



Integrating the Pay360 Mobile SDK

March 2022

Copyright © 2021 Capita plc, all rights reserved

This document is intended to act as a reference guide to Capita's products.

Capita however, makes no representation or warranty, express, implied, or statutory, with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency or usefulness of the information contained herein.

The information in this manual is subject to change without prior notices and should not be constructed as a commitment by Capita.

No part of this publication may be copied or distributed, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, manual, or disclosed to third parties without the express written permission of Capita.

Pay360 by Capita is a trading name of Pay360 Limited



Introduction

The Pay360 Mobile SDK provides a simple integration to process payments on the Pay360 payment platform via a Mobile App. This document will help you with the process of integrating to the SDK.

There are two SDK's developed by Pay360 for the iOS and Android Mobile apps.

The Pay360 Mobile SDK has been developed to integrate into one of three Pay360 payment platforms. Before integrating the SDK into your Mobile App, you will need to register for an account onto one of these payment platforms. For further information about the payment platform integration please refer to the following:

- Advanced Payments (AP) – Integrating the Pay360 Mobile SDK, Advanced Payments (AP) - (Mobile SDK – Integrating AP.pdf)
- Evolve – Integrating the Pay360 Mobile SDK, Evolve - (Mobile SDK – Integrating Evolve.pdf)
- Secure Bureau Service (SBS) – Integrating the Pay360 Mobile SDK, Secure Bureau Service (SBS) - (Mobile SDK – Integrating SBS.pdf)

You should also refer to:

- Pay360 Mobile SDK PCI Compliance Statement (Mobile SDK – PCI Compliance Statement.pdf)

Development guidelines

All app developers will need to take into consideration the following features when building an app which integrates with the Pay360 Mobile SDK.

- Root/Jailbreak detection
- Strong User IDs
- Secure Cookie Storage
- Developers will be responsible for SSL pinning against your app

The mobile SDK is available via GitHub at:

- Android: <https://github.com/pay360/Pay360-Android-Mobile-SDK>
- iOS: <https://github.com/pay360/Pay360-iOS-Mobile-SDK>

Development pre-requisite

iOS

The Pay360 iOS SDK enables you to incorporate our payment-handling capabilities into your mobile applications for Apple devices.

This SDK is available for native iOS applications developed in Objective-C or Swift, as well as other hybrid mobile frameworks like React Native or Xamarin.

Prerequisites for running/building a project

- iOS 9+
- XCode 10+

Android

This is Android mobile SDK for you to integrate Pay360 to your Android mobile app. It is available for native Android applications (Java/Kotlin) and other hybrid mobile framework for example React Native, Xamarin, etc.

Prerequisites for running/building the project

At minimum Android 4.1 (API level 14)

Processing pre-requisite

When processing a payment or verification in the Mobile SDK, your chosen payment platform integration will provide the required data. This will consist of the following:

	Referred to as in Advanced Payment	Referred to as in Evolve	Referred to as in SBS	
Integrator Identifier {{integrator_id}}	instId	merchantId	merchantId	Used to identify <i>you</i> as the merchant integrating into the Mobile SDK
Client Access Token {{client_access_token}}	clientToken	sessionId	sessionId	Used as a one-time token to process a single payment or verification request

Authorisation

The Mobile SDK must be initialised with a valid Client Access Token to process a payment. Once obtained, the following implementation should be performed.

Create the credentials

iOS

```
let credentials = new PPOCredentials()

    credentials.token = {{client_access_token}}

    credentials.installationId = {{integrator_id}}

    credentials.environment = {{environment}}
```

Android

```
PPOCredentials credentials = new PPOCredentials({{client_access_token}}, {{integrator_id}}, {{environment}})
```

The credentials object will be used in the next section to process a payment.

Card Payment

The Mobile SDK supports the processing of card payments in the following ways:

- Card number and payment data provided by the payer to process a payment
- Stored token from a previous payment or verification supplied to process a payment

The integration for each route is described below.

Payment using card number

The processing of making a payment with card number and payment details is as follows.

- Mobile App validates the card number
- Mobile App creates a payment object and populates it with **card** data

- Mobile App submits the payment
- Mobile SDK submits the payment data for processing
- (conditional) Mobile SDK opens a WebView to process the 3DS challenge
- Mobile SDK resumes the payment after 3DS challenge is complete
- Mobile SDK returns the payment response
- Mobile App handles the payment response

Validate card number

iOS

```
let luhnCheck = PPOluhn.init(self)

    luhnCheck.validateCreditCardNumber(cardNumber)
```

Android

```
boolean isCardValid = PPOluhn.validateCreditCardNumber(cardNumber);
```

Create Payment

iOS

```
let payment = PPOPayment.init(self)

    payment.credentials = credential

    payment.card = card

    payment.address = billingAddress

    payment.customer = customer

    payment.providerId = providerId
```

Android

```
PPOPayment payment = new PPOPayment(this, this)

    .setCredentials(this.credentials)
```

```

.setCard(card)

.setBillingAddress(billingAddress)

.setCustomer(customer)

.setProviderId(providerId);

```

The key attributes within the request are documented below.

credentials	Object created in the “Create the credentials” section
card	Object containing the card payment data
billingAddress	(Optional) Object containing the billing address Note: some of the billing address fields are mandatory to perform 3DS v2.1. Therefore, if the data has not been previously provided, it must be passed into the Mobile SDK.
customer	(Optional) Object containing the customer information
providerId	This field indicate the payment platform to be used for processing the payment. Can be one of the following values. <ul style="list-style-type: none"> • “AP” • “EVOLVE” • “SBS” If not supplied, then “AP” will be set as default.

