amazon payment services

Android Integration Guide

Document Version: 3.7

April, 2021

Contents

1 About this document 5

1.1 Intended audience 5
1.2 Note regarding PayFort / FORT 5

2 Before you start the integration 6

3 Mobile SDK transaction workflow 7

4 Installing the Mobile SDK 8

4.1 Installing the Android Mobile SDK 8
4.2 Setting Android OS permissions 8

5 Creating a mobile SDK token 9

5.1 Android mobile SDK token URLs 9
5.2 Submitting token request parameters 9
5.2.1 Android Mobile SDK Token Request Parameters 10
5.2.2 Android Mobile SDK Token Response Parameters 11

6 Processing transactions with the Android SDK 12

6.1 Two routes for in-app payment processing 12
6.2 Standard checkout implementation 13
6.3 Android Mobile SDK Device ID Value 16
6.4 Android mobile SDK parameters 16
6.4.1 Request parameters 16
6.4.2 Response parameters 20

7 Standard checkout UI sample code 23

8 Customizing the payment UI 25

8.1 Customizing the Android Mobile SDK Payment Layout 25
8.2 Design Customization Codes: 26

9 Custom-coding a payment processing UI 28
9.1 Stage 1: Generate an SDK token 28
9.2 Stage 2: creating the card components 28
Table of components 28
9.3 Stage 3: Initiate the actual payment. 30
9.4 Sample Code 31
10 Using Direct Pay 32
10.1 Direct Pay Sample code: 32
11 Amazon Payment Services Android SDK transaction feedback 34
11.1 Receiving transaction feedback 34
11.2 Registering Transaction Feedback URLs 34
11.3 Transaction Feedback Implementation 35
12 Validate API call 36
13 Migrating the previous version 38
14 Appendix: About the Software 41
14.1 About this software 41
14.2 Supported Platforms 41
14.3 Localization 41
14.4 Screen Orientation 41
14.5 Supported Payment Methods 41
14.6 Supported Payment Options 41

Contact Us

integration-ps@amazon.com
https://paymentservices.amazon.com

1 About this document

This document describes our Mobile SDK for Android and includes information on how to integrate it with your mobile application.

1.1 Intended audience

This document was created for Android developers that integrate the Amazon Payment Services Android mobile SDK with their merchant application.

1.2 Note regarding PayFort / FORT

Amazon Payment Services is the new name for PayFort. PayFort is a leading provider of payment processing services that was acquired by Amazon in 2017.

Throughout this section, and in our API reference and SDK guides, you will see references to PayFort. You may also see references to Fort or FORT.

We continue to use PayFort and Fort in our documentation for the simple reason that the code that powers Amazon Payment Services still contains references to PayFort.

To ensure ongoing stability, and to minimize the development overhead for our customers, we are slowly but steadily changing references to PayFort across our core code and our documentation.

In the meantime, when you see PayFort or Fort, you can safely assume that we are referring to Amazon Payment Services features and benefits.

2 Before you start the integration

Read through the following steps first to understand how the integration process works. This will help you to understand why these steps are required and why you need to follow a specific sequence.

Step 1: Access your test account

You need to make sure that you have access to a test account with Amazon Payment Services. It is a full test environment that allows you to fully simulate transactions.

Step 2: Choose between a standardized or custom payment UI

The Amazon Payment Services Android SDK provides you with a standard payment UI that you can use to quickly integrate in-app payments. The standard UI offers limited customizability.

Alternatively, you can choose to build your own payment UI using Amazon Payment Services Android SDK building blocks, we describe this in the section on custom-coding a payment processing UI.

Step 3: Make sure that you are using the correct integration type

Prior to building the integration, you need to make sure that you are selecting and using the proper parameters in the API calls as per the required integration type. All the mandatory parameters are mentioned under every section in the API document

Step 4: Install the Amazon Payment Services Android SDK in your developer environment

You need to add the repositories to your build file and add the SDK to your build.gradle, also setting the correct Android permissions.

Step 5: Create a test transaction request

You need to create a test transaction request. Processing a valid API request depends on transaction parameters included, you need to check the documentation and read every parameter possible to reduce the errors in processing the transaction.

Step 6: Process a transaction response

After every payment, Amazon Payment Services returns the transaction response on the URL configured in your account under **Technical Settings**, **Channel Configuration**.

For more details check the transaction feedback instructions. You need to validate the response parameters returned on this URL by calculating the signature for the response parameters using the SHA response phrase configured in your account under security settings.

