amazon payment services

Android

Merchant Integration Guide

Document Version: 3.6.2

Oct, 2020

Contact Us

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

Table of Contents

1 Amazon Payment Services

Amazon Payment Services is a trusted online payment gateway enabling businesses, governments, SMEs, startups and institutions with innovative payment options for both the banked and non-banked online shoppers.

We work with our customers first by understanding both their financial and revenue model; identify areas of risk exposure, and payment processes in order to formulate strategies to maximize online payment acceptance. We work under the notion that “People are different” thus we help our Merchants in offering different payment options that mirror their online shoppers behavior for both credit card and non-credit cardholders.

Our team is comprised of seasoned bankers, technology gurus, and risk management experts that have been helping hundreds of firms manage and innovate their online payment processes across the Arab world and beyond

2 About this Document

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

2.1 Intended Audience

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

2.2 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 reference to PayFort. You may also see reference 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.

3 Before Starting the integration

These are the steps you need to know to start building an integration with Amazon Payment Services:

Step 1: Access your test account You need to make sure that you have access to the test account, it's a full test environment allow you to simulate and process simulation transactions.

Step 2: make sure that you are using the correct integration type

Prior 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 mentioned under every section in the API document

Step 3: Create the Transaction Request

Process the valid API request depends on transaction parameters included, you need to check the documentation and read every parameter possible values in order to reduce the errors in processing the transaction.

Step 4: Process the 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 Direct Transaction Feedback section.

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 5: Test and Go Live

You can use our testing cards to test your integration and simulate your test cases. The Amazon Payment Services team may require testing your integration before going live to check your application integration.

4 About the Software

4.1 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.

4.2 Localization

The Android mobile SDK supports both English and Arabic languages.

4.3 Screen Orientation

Portrait is the only orientation supported within the Amazon Payment Services mobile SDK.

4.4 Supported Payment Methods

Through the first version of the Android mobile SDK, the merchant has the ability to process **CREDIT CARD** transactions only.

4.5 Supported Payment Options

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

5 Android SDK – Device ID permission

This section helps the developers to understand the need and usage of the permission requested by the Android SDK to generate a unique device ID.

NOTE!
You might not face this issue through your payment flow. It depends on the time you are requesting the getDeviceID function and ask about the permission for the first time.

A part of the Android mobile SDK flow is to get a unique ID for the device. Generating the ID is based on more than one input (collecting as much as possible will lead to a real unique ID).

Getting the needed permission through the flow on the case of getting the client granted that will produce a miss match device ID on the 1st SDK call.

To avoid the mismatch flow we are suggesting you the following solutions:

- **Handle the 1st SDK call in the onActivityResults().**
  Since The Amazon Payment Services SDK is a module running within the main application context the requested permission response will be returned to the merchant context. Once the activity that called geDeviceId for the 1st received a call-back in the onActivityResult() with request code = 222 you can for sure starts the payment flow of creating an SDK token and calling the SDK afterwards.

- Call getDeviceID on a previous activity or in your application class to make sure that the permission request was triggered before you reach the payment step.

6 Android Mobile SDK

The Android Mobile SDK allows merchants to securely integrate the payment functions. It also 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 be completed through our Android mobile SDK. In turn, this gives the merchant's consumers a smooth, pleasing user-experience by using in-app payment functions through the native applications.

6.1 Download the Android mobile SDK

To download the Android mobile SDK, click here.

6.2 Create Android mobile SDK authentication 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 in Amazon Payment Services through our Android mobile SDK.

NOTE!

- A unique token should be created for each transaction. Each token has a lifetime of only one hour if no new request from the same device is sent.
- The creation and initiation of a mobile SDK authentication token happens on the Merchant's server side.

6.3 Android mobile SDK authentication token URLs

Test Environment URL
https://sbpaymentservices.payfort.com/FortAPI/paymentApi

Production Environment URL
https://paymentservices.payfort.com/FortAPI/paymentApi

6.4 Parameters Submission Type

REST POST request using JSON.

6.4.1 Android mobile SDK authentication token Request Parameters

Include the following parameters in the request you will send 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			zx0IPmPy 5jp1vAz8 Kpg7
merchant_identifier	Alphanumeric	Yes	The ID of the Merchant.	20			CycHZxVj
language	Alpha	Yes	The checkout page and messages language.	2		- en ar	
device_id	Alphanumeric	Yes	A unique device identifier.	100	-	1	ffffffff-a9fa0b44-7b2729e70033c587
signature	Alphanumeric	Yes	A string hashed using the Secure Hash Algorithm. (More details are available in our Merchant Integration Guide).	200		2	7cad05f0212ed933c9a5d5dffa31661acf2c827a

NOTE!
device_id - This value to be generated from the UIDevice Class Reference, and you can generate this parameter as the following:

[payFort getUDID];

6.4.2 Android Mobile SDK Token Response Parameters

The following parameters will be returned in 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		zx0IPmPy5jp1vAz8Kpg7
merchant_identifier	Alphanumeric	The ID of the Merchant.	20		CycHZxVj
language	Alpha	The checkout page and messages language.	2	en ar	
device_id	Alphanumeric	The ID of the used device for this payment.	100		ffffffff-a9fa-0b44-7b2729e70033c587
signature	Alphanumeric	A string hashed using the Secure Hash Algorithm. (More details are available in our Merchant Integration Guide).	200		7cad05f0212ed933c9a5d5dffa31661acf2c827a
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 section 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 ones.

