should be very similar to this:

```
# The following line should not change
source 'https://github.com/CocoaPods/Specs.git'

workspace       'YourProjectWS'
project         'path/YourProject.xcodeproj'

platform        :ios, '12.0'
use_frameworks!

target :        'YourProjectApp' do
                pod 'CurrencyPaySDK'
end
```

  That's it, your podfile is complete and ready to be used. Basically, you tell CocoaPods what and where your app resides, and what pod you want to use with your app. In this case, CurrencyPay. CocoaPods handles the rest. We recommend placing your podfile in your mobile app project directory.

- Then execute from a command line within Terminal application. Be sure you are in the directory where the podfile exist:

  pod install

  After running this command line request, you will see the following messages:

```
Pod installation complete! There are x dependencies from the Podfile and x total pods
installed.
Analyzing dependencies
Downloading dependencies
Installing CurrencyPaySDK (2.0.0)
```

  That's it. The SDK has been added to your project and you can now access the services of the SDK. You can now open your project by opening the newly created project Workspace file, and not the project file itself. You can proceed to the usage instructions for the CurrencyPay SDK.

  To better understand podfiles and the integration with iOS apps refer to the following link:

  https://guides.cocoapods.org/using/using-cocoapods.html

  CocoaPods provides extensive documentation at this site that goes beyond what is covered in this document.

Using the CurrencyPay SDK

There are two primary frameworks that make up the CurrencyPay SDK. The first is PayCard which handles card reader management and communications, and the second is PayCore which handles the financial transaction processing.

PayCard Services

Note that for the classes and methods shown Xcode help is available. By holding the Option Key and clicking on the Class / Method a help popup screen will provide a description, the input parameters (if any), and the returned values. Most complex methods where the response may not be immediate may return the results via a callback and will be shown below.

Class	Method / Property (p)	Purpose	Delegate / Callback
PayCardRDRMgr	sharedInstance (p)	Access to PayCardRDRMgr singleton class	delegate: Set to the class that will handle method specific callbacks.
	startPayCardRDRMgr	Start Reader Services.	N/A
	scanForReaders	Scans for available Bluetooth readers.	didFindRDRDevices didReceiveBTScanTimeOut
	connectBTReader	Connects a Bluetooth device by Device ID.	didSuccessfulBTConnect didReceiveCardReaderConnectionFailed
	connectAudioReader	Connects to an Audio Card Reader that is plugged into the audio port.	didReceiveAudioConnectedNotice didReceiveCardReaderConnectionFailed
	disconnectAudioReader	Disconnects an Audio Card Reader from transaction processing capability (even if still plugged in)	didReceiveAudioDisconnectedNotice
	detectConnectionType	Detects and returns the type of reader connection: • Audio • Bluetooth • USB (not Supported) • None	Returned value type enum: PayCardConnectionMode
	detectAudioReaderChange	This notifies the SDK to listen for changes to the Audio Reader, such as device being plugged in or unplugged.	Listener Callbacks: didReceiveAudioAttachedNotice didReceiveAudioDisconnectedNotice **Note**: Some reader devices may not offer this capability.
	isAudioReaderAttached	Check if Audio Reader Attached. This only checks if there is a device plugged into audio port. This does not check if reader services active (see isAudioreaderConnected)	Returned value type Boolean: True = Reader Attached False = Reader Not Attached
	isAudioreaderConnected	Check if Audio Reader is attached and has card transaction processing capabilities active.	Returned value type Boolean: True = Reader Connected False = Reader Not Connected

Class	Method / Property (p)	Purpose	Delegate / Callback
PayCardRDRMgr	isBTReaderConnected	Check if a Bluetooth reader is connected.	Returned value type Boolean: True = Reader Connected False = Reader Not Connected
	disconnectBTReader	Disconnect the Bluetooth reader device from mobile device and transaction processing capability.	didReceiveBTDisconnect
	getDeviceData	Retrieves device specific information about card reader in use.	didReturnDeviceData **Note:** Not all device services offer access to this information and the information may vary.
PayCardMaster	sharedInstance (p)	Access to PayCardMaster singleton class	delegate: Set to the class that will handle method specific callbacks.
	doReadCard	Stage 1: Request PayCard initiate the card reading process	didReceiveReaderModeUpdate: • Returns an enum value found in PayCardDeviceMode that tells app how to proceed. Example “Swipe Card” didReceiveCardIssue
		Stage 2: Card is Swiped (or other action taken)	didReceiveSwipeSuccess didReceiveReaderModeUpdate
		Stage 3: (EMV) Do Card Transaction Confirmation	didReceiveReaderModeUpdate: • DeviceMode_Confirm • See doCardConfirm method
		Stage 4: (EMV) Complete transaction on Gateway to complete EMV transaction request.	requestForHostEMVProcess: • See responseFromHostEMVProcess method
	doDetermineCardType	A utility to take the card number and return the card brand such as Visa or MasterCard.	Returned value type String: Supported Card Brand Name.
	doCardConfirm (EMV)	Performs an EMV transaction confirmation.	requestForHostEMVProcess
	responseFromHostEMVProcess	Sends final EMV Tag (8A, 91, 71, 72) data to reader to complete transaction.	didCompleteEMVCardTransaction didReceiveFinalEMVBatchData
	- General Utility Callback	Message Capture. Helpful during development and debugging.	didReceiveMessageToDisplay

PayCore Services

Note that for the classes and methods shown, Xcode help is available. By holding the Option Key and clicking on the Class / Method a help popup screen will provide a description, the input parameters (if any), and the returned values. Most complex methods where the response may not be immediate may return the results via a callback and will be shown below, otherwise the response is in a closure (block) completion handler.

Class	Method / Property (p)	Purpose	Delegate / Callback
PayCoreMaster	sharedInstance (p)	Access to PayCoreMaster singleton class	delegate: Set to the class that will handle method specific callbacks.
	validateLoginCredentials	Login Authentication	Response in completion handler
	verifySessionKey	Verifies the session is still valid	Response in completion handler
	getMerchantId	Retrieves the Merchant ID of signed in Merchant	Response in completion handler
	processTransaction	Initiate payment transaction with Host-API gateway processing.	Response in completion handler
	addSignatureToTransaction	Allows the adding a signature to the active payment transaction	Response in completion handler
	retrieveTransactionsFor	Retrieves transactions by merchant.	Response in completion handler
	processRefund	Request a refund on original Sale transaction.	Response in completion handler
	retrieveSubsequentTransactionsFor	Retrieves subsequent transactions related to the original Sale transaction.	Response in completion handler
	retrieveChargebacksFor	Retrieve chargebacks against an original Sale	Response in completion handler
	checkTXNRefundEligible	Determines if a Sale transaction is refund eligible.	Response in completion handler
PayCoreRDRTransMgr	sharedInstance (p)	Access to PayCoreRDRTransMgr singleton class	delegate: Set to the class that will handle method specific callbacks.
	doCardRDRTransaction	This is the primary point to initiate payment transaction processing that consists of data from Card Reader.	didReceiveTransactionSuccess didReceiveTransactionFailed

Feedback and Support

To provide feedback on this document and any of the specifications contained within, contact us at: https://www.currencypay.com/contact-us/

Document Revision History

Revision Date	Author	Revision Description
01.29.2019	Mobile Development	Original Document
10.01.2019	Mobile Development	SDK Version 2.0 Revisions