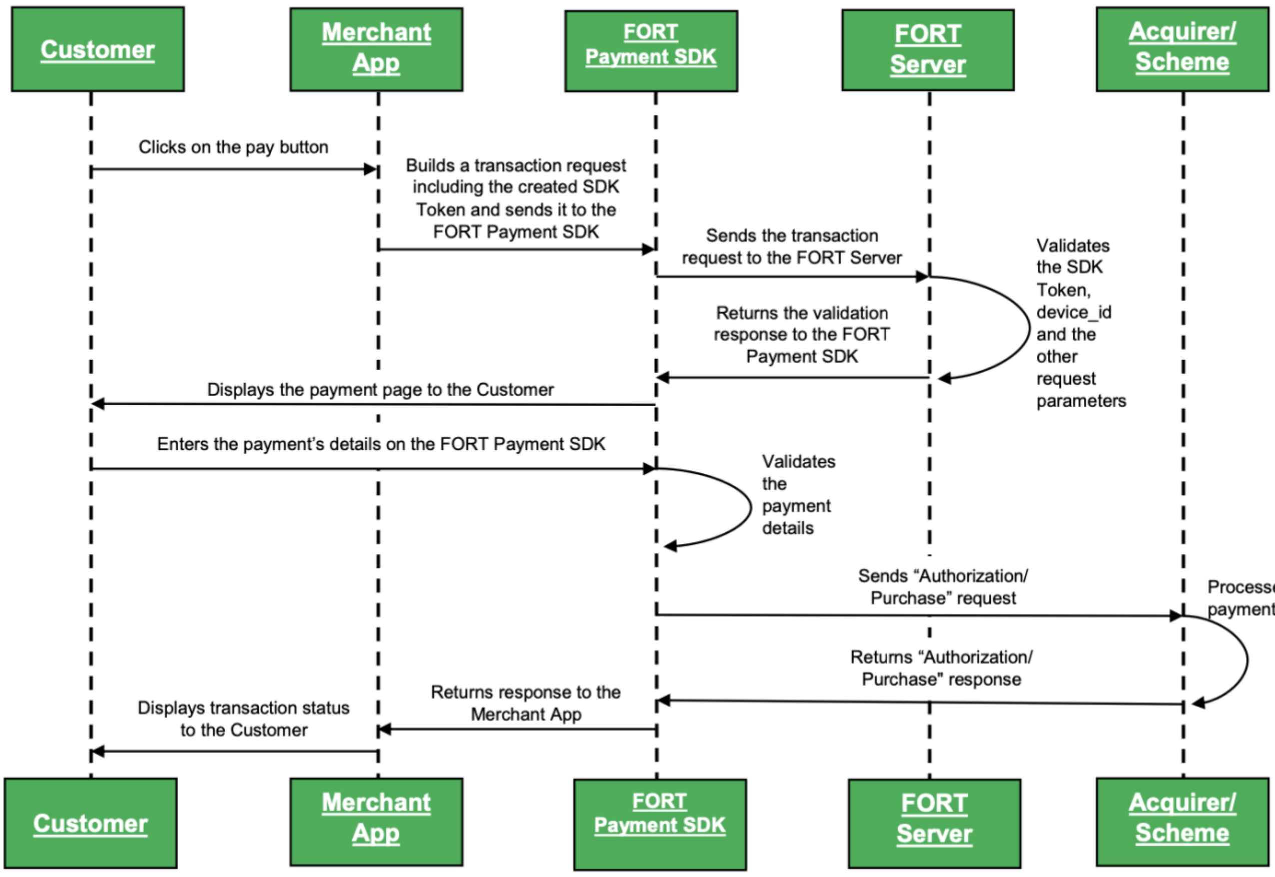
Step 7: Test and Go Live

You can use our test card numbers to test your integration and simulate your test cases. The Amazon Payment Services team may need to test your integration before going live to assure your application integration.

3 Mobile SDK transaction workflow

Below, we describe the transaction workflow when you process a payment using the Amazon Payment Service Android Mobile SDK.

1. Your customer clicks on the Pay button in your app.
2. Your merchant system (back-end) generates a mobile SDK token using the Amazon Payment Services API
3. Your app passes the parameters, including the SDK token, to the Android mobile SDK
4. The Android mobile SDK starts a secure connection and sends the transaction request to the Amazon Payment Services server to be validated.
5. The Amazon Payment Services API validates the SDK token, device_ID and other request parameters and returns the validation response the Android SDK.
6. Assuming validation is passed your merchant app displays a payment page to the customer.
7. Your customer enters their payment details on the payment page in the Android SDK prompt on their device.
8. The Amazon Payment Services Android SDK validates your customer's details and sends a transaction (Authorization or Purchase) request to the relevant payment processor and issuing bank.
9. The payment processor processes the transaction request and returns a transaction response to the Android SDK
10. The Amazon Payment Services Android SDK returns the transaction response to your merchant app
11. Your merchant app displays the response to your customer


Customer
Merchant App
FORT Payment SDK
FORT Server
Acquirer/ Scheme
Clicks on the pay button
Builds a transaction request including the created SDK Token and sends it to the FORT Payment SDK
Sends the transaction request to the FORT Server
Validates the SDK Token, device_id and the other request parameters
Returns the validation response to the FORT Payment SDK
Displays the payment page to the Customer
Enters the payment's details on the FORT Payment SDK
Validates the payment details
Sends "Authorization/ Purchase" request
Processes payment
Returns "Authorization/ Purchase" response
Returns response to the Merchant App
Displays transaction status to the Customer
Customer
Merchant App
FORT Payment SDK
FORT Server
Acquirer/ Scheme

4 Installing the Mobile SDK

These are the first steps you need to follow to integrate the Amazon Payment Services Android SDK into your Android application.

4.1 Installing the Android Mobile SDK

You need to complete two steps to install the Amazon Payment Services Android SDK.

Step 1

First, you need to add the repository to your build file. Add it in your root build.gradle at the end of repositories:

```
allprojects {
  repositories {
    ...
    maven { url https://android-sdk.payfort.com }
  }
}
```

Step 2

Add the Add fortSDK to your build.gradle dependencies using this code:

```
apply plugin: 'com.android.application'
android { ... }
dependencies {
  implementation("com.amazon:fortpayment:+:release@aar"){
    transitive = true
}
}
```

4.2 Setting Android OS permissions

You need to set the following two permission so that the Android mobile SDK works properly:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

5 Creating a mobile SDK token

A mobile SDK authentication token is required to authenticate every request sent to the SDK. The token is also significant to process payment operations with Amazon Payment Services through our Android mobile SDK.

To get started with our Android mobile SDK you must first establish the ability to generate a mobile SDK token.

NOTE: The creation and initiation of a mobile SDK token happens on your server – your server must generate the token by sending a request to the Amazon Payment Services API.

A unique authentication token must be created for each transaction. Each authentication token has a life-time of only one hour if no new request from the same device is sent.

5.1 Android mobile SDK token URLs

These are the URLs you need to use when you request a mobile SDK token for your Android app:

Test Environment URL

https://sbpaymentservices.payfort.com/FortAPI/paymentApi

Production Environment URL

https://paymentservices.payfort.com/FortAPI/paymentApi

5.2 Submitting token request parameters

You need to submit parameters as a REST POST request using JSON. Below we list the range of parameters for the Android mobile SDK token request.

5.2.1 Android Mobile SDK Token Request Parameters

When you send your request for an Android mobile SDK token you must send the following parameters to Amazon Payment Services:

Android Mobile SDK Token Request Parameters							
Parameter Name	**Type**	**Mandatory**	**Description**	**Length**	**Special Characters**	**Possible /Expected Values**	**Example**
service_command	Alpha	Yes	Command	20	_	SDK_TOKEN	
access_code	Alphanumeric	Yes	Access Code	20			zx0IPmPy5j p1vAz8Kpg 7
merchant_identifier	Alphanumeric	Yes	Your merchant ID	20			CycHZxVj
language	Alpha	Yes	The checkout page and messages language.	2		- en - ar	
device_id	Alphanumeric	Yes	A unique device identifier.	100	-		ffffffff-a9fa0b44-7b2729e7003 3c587
signature	Alphanumeric	Yes	A string hashed using the Secure Hash Algorithm.	200			7cad05f021 2ed933c9a 5d5dffa316 61acf2c827 a

NOTE: device_id is a value to be generated from the UUID Class Reference, you can generate this parameter by using the following command: FortSdk.getDeviceId(appContext)

5.2.2 Android Mobile SDK Token Response Parameters

These parameters will be returned in the Amazon Payment Services response:

Android Mobile SDK Token Response Parameters					
Parameter Name	**Type**	**Description**	**Length**	**Possible /Expected Values**	**Example**
service_command	Alpha	Command	20	SDK_TOKEN	
access_code	Alphanumeric	Access Code	20		zx0IPmPy5j p1vAz8Kpg 7
merchant_identifier	Alphanumeric	Your merchant ID	20		CycHZxVj
language	Alpha	The checkout page and messages language.	2	en ar	
device_id	Alphanumeric	A unique device identifier.	100		ffffffff-a9fa- 0b44-7b2729e70033c58 7
signature	Alphanumeric	A string hashed using the Secure Hash Algorithm.	200		7cad05f021 2ed933c9a 5d5dffa316 61acf2c827 a
sdk_token	Alphanumeric	An SDK authentication token to enable using the Android Mobile SDK.	100		Dwp78q3
response_message	Alphanumeric	Message description of the response code. It returns according to the request language.	150		Insufficient Funds
response_code	Numeric	Response Code carries the value of our system's response. *The code is made up of five digits, the first 2 digits refer to the statuses, and the last 3 digits refer to the messages.	5		20064
status	Numeric	A two-digit numeric value that indicates the status of the transaction.	2	(Please refer to Statuses in our Merchant Integration Guide).	