Submit Payment

iOS

```

self?.payment!.processGuestPayment { (Data: PPOPaymentResponse) in

    print(Data.transactionId)

} failure: { (error) in

    print(error)

}

```

The payment response is handled by the result of calling the method above. Your Mobile App must implement appropriate code to handle the different states.

Android

```
this.payment.processGuestPayment();
```

The payment response must be handled by a delegate function

```
@Override
public void cardPaymentProceedWithSuccess(PPOPaymentType paymentType, PPOPaymentResponse response) {
}

@Override
public void cardPaymentProceedWithFailure(PPOPaymentType paymentType, String reason) {
}
```

Payment using stored card

The processing of making a payment using a stored card is as follows.

- Mobile App creates a payment object and populates it with **cardToken** data
- Mobile App submits the payment
- Mobile SDK submits the payment data for processing
- (conditional) Mobile SDK opens a WebView to process the 3DS challenge
- Mobile SDK resumes the payment after 3DS challenge is complete
- Mobile SDK returns the payment response
- Mobile App handles the payment response

The implementation is the same as “Payment using card number” above. However, instead of populating the payment object with card data, this should be populated with **cardToken** data.

Payment Objects

Creating a payment object enables you to set the following attributes.

Card

iOS

```
let card = PPOCard()  
  
    card.pan = is3DS ? "9903000000005131" : "9903000000000017"  
  
    card.cvv = "123"  
  
    card.expiry = "0117"  
  
    card.cardHolderName = "John Smith"
```

Android

```
PPOCard card = new PPOCard()  
  
    .setPan(is3DS ? "9903000000005131" : "9903000000000017")  
  
    .setCv2("123")  
  
    .setExpiryDate("0117")  
  
    .setCardHolderName("John Smith");
```

Card Token

iOS

```
let card = PPOCard()  
  
    card.token = "CARD TOKEN"  
  
    card.cvv = "123"  
  
    card.lastFour = "LAST 4 DIGITS OF THE CARD NUMBER"
```

Android

```
PPOCard card = new PPOCard()  
  
    .setToken("CARD TOKEN")  
  
    .setCv2("123")
```

```
.setLastFour("4444");
```

Billing Address

iOS

```
let billingAddress = PPOBillingAddress()  
  
    billingAddress.line1 = "YOUR ADDRESS LINE 1"  
    billingAddress.line2 = " YOUR ADDRESS LINE 2"  
    billingAddress.line3 = " YOUR ADDRESS LINE 3"  
    billingAddress.line4 = " YOUR ADDRESS LINE 4"  
  
    billingAddress.city = "YOUR CITY"  
    billingAddress.region = "YOUR REGION"  
    billingAddress.postcode = "YOUR POSTCODE"  
    billingAddress.countryCode = "GBR"
```

Android

```
PPOBillingAddress address = new PPOBillingAddress()  
  
    .setLine1("YOUR ADDRESS LINE 1")  
    .setLine2(" YOUR ADDRESS LINE 2")  
    .setLine3(" YOUR ADDRESS LINE 3")  
    .setLine4(" YOUR ADDRESS LINE 4")  
  
    .setCity("YOUR CITY")  
    .setRegion("YOUR REGION")  
    .setPostcode("YOUR POSTCODE")  
    .setCountryCode("GBR");
```

Customer

iOS

```
let customer = PPOCustomer()
```

```
customer.email = "YOUR EMAIL ADDRESS"

customer.dateOfBirth = "120479"

customer.telephone = "YOUR TELEPHONE NUMBER"

customer.merchantRef = "YOUR CUSTOMER REFERENCE"
```

Android

```
PPOCustomer customer = new PPOCustomer()

    .setEmail("YOUR EMAIL ADDRESS")

    .setDateOfBirthday("120479")

    .setTelephone("YOUR TELEPHONE")

    .setMerchantRef("YOUR CUSTOMER REFERENCE");
```

Custom Fields

iOS

```
let customFieldSDKVersion = PPOCustomField()

    customFieldSDKVersion.name = "sdkVersion"

    customFieldSDKVersion.value = "10.1"

let customFieldMerchantAppName = PPOCustomField()

    customFieldMerchantAppName.name = "merchantAppName"

    customFieldMerchantAppName.value = "Test App"

let customFieldMerchantAppVersion = PPOCustomField()

    customFieldMerchantAppVersion.name = "merchantAppVersion"

    customFieldMerchantAppVersion.value = "0.1"
```

```
let customFieldsSet = NSMutableSet()

customFieldsSet.addObjects(from: [customFieldSDKVersion, customFieldMerchantAppName, customFieldMerchantAppVersion])
```

Card Verification

The Mobile SDK supports the verification of a card which returns a token that can be used to process subsequent payments.

The processing of verifying a card number is as follows.

- Mobile App validates the card number
- Mobile App creates a payment object and populates it with card data
- Mobile App submits the verification
- Mobile SDK submits the payment data for verification
- (conditional) Mobile SDK opens a WebView to process the 3DS challenge
- Mobile SDK resumes the verification after 3DS challenge is complete
- Mobile SDK returns the verification response
- Mobile App handles the verification response

The implementation to validate the card number and create a payment object is the same as “Payment using card number” above.

The implementation to Submit the verification is described below.

Submit the verification

iOS

```
self?.payment!.processVerifyRequest { (Data: PPOPaymentResponse) in

    print(Data.transactionId)

} failure: { (error) in

    print(error)

}
```

The payment response is handled by the result of calling the method above. Your Mobile App must implement appropriate code to handle the different states.

Android

```
this.payment.processVerifyRequest();
```

The payment response must be handled by a delegate function

```
@Override
public void cardPaymentProceedWithSuccess(PPOPaymentType paymentType, PPOPaymentResponse response) {
}

@Override
public void cardPaymentProceedWithFailure(PPOPaymentType paymentType, String reason) {
}
```