NOTE: Every parameter the merchant sends in the request should be received by the merchant in the response - even the optional parameters.

6 Processing transactions with the Android SDK

In this section we outline how you process a transaction using the Android mobile SDK. Your first step is to create a mobile SDK token as described above.

6.1 Two routes for in-app payment processing

As a merchant you have two ways in which you can process payments using the Amazon Payment Services Android mobile SDK.

1. **Standard payment processing UI**

   You can use the standard Amazon Payment Services Android SDK interface to display a standard payment UI screen. This option is customizable in two ways – you can hide the loading screen, and some of the UI elements can be customized. We address standard UI customization options in this section.

2. **Custom payment processing UI**

   Alternatively, you can choose to build your own payment processing UI by custom-coding an in-app payment processing feature. With this mobile SDK feature we allow Amazon Payment Services merchants to integrate and implement a native app checkout experience without displaying a default payment screen.

   Using this integration method your customers can securely enter their payment card details on a fully customized merchant checkout landing page using the your customized payment UI.

   The Amazon Payment Services Android SDK provides key building blocks including payment card input fields and an action pay button that you can encapsulate inside the checkout landing page and to provide your own inline customer experience.

   Amazon Payment Services' SDK card input fields will securely transmit the completed payment card data to Amazon Payment Services APIs for processing and to complete the transaction placement.

   Read more about a customized payment processing UI in this section.

6.2 Standard checkout implementation

1. Define a Callback Manager

Define and initialize an instance of the FortCallBackManager in your activity as follows:

```
private FortCallBackManager fortCallback= null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (fortCallback == null)
        fortCallback = FortCallback.Factory.create();
}
```

2. Attach the Callback to the Activity

You need to add the following statement to the onActivityResult function as follows:

```
@Override protected void onActivityResult(int requestCode, int resultCode, Intent
data){
super.onActivityResult(requestCode, resultCode,  data);
fortCallback.onActivityResult(requestCode,resultCode,data);
}
```

3. Call the Android Mobile SDK and Collecting the Android SDK request

The below code registers a new callback for a new request. The registerCallBack requires the following inputs:

```
public void registerCallback(
        Activity context, final FortRequest fortRequest, String
        environment, final int requestCode, final FortCallBackManager
callbackManager, boolean   showLoading, final
        FortInterfaces.OnTnxProcessed callback)
```

Below is a description for each parameter:

Parameter	Description
context	Passes the current activity context.
fortRequest	An instance of the model mentioned in Collect the Android Mobile SDK Request section.
environment	This parameter used to determine whether the request is going to be submitted to the test or production environment. It has two possible values: - FortSdk.ENVIRONMENT.TEST - FortSdk.ENVIRONMENT.PRODUCTION
requestCode	A unique ID for this request
callBackManager	The instance defined in section Define a Callback Manager.
showLoading	A Boolean flag to show or hide the loading dialog.
callback	A transaction callback listener that overrides the three callback options: - onCancel(): called when the user cancels the payment by clicking the back button. - onSuccess(): called when the transaction is processed successfully. - onFailure(): called when the transaction is failed.

The Java model/ bean of the Android Mobile SDK request is as follows:

```
public class FortRequest implements Serializable {
    private Map<String, Object> requestMap;
    private boolean showResponsePage;

    public Map<String, Object> getRequestMap() {
        return
                requestMap;
    }

    public void setRequestMap(Map<String, Object> requestMap) {
        this.requestMap = requestMap;
    }

    public boolean isShowResponsePage() {
        return showResponsePage;
    }

    public void setShowResponsePage(boolean showResponsePage) {
        this.showResponsePage = showResponsePage;
    }
}
```

NOTE: You can choose to display the Amazon Payment Services response page by passing "showResponsePage" value as "True".

The "requestMap" must contain all the Amazon Payment Services parameters for the order/ transaction. (Detailed information can be found in our Amazon Payment Services Merchant Integration Guide).

Example:

```
private Map<String, Object> collectRequestMap(String sdkToken) {
        Map<String, Object> requestMap = new HashMap<>();
        requestMap.put("command", "PURCHASE");
        requestMap.put("customer_email", "Sam@gmail.com");
        requestMap.put("currency", "SAR");
        requestMap.put("amount", "100");
        requestMap.put("language", "en");
        requestMap.put("merchant_reference", "merchant_reference ");

        requestMap.put("sdk_token", sdkToken);
        return requestMap;
    }
```

4. Initiate a checkout request:

For every transaction that needs to be processed, do the following call and handle the callback methods upon your business flow:

Example for register callback:

```
FortSdk.getInstance().registerCallback(this,fortRequest,5,fortCallback,
showLoading,new
    FortInterfaces.OnTnxProcessed() {
    @Override public void onCancel (Map < String, Object > requestParamsMap, Map
< String, Object > responseMap)
    {
    }

    @Override public void onSuccess (Map < String, Object > requestParamsMap, Map
< String, Object > fortResponseMap)
    {
    }

    @Override public void onFailure (Map < String, Object > requestParamsMap, Map
< String, Object > fortResponseMap)
    {
    }
}
```

6.3 Android Mobile SDK Device ID Value

Please make sure to use the following Android SDK function to generate the device_id parameter value that must be used for creating the sdk_token from your business security server:

```
String device_id = FortSdk.getDeviceId(this);
```

6.4 Android mobile SDK parameters

6.4.1 Request parameters

This is a list of the parameters you need to send when you send a request to the Android SDK.

Request Parameters							
Parameter Name	**Type**	**Mandatory**	**Description**	**Length**	**Special Characters**	**Possible or Expected Values**	**Example**
Command	Alpha	Yes	Command	20		AUTHORIZATION PURCHASE	
merchant_reference	Alphanumeric	Yes	Your unique order ID	40	- _		XYZ9239yu898
Amount	Numeric	Yes	The transaction's amount. *Each currency has predefined allowed decimal points that should be taken into consideration when sending the amount.	10			10000
Currency	Alpha	Yes	The currency of the	3			AED

			transaction’s amount in ISO code 3.				
Language	Alpha	Yes	Language used on the checkout page and for messages	2		en ar	
Customer_email	Alphanumeric	Yes	Your customer’s email address	254	_ - . @ +		customer @domain. com
SDK_token	Alphanumeric	Yes	The SDK token you generate for every SDK transaction	100			Dwp78q3
token_name	Alphanumeric	No	The token received from the tokenization process	100	. @ - _		Op9Vmp
payment_optio n	Alpha	No	Payment option	10		VISA MASTERCARD AMEX MADA (for Purchase operations and eci Ecommerce only).	
eci	Alpha	No	E-commerce indicator	16		ECOMMERCE	
order_description	Alphan umeric	No	It holds the description of the order	150	# ' / . _ - : $ Space		iPhone 12 Pro
customer_ip	Alphan umeric	No	It holds the customer 's IP address. *It's Mandatory, if the fraud service is active.	45			192.178.1. 10
customer_ name	Alpha	No	The customer’s name	40	_ \ / - . '		John Smith

phone_number	Alphanumeric	No	The customer’s phone number	19	+ - ( ) Space		00962797219966
settlement_reference	Alphanumeric	No	The Merchant submits this value to Amazon Payment Services. The value is then passed to the Acquiring bank and displayed to the merchant in the Acquirer settlement file.	34	- _ .		XYZ9239yu898
merchant_ extra	Alphanumeric	No	Extra data sent by merchant . Will be received and sent back as received. Will not be displayed in any report.	999	; / _ - , ' @		JohnSmith
merchant_ extra1	Alphanumeric	No	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	; / _ - , ' @		JohnSmith
merchant_ extra2	Alphanumeric	No	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	; / _ - , ' @		JohnSmith
merchant_ extra3	Alphanumeric	No	Extra data sent by merchant. Will be received and sent back as received. Will	250	; / _ - , ' @		JohnSmith

			not be displayed in any report.				
merchant_extra4	Alphanumeric	No	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	; / _ - , ' @		JohnSmith
merchant_extra5	Alphanumeric	No	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	; / _ - , ' @		JohnSmith

NOTE: Before sending the transaction value you must multiply the value by a factor that matches the ISO 4217 specification for that currency. Multiplication is necessary to accommodate decimal values. Each currency's 3-digit ISO code will have a specification for the number of digits after the decimal separator.

For example: If the transaction value is 500 AED; according to ISO 4217, you should multiply the value with 100 (to accommodate 2 decimal points). You will therefore send an AED 500 purchase amount as a value of 50000.

Another example: If the amount value was 100 JOD; according to ISO 4217, you should multiply the value with 1000 (to accommodate 3 decimal points). You therefore send a JOD 100 purchase amount as a value of 100000

6.4.2 Response parameters

Response Parameters					
Parameter Name	**Type**	**Description**	**Length**	**Possible or Expected Values**	**Example**
Command	Alpha	Command	20	AUTHORIZATION PURCHASE	
merchant_r eference	Alphanu meric	Your unique order ID	40		XYZ9239y u898
Amount	Numeric	*Each currency has predefined allowed decimal points that should be taken into consideration when sending the amount.	10		10000
Currency	Alpha	The currency of the transaction's amount in ISO code 3.	3		AED
Customer_email	Alphanumeric	Your customer's email address	254		customer @domain. com
fort_id	Numeric	The order's unique reference returned by our system.	20		14437968668 48
SDK_token	Alphanumeric	The SDK token you generate for every SDK transaction	100		Dwp78q3
token_name	Alphanumeric	The token received from the tokenization process	100		Op9Vmp
payment_optio n	Alpha	Payment option	10	VISA MASTERCARD AMEX MADA (for Purchase operations and eci Ecommerce only).	
eci	Alpha	E-commerce indicator	16	ECOMMERCE	
authorization_code	Alphanu meric	The authorization code returned from the 3rd party.	100	-	P100000000 0000372136

order_description	Alphan umeric	It holds the description of the order	150	-	iPhone 12 Pro
response_message	Alphanumeric	The message description of the response code; it returns according to the request language.	150	-	Insufficient Funds
response_code	Numeric	Response code carries the value of our system's response. *The code consists of five digits, the first 2 digits represent the response status, and the last 3 digits represent the response message.	5	-	20064
customer_ip	Alphan umeric	It holds the customer 's IP address. *It's Mandatory, if the fraud service is active.	45		192.178.1. 10
customer_ name	Alpha	The customer’s name	40		John Smith
expiry_date	Numeric	The card's expiry date.	4	-	1705
card_number	Numeric	The masked credit card's number. *Only the MEEZA payment option takes 19 digits card number. *AMEX payment option takes 15 digits card number. *Otherwise,they take 16 digits card number.	16	-	400555** ****0001
status	Numeric	A two-digit Numeric value that indicates the status of the transaction.	2	- (Please refer to section statuses).	
phone_number	Alphan umeric	The customer’s phone number	19		00962797 219966
settlement_reference	Alphan umeric	The Merchant submits this value to Amazon Payment Services. The value is then passed to the Acquiring bank and displayed to the merchant in the Acquirer settlement file.	34		XYZ9239 yu898

merchant_ extra	Alphan umeric	Extra data sent by merchant . Will be received and sent back as received. Will not be displayed in any report.	999		JohnSmith
merchant_ extra1	Alphan umeric	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250		JohnSmith
merchant_ extra2	Alphan umeric	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250		JohnSmith
merchant_ extra3	Alphan umeric	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250		JohnSmith
merchant_ extra4	Alphan umeric	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250		JohnSmith
merchant_ extra5	Alphan umeric	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250		JohnSmith

7 Standard checkout UI sample code

Below we include a complete sample code section illustrating the full checkout process when you use the standard checkout UI. The sample includes capturing payment card data, initializing the Amazon Payment Services Android SDK, submitting payment card data and processing a transaction.

```
public class PayFortSdkSample extends Activity {
    private FortCallBackManager fortCallback = null;
    String deviceId = "", sdkToken = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

     // create Fort callback instance
        fortCallback = FortCallback.Factory.create();

        // Generating deviceId
        deviceId = FortSdk.getDeviceId(PayFortSdkSample.this);
        Log.d("DeviceId ", deviceId);

        // prepare payment request
        FortRequest fortrequest = new FortRequest();

fortrequest.setRequestMap(collectRequestMap("PASS_THE_GENERATED_SDK_TOKEN_HERE"));
        fortrequest.setShowResponsePage(true); // to [display/use] the SDK
response page
        // execute payment request callSdk(fortrequest);
    }

    private Map<String, Object> collectRequestMap(String sdkToken) {
        Map<String, Object> requestMap = new HashMap<>();
        requestMap.put("command", "PURCHASE");
        requestMap.put("customer_email", "Sam@gmail.com");
        requestMap.put("currency", "SAR");
        requestMap.put("amount", "100");
        requestMap.put("language", "en");
        requestMap.put("merchant_reference", "ORD-0000007682");
        requestMap.put("customer_name", "Sam");
        requestMap.put("customer_ip", "172.150.16.10");
        requestMap.put("payment_option", "VISA");
        requestMap.put("eci", "ECOMMERCE");
        requestMap.put("order_description", "DESCRIPTION");
        requestMap.put("sdk_token", sdkToken);
        return requestMap;
    }

    private void callSdk(FortRequest fortrequest) {

        FortSdk.getInstance().registerCallback(PayFortSdkSample.this, fortrequest,
                FortSdk.ENVIRONMENT.TEST, 5, fortCallback, new
FortInterfaces.OnTnxProcessed() {
                    @Override
                    public void onCancel(Map<String, Object> requestParamsMap,
Map<String, Object> responseMap) {
                        Log.d("Cancelled ", responseMap.toString());
                    }
                    @Override
                    public void onSuccess(Map<String, String> requestParamsMap,
Map<String, Object> fortResponseMap) {    //TODO: handle me
```

```
                        Log.i("Success ", fortResponseMap.toString());
                    }
                    @Override
                    public void onFailure(Map<String, String> requestParamsMap,
                                          Map<String, Object> fortResponseMap) {
                        Log.e("Failure ", fortResponseMap.toString());
                    }

                });


    }


    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        fortCallback.onActivityResult(requestCode, resultCode, data);
    }

}
```

8 Customizing the payment UI

You can customize the payment UI presented by our Android SDK in a number of ways to better reflect your business. This is when you use the standard UI.

8.1 Customizing the Android Mobile SDK Payment Layout

We provide you with the res folder that includes the source code of the pages in order to customize the design, themes, etc. You can customize both English and Arabic layouts as needed. However, please take the following tips into consideration:

- Don’t change the layout name because it is considered an override process.
- Make sure to use all the views that has the ID property in order to avoid the NullPointerException.
- Redesign the view for portrait orientation. Note that Landscape orientation isn't supported.
- You can support as much layout densities as you want.

Our Mobile SDK v 1.9 consists one of the following three main activities design:

- activity_cc_payment.xml
- activity_cc_response.xml
- activity_init_secure_conn

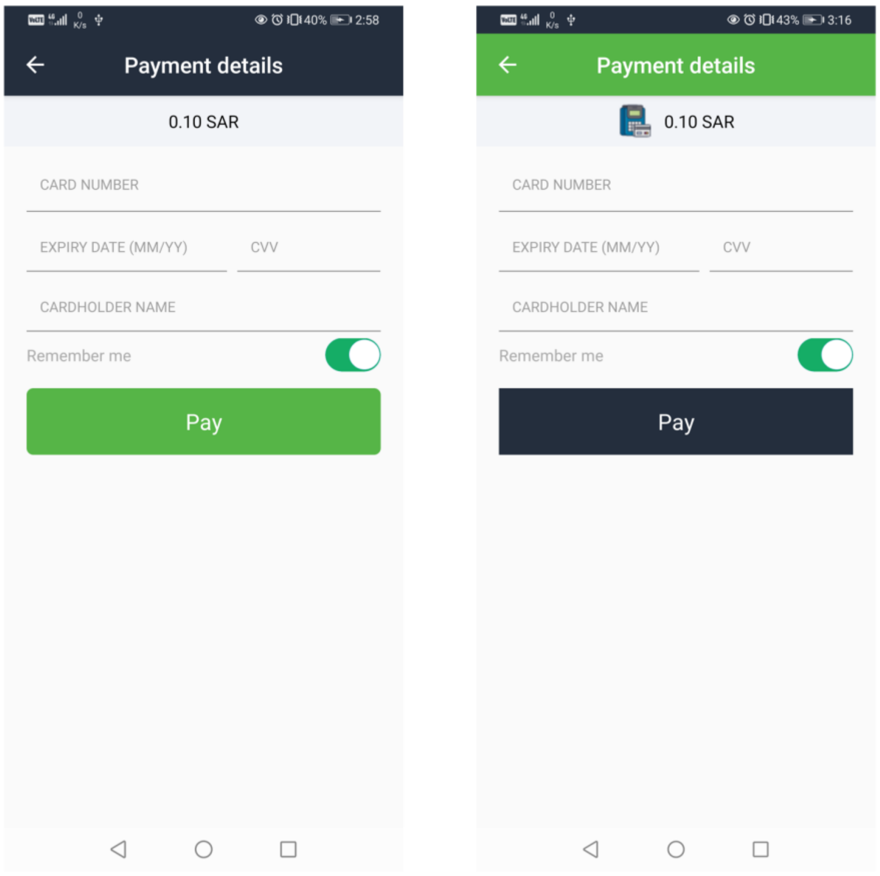


Figure 1: Standard vs. Customized Mobile SDK Payment Page Design

8.2 Design Customization Codes:

The following code was used to customize the way the "Amount" is displayed in the Standard Mobile SDK Payment Page:

```
<TextView
    android:id="@+id/amountTV"
    android:layout_width="match_parent"
    android:layout_height=”38dp"
    android:background="@color/pf_light_gray"
    android:gravity="center_horizontal|center_vertical"
    android:textColor="@color/colorBlack"
    android:textSize="@dimen/_13ssp"
app:layout_constraintTop_toBottomOf="@+id/appbarLayout
/>
```

The following code was used to customize the way the "Amount" is displayed in the Customized Mobile SDK Payment Page:

```
<LinearLayout
    android:id="@+id/amountContainer"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/pf_light_gray"
    android:gravity="center"
    app:layout_constraintTop_toBottomOf="@+id/appbarLayout">


    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="10dp"
        android:src="@drawable/merchant"
        android:layout_marginRight="10dp" />

    <TextView
        android:id="@+id/amountTV"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/_38sdp"
        android:gravity="center_horizontal|center_vertical"
        android:textColor="@color/colorBlack"
        android:textSize="@dimen/_13ssp"
        tools:text="10.00 USD" />

</LinearLayout>
```

As appears in the previous codes, elements with IDs haven't been changed in type or removed. For example: android:id="@+id/amountTV".

We were able to add static elements such as: “ImageView“ element that contains the Merchant's logo.

To sum up, you can add any static elements or redesign the view, while keeping the views' elements used in the Standard layout that hold IDs.

NOTE: The customized XML file should be added to the layout file in the target project (Merchant Application) to override the SDK file.

9 Custom-coding a payment processing UI

In this section we outline key information that you need to create your own payment processing UI using the tools in our Android SDK.

9.1 Stage 1: Generate an SDK token

You need to generate an SDK token before you can start processing payments using your custom payment processing UI. Refer to the SDK token section earlier in this document for instructions on creating an SDK token.

9.2 Stage 2: creating the card components

You create your custom payment screen by using the following five components included in the Amazon Payment Services Android SDK:

Table of components

The Android Mobile SDK allows merchants to securely integrate the payment functions through Custom Components:

- FortCardNumberView (Check CardBrand,CardNumber)
- CardCvvView (Check if cvv match cardBrand)
- CardExpiryView (Check date)
- CardHolderNameView
- PayfortPayButton (Collect data from previous Components)

Custom Components		
Attributes	**Type**	**Description**
app:boxBackgroundShape	Enum 1- none 2- filled 3- outlione	Whether the text input area should be drawn as a filled box, an outline box, or not as a box.
app:hintTextColor	Reference or color Ex : #fff123	Color of the hint text.
app:hintText	String	Hint text to display when the text is empty.
app:textSize	Dimension Ex : 12sp	Size of the text. Recommended dimension type for text is "sp"
app:textColor	Reference or color Ex : #fff123	text Color of the text displayed in the text input area
app:boxStrokeErrorColor	Color Ex : #fff123	The color to use for the box's stroke in outline box mode when an error is being displayed. If not set, it defaults to errorTextColor if on error state

app:errorTextColor	Color Ex : #fff123	Text color for any error message displayed. If set, this takes precedence over errorTextAppearance.
app:errorTextAppearance	Reference Ex: @style/customStyle	TextAppearance of any error message displayed.
app:boxBackgroundColor	Color Ex : #fff123	The color to use for the box's background color when in filled box mode.If a non-stateful color resource is specified, default colors will be used for the hover and disabled states.

XML Usage

Card components

```
<!--CardNumber-->
   <com.payfort.fortpaymentsdk.views.FortCardNumberView
                android:id="@+id/etCardNumberView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                app:boxBackgroundShape="outline"
                />

   <!--CardExpiryView-->
    <com.payfort.fortpaymentsdk.views.CardExpiryView
                    android:id="@+id/etCardExpiry"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    app:boxBackgroundShape="outline" />

              <!--CardCvvView-->
                <com.payfort.fortpaymentsdk.views.CardCvvView
                    android:id="@+id/etCardCvv"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    app:boxBackgroundShape="outline" />

        <!--CardHolderNameView-->
       <com.payfort.fortpaymentsdk.views.CardHolderNameView
                android:id="@+id/cardHolderNameView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                app:boxBackgroundShape="outline" />
```

9.3 Stage 3: Initiate the actual payment.

PayfortPayButton

Responsible for collecting card data from Card Components above and Submit Successful payment with simple Few steps also it's had the capability to do Direct Pay without needs for Card Components

```
<!--PayfortPayButton-->
       <com.payfort.fortpaymentsdk.views.PayfortPayButton
                android:id="@+id/btntPay"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
             />
```

PayfortPayButton Methods

```
/**
 * Responsible for Save token or not
 * @param isRememberMe Boolean
 */
fun isRememberMeEnabled(isRememberMe: Boolean)

/**
 * Update Request After doing Setup
 * @param fortRequest FortRequest
 */
fun updateRequest(fortRequest: FortRequest)


/**
 * this Setup used with Pay with Full Components from outside SDK
 * @param environment String
 * @param request FortRequest
 * @param payComponents PayComponents
 * @param payFortCallback PayFortCallback
 */
fun setup(environment: String,
          request: FortRequest,
          payComponents: PayComponents,
          payFortCallback: PayFortCallback)

/**
 * this Setup used with DirectPay only
 * @param environment String
 * @param request FortRequest
 * @param payFortCallback PayFortCallback?
 */
fun setup(environment: String,
          request: FortRequest,
          payFortCallback: PayFortCallback?)
```

9.4 Sample Code

In this example we explain how to use PayfortPayButton with Card Components in the custom UI screen.

Kotlin

```
//You Need to Create PayComponents Object that holds all Card Components
val payComponents = PayComponents(etCardNumberView, cvvView = etCardCvv,
etCardExpiry, holderNameView = cardHolderNameView)

//then you need to Create FortRequest via FortRequest.class
val fortRequest= FortRequest() // fill all the parameters required

//then you need to choose the environment Via FortSdk.ENVIRONMENT it's contains
TEST and PRODUCTION ENVIRONMENTS
val  environment=  FortSdk.ENVIRONMENT.TEST

// Finally you need to decleare a PayFortCallback
var callback=  object : PayFortCallback {
    override fun startLoading() {   }
    override fun onSuccess(requestParamsMap: Map<String, Any>, fortResponseMap:
Map<String, Any>) {}
    override fun onFailure(requestParamsMap: Map<String, Any>, fortResponseMap:
Map<String, Any>) { }
}

btnPay.setup(environment, fortRequest, payComponents, callback)
```

Java

```
//You Need to Create PayComponents Object that holds all Card Components
PayComponents payComponents = new PayComponents(etCardNumberView,  etCardCvv,
etCardExpiry,  cardHolderNameView);

//then you need to Create FortRequest via FortRequest.class
FortRequest fortRequest = new FortRequest(); // fill all the parameters required

//then you need to chsose the environment Via FortSdk.ENVIRONMENT it's contains
TEST and PRODUCTION ENVIRONMENTS
String environment = FortSdk.ENVIRONMENT.TEST;

// Finally you need to decleare a PayFortCallback
PayFortCallback callback = new PayFortCallback() {
    @Override
    public void startLoading() { }
    @Override
    public void onSuccess(@NotNull Map<String, ?> requestParamsMap, @NotNull
Map<String, ?> fortResponseMap) { }
    @Override
    public void onFailure(@NotNull Map<String, ?> requestParamsMap, @NotNull
Map<String, ?> fortResponseMap) { }
};

    btnPay.setup(environment, fortRequest, payComponents, callback)
```

10 Using Direct Pay

The Direct Pay feature enables Amazon Payment Services merchants to securely process e-commerce transactions using tokenized payment card details.

For customers that already supplied their payment card details in a previous transaction and where a card token was generated, customers need to just provide the card security code to complete their purchase.

The card token and provided card security code can be sent to the Amazon Payment Services iOS mobile SDK to complete the customer purchase through Direct Pay operation to complete the order placement using eCommerce channel.

Note you can use Direct Pay both with the standard payment UI or with a customized payment UI.

10.1 Direct Pay sample code

In this example we demonstrate how to use PayfortPayButton with DirectPay. This Method is required to pass card_security_code and token_name to FortRequest.

Kotlin

```
private fun collectRequestMap(sdkToken: String): FortRequest {
    val fortRequest = FortRequest()
    val requestMap: MutableMap<String, Any> = HashMap()
    requestMap["command"] = "AUTHORIZATION"
    requestMap["customer_email"] = "Sam@gmail.com"
    requestMap["currency"] = "AED"
    requestMap["amount"] = "100"
    requestMap["language"] = "en"
    requestMap["card_security_code"] = "123"
    requestMap["token_name"] = "payfort"
    requestMap["merchant_reference"] = "merchant_reference"
    requestMap["sdk_token"] = sdkToken
    fortRequest.requestMap = requestMap
    fortRequest.isShowResponsePage = false
    return fortRequest
}
//you need to Create FortRequest via FortRequest.class
val fortRequest=collectRequestMap("SDK_TOKEN")

//then you need to chsose the environment Via FortSdk.ENVIRONMENT it's contains
TEST and PRODUCTION ENVIRONMENTS
val  environment=  FortSdk.ENVIRONMENT.TEST

// Finally you need to decleare a PayFortCallback
var callback=  object : PayFortCallback {
    override fun startLoading() {   }
    override fun onSuccess(requestParamsMap: Map<String, Any>, fortResponseMap:
Map<String, Any>) {}
    override fun onFailure(requestParamsMap: Map<String, Any>, fortResponseMap:
Map<String, Any>) { }
}

btnPay.setup(environment, fortRequest, callback)
```

Java

```
private FortRequest collectRequestMap(String sdkToken) {
    FortRequest fortRequest= new FortRequest();
    Map<String, Object> requestMap = new HashMap<>();
    requestMap.put("command", "AUTHORIZATION");
    requestMap.put("customer_email", "Sam@gmail.com");
    requestMap.put("currency", "AED");
    requestMap.put("amount", "100");
    requestMap.put("language", "en");
    requestMap.put("card_security_code", "123");
    requestMap.put("token_name", "payfort");
    requestMap.put("merchant_reference", "merchant_reference");
    requestMap.put("sdk_token", sdkToken);
    fortRequest.setRequestMap(requestMap);
    fortRequest.setShowResponsePage(false);
    return fortRequest;
}

//you need to Create FortRequest via FortRequest.class
FortRequest fortRequest = collectRequestMap(“SDK_TOKEN”);


//then you need to chsose the environment Via FortSdk.ENVIRONMENT it's contains
TEST and PRODUCTION ENVIRONMENTS
String environment = FortSdk.ENVIRONMENT.TEST;

// Finally you need to decleare a PayFortCallback
PayFortCallback callback = new PayFortCallback() {
    @Override
    public void startLoading() {}

    @Override
    public void onSuccess(@NotNull Map<String, ?> requestParamsMap, @NotNull
Map<String, ?> fortResponseMap) {}


    @Override
    public void onFailure(@NotNull Map<String, ?> requestParamsMap, @NotNull
Map<String, ?> fortResponseMap) {}
};

    btnPay.setup(environment, fortRequest, callback);
```

11 Amazon Payment Services Android SDK transaction feedback

While a transaction is completed, we will send a response directly to your direct transaction feedback URL. In theory, direct response feedback cannot be interrupted unless the URL you provided for responses is not functional at the time of the response.

We do this so that your server receives a response even if your customer does not successfully redirect to the return URL on your website, which may happen if your customer's browser or connection fails during the transaction.

11.1 Receiving transaction feedback

There are two ways in which you receive transaction feedback:

Direct transaction feedback. Amazon Payment Services sends an immediate payment processing response whenever a transaction is completed.

You can rely on this response for transaction feedback even where your user closed the browser before getting redirected successfully to the redirection URL or where your user was not redirected due to a drop in the internet connection.

Notification feedback. Were we need to provide you with the status of a transaction once it is received. In other words, we send notification feedback to alert you to any changes in the status of a transaction.

For example, if the transaction was pending due to the unavailability of any party to the transaction, the final update will be pushed to your notification feedback endpoint.

Notification feedback deals with a wide range of scenarios and it is critical that your website is configured to receive notification feedback correctly. For example, transaction feedback can alert you to any transactions that were stuck in "uncertain" status, but which have recovered to final status.

Direct feedback allows you to subscribe to transaction updates for uncertain transactions whenever you process a payment. It is a method for receiving the transaction response automatically once the transaction had completed or if there was an update.

11.2 Registering Transaction Feedback URLs

To receive transaction feedback, you must register with Amazon Payment Services the transaction feedback URLs you set up on your server. Follow these steps:

1. Log in to your back-office account.
2. Select the active channel under Integration SettingsTechnical Settings.
3. Enter your Direct Transaction Feedback URL and Notification Transaction Feedback URL.
4. Click “Save Changes” button.

11.3 Transaction Feedback Implementation

We will send the response via HTTP POST request in POST form format to your webhook URL. The submission type can be changed from POST form to JSON or XML in your APS account settings. We only permit configuring URLs in HTTPS for direct feedback and for notification feedback.

To acknowledge receiving the direct feedback and notification successfully, the webhook URL must return a 2xx or 302 HTTP status. In case the URL returns different HTTP responses, our server will keep retrying up to 10 times until we receive a success response. We wait ten seconds between retries.

You can change and configure the retry mechanism attributes to increase or decrease the number of retries, time between retries and the grace period before we send the notifications.

You must create endpoints that accept direct transaction feedback and notification feedback from us via a webhook. You can customize how transaction feedback works -- for example, by including the grace period before a direct feedback notification is sent, and the time elapsed between retrying feedback submission.

To customize transaction feedback, email integration-ps@amazon.com. You can request to change the submission type to JSON or XML. You can also change the grace period or the time interval between the retries please contact us on integration@payfort.com

NOTE: You can check the direct and notification feedback logs in your back office account to check the details related to the submission like the Transaction Feedback URL which was triggered, The response which our system pushed, the response Code and Status retuned from your Transaction Feedback URL.

NOTE: The specifics of the data will differ based upon the financial operation that has been processed. Your code must be able to accommodate different data.

12 Validate API call

Amazon Payment Services offers an API to request validation of the payment card details without performing an actual transaction.

Validate API can help you to initiate an API request in order to validate the input parameters values, this will reduce the possibility of encountered API errors due to wrong user inputs before processing the actual payment card transaction.

Fort SDK has a new method (validate) in FortSdk.class This method can validate FortRequest to check all parameters for validity.

```
/**
 * Responsible for Validating FortRequest
 * @param context Context
 * @param environment String
 * @param fortRequest FortRequest
 * @param fortCallback PayFortCallback
 */
fun validate(context: Context,
             environment: String
             ,fortRequest: FortRequest,
             fortCallback: PayFortCallback)
```

Example:

Java

```
// you need to Create FortRequest via FortRequest.class
FortRequest fortRequest = new FortRequest(); // fill all the parameters required
//then you need to chsose the environment Via FortSdk.ENVIRONMENT
//it's contains TEST and PRODUCTION ENVIRONMENTS
String environment = FortSdk.ENVIRONMENT.TEST;
// Finally you need to decleare a PayFortCallback
PayFortCallback callback = new PayFortCallback() {
    @Override
    public void startLoading() { }
    @Override
    public void onSuccess(@NotNull Map<String, ?> requestParamsMap, @NotNull
Map<String, ?> fortResponseMap) { }
    @Override
    public void onFailure(@NotNull Map<String, ?> requestParamsMap, @NotNull
Map<String, ?> fortResponseMap) { }
};

FortSdk fortSdk = FortSdk.getInstance();
    fortSdk.validate(this,environment,fortRequest,callback);
```

Kotlin

```
// you need to Create FortRequest via FortRequest.class
val fortRequest = FortRequest() // fill all the parameters required
//then you need to chsose the environment Via FortSdk.ENVIRONMENT it's contains
TEST and PRODUCTION ENVIRONMENTS
```

```
val environment = FortSdk.ENVIRONMENT.TEST
// Finally you need to decleare a PayFortCallback
val callback: PayFortCallback = object : PayFortCallback {
    override fun startLoading() {}
    override fun onSuccess(requestParamsMap: Map<String, Any>, fortResponseMap:
Map<String, Any>) {}

    override fun onFailure(requestParamsMap: Map<String, Any>, fortResponseMap:
Map<String, Any>) {}
}

val fortSdk = FortSdk.getInstance()
fortSdk.validate(this, environment, fortRequest, callback)
```

NOTE: When you make use of the Validate API call you still need to generate a mobile SDK token even though you are not processing a transaction.

13 Migrating the previous version

In this section we outline the steps you need to take to migrate the Amazon Payment Services Android SDK to the latest release. To complete the migration, follow these steps:

1. Remove old SDK from Libs Folder.
2. Remove old dependencies from Gradle file that related to old SDK

```
dependencies {
    implementation  fileTree(include: ['*.jar'], dir: 'libs')
    api project(path: ':FORTSDKv1.6')
    api 'com.android.support:design:29+'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    api 'com.victor:lib:1.0.1'
    api 'com.google.code.gson:gson:2.8.0'
    api 'com.shamanland:fonticon:0.1.8'
    api('com.nispok:snackbar:2.11.+') {
        // exclusion is not necessary, but generally a good idea.
        exclude group: 'com.google.android', module: 'support-v4'
    }
    api 'com.google.guava:guava:19.0'
    api 'org.bouncycastle:bcprov-jdk16:1.46'
}
```

3. Check the installation steps in section 3

4. Make sure to sync new SDK in your Gradle File

```
File > Sync Project with Gradle Files
```

5. For merchants that already use custom layout feature, they need to replace their custom layout xml files that related to previous SDK versions (< 2.0) and override it with new one.

   Note: if you override Arabic xml files you need to delete it since new SDK support it without the need to override it.

   Below the difference between old SDK layout and new one:

XML Files	old SDK	new SDK
activity_cc_payment	Download	Download
activity_cc_response	Download	Download
activity_init_secure_conn	Download	Download

6. Change old imports to new imports

   Old imports

```
import com.payfort.fort.android.sdk.base.FortSdk;
import com.payfort.sdk.android.dependancies.commons.Constants.FORT_PARAMS;
import com.payfort.fort.android.sdk.base.FortSdk;
import com.payfort.fort.android.sdk.base.callbacks.FortCallBackManager;
import com.payfort.fort.android.sdk.base.callbacks.FortCallback;
import com.payfort.sdk.android.dependancies.base.FortInterfaces;
import com.payfort.sdk.android.dependancies.models.FortRequest;
```

   New imports

```
import com.payfort.fortpaymentsdk.constants.Constants.FORT_PARAMS;
import com.payfort.fortpaymentsdk.FortSdk;
import com.payfort.fortpaymentsdk.callbacks.FortCallBackManager;
import com.payfort.fortpaymentsdk.callbacks.FortCallback;
import com.payfort.fortpaymentsdk.callbacks.FortInterfaces;
import com.payfort.fortpaymentsdk.domain.model.FortRequest;
```

7. Make Sure to enable ViewBinding in your project

   To enable view binding in a module, set the viewBinding build option to true in the module-level build.gradle file, as shown in the following example:

```
android {
    ...
    buildFeatures {
        viewBinding true
   }
}
```

8. Make Sure to use Java 8

   To start using supported Java 8 language features, update the Android plugin to 3.0.0 (or higher). After that, for each module that uses Java 8 language features (either in its source code or through dependencies), update the module's build.gradle file, as shown below:

```
android {
    ...
    // Configure only for each module that uses Java 8
    // language features (either in its source code or
    // through dependencies).
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
                targetCompatibility JavaVersion.VERSION_1_8
    }
    // For Kotlin projects
    kotlinOptions {
        jvmTarget = "1.8"
    }
}
```

14 Appendix: About the Software

14.1 About this software

The Amazon Payment Services Android mobile SDK allows merchants to securely integrate payment functionality into a native Android app. It allows merchants to easily accept in-app payments.

Instead of the traditional, time-consuming, and complex way of being redirected to the mobile browser to complete the payment, in-app payments can easily be offered thanks to the Android Mobile SDK. In turn, this gives the merchants' customers a smooth, pleasing user experience thanks to in-app payment functions through the native applications. (move to about this software)

14.2 Supported Platforms

The Amazon Payment Services Android SDK supports devices running Android 4.1.x and later (API level 16). In other words, Android Ice Cream Sandwich or higher is supported. This release supports Android Pie API 28.

14.3 Localization

You can use both English and Arabic when you implement the Android SDK.

14.4 Screen Orientation

Currently, portrait is the only orientation supported within the Amazon Payment Services mobile SDK – unless you build a customized payment integration.

14.5 Supported Payment Methods

Using the Android SDK the merchant can process debit or credit card transactions only.

14.6 Supported Payment Options

The supported credit card payment options are VISA, MASTERCARD, American Express (AMEX), MADA and MEEZA.