7 Integrate the Android mobile SDK

To process a transaction using the Android Mobile SDK, create a mobile SDK authentication token (Please refer to section Create Mobile SDK authentication token) and proceed through the following sections.

7.1 IDE Configurations

To start using the Android Mobile SDK, do the following:

1. Extract the folder found in section 3.1
2. Download SDK for Android.
3. Unzip the downloaded Android SDK.
4. The folder content is as follows:

- Dependencies folder which includes: - Eclipse folder:
  a. .jar files for the SDK dependencies.
  b. LINKS_README text file that contains a list for sources of the above jars and a list of required libraries to be added as dependencies as well (Manual configuration).
  - AndroidStudio_gradle text file; add the compile command for what your project does not include.
- Res folder which includes:
  - Layout
  - Layout-ar
- FORTSDKv1.6.aar
- Make sure to browse the READ_ME file for the release changes and SDK updates

> **NOTE!**
> Please make sure to read our notes according to the IDE you are using whether it's the Eclipse or Android Studio.

7.1.1 Android Studio

To continue the integration, please proceed with the following steps:

1. Go to File → New → New Module.
2. Select "Import .JAR/.AAR Package" and click next.
3. Enter the path to .aar file and click finish.
4. Browse to the dependencies folder and open the "AndroidStudio_gradle" text file.
5. Copy and paste the implementation/ api lines that are NOT already supported in your dependency block. (All listed dependencies are required).
6. Sync the project with gradle files by clicking the  button.
7. Clean the project.
8. The SDK is now ready for your use.

7.1.2 Eclipse

Since you have extracted the .aar file, you are ready to start. The integration will include two main steps. For the first step, you need to create a library project by following the below steps:

1. Create a new project (from this time it's called “library project”) in your workspace.
2. Do not forget to mark it as library.
3. Clear the src folder of the library project.
4. Unzip the .aar file. You can rename it to zip and then unzip it or use any tool.
5. Copy the classes.jar file to libs folder on the library project. 6. Replace the res folder on library project with the res folder of the .aar file.

The project you have created contains almost everything you need. Now let’s start configuring your project to reference this library project by following the below points:

1. In the target project, use the library created in step one (mentioned above) as a dependency.
2. Open the AndroidManifest.xml file inside .aar file and make sure to copy everything it takes (permissions, activities, services, receivers …) in the AndroidManifest.xml file of the target project.
3. Copy the entire contents (if any) inside the assets folder of the .aar file to the assets folder of the target project.
4. Copy the entire contents (if any) inside the libs folder of the .aar file to the libs folder of the target project.
5. Open the dependencies file → Eclipse, then copy all .jar files and add them to the libs folder of the target project.
6. Check if your target project has the project dependencies included in the LINKS_README text file under the libraries [projects/aar]. Otherwise, use the links included in the previously mentioned file and add them as a dependency project on your target project.
7. Clean and rebuild your target project.

7.2 OS permissions

The Android mobile SDK requires the following permissions to work properly:

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

7.3 Using the Android Mobile SDK

7.3.1 Payment Process

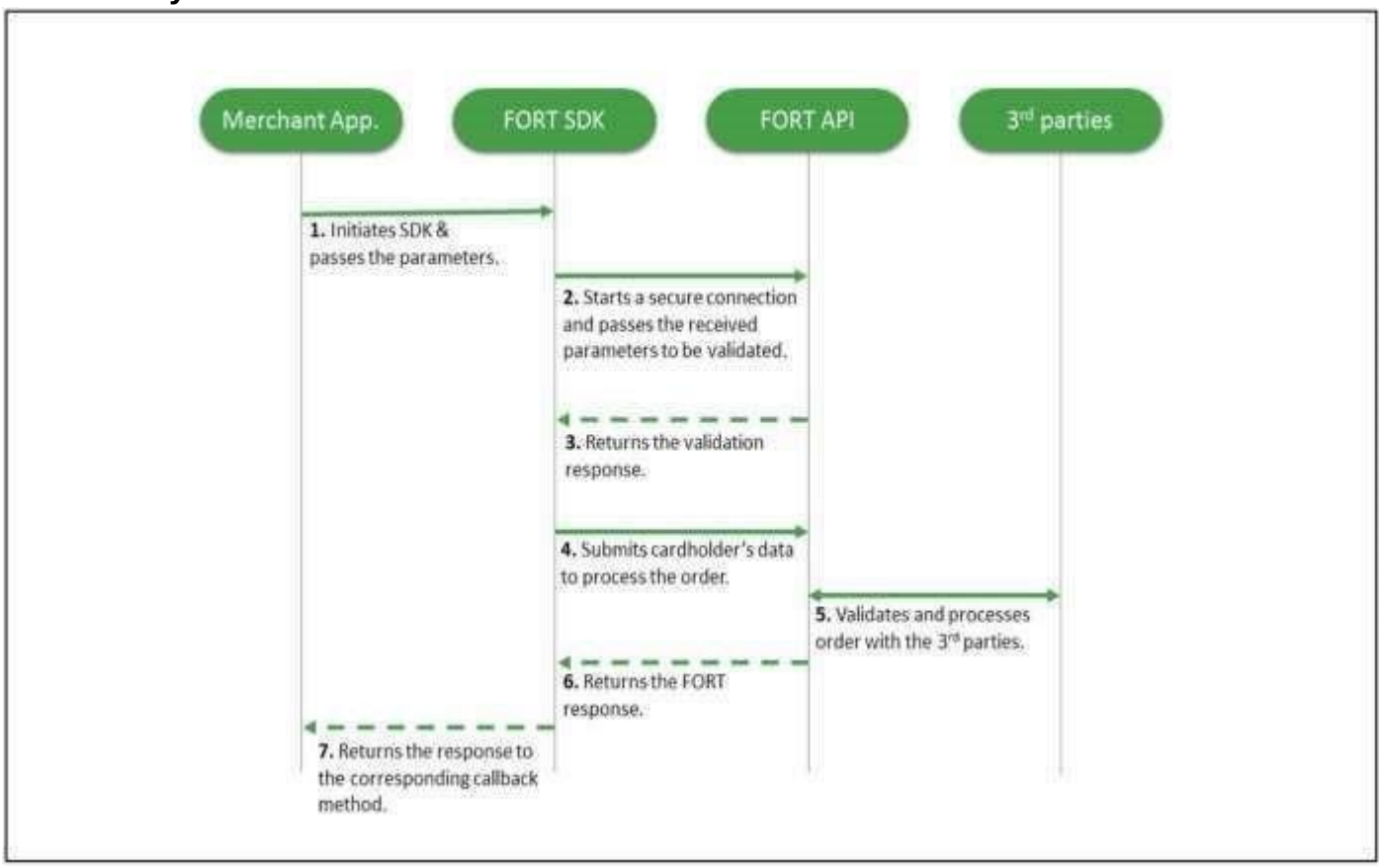

Figure 1: Payment Workflow

Workflow Description:

1. Your application initiates the Android mobile SDK and passes the parameters to the Android mobile SDK.
2. The Android mobile SDK starts a secure connection and passes the received parameters to the Amazon Payment Services API to be validated.
3. The Amazon Payment Services API returns the validation response.
4. The Android Mobile SDK submits the cardholder's data to the Amazon Payment Services API to process the order.
5. The Amazon Payment Services API validates and processes the order with the third parties.
6. The Amazon Payment Services API returns the transaction response.
7. The Android Mobile SDK returns the response to the corresponding callback method.

7.3.2 Collect the Android Mobile SDK Request

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;
   }
}
```

- 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).

- “showResponsePage” is the Boolean field where you can determine if you want the Amazon Payment Services response page to be displayed or not.

7.3.3 Define a Callback Manager

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

```
private FortCallBackManager fortCallback= null; fortCallback
=FortCallback.Factory.create();
```

7.3.4 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);
}
```

7.3.5 Call the Android Mobile SDK

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

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

   @Override public void onSuccess(Map<String, Object> requestParamsMap, Map<String, Object>
fortResponseMap) {
//TODO: handle me
}

   @Override public void onFailure(Map<String, Object> requestParamsMap, Map<String, Object>
fortResponseMap) { //TODO: handle me
}
   @Override public void onSuccess(Map<String, Object> requestParamsMap, Map<String, Object>
fortResponseMap) { //TODO: handle me
}



});
```

The above 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)
```

Mobile SDK Call Parameters

Parameter	Description
context	Passes the currency 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: - ENVIRONMENT.TEST - 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.

7.3.6 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);
```

NOTE!
The Merchant can choose to display the Amazon Payment Services response page by passing "showResponsePage" value as "True".

7.3.7 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:

1. Don’t change the layout name because it's considered an override process.
2. Make sure to use all the views that has the ID property in order to avoid the NullPointerException.
3. Redesign the view for portrait orientation. Note that Landscape orientation isn't supported.
4. You can support as much layout densities as you want.
5. Don’t forget to redesign the layout-ar file too (right-to-left).
6. Don’t change, rename, or remove onClick functions.

Our Mobile SDK v 1.6 consists one of the following three main activities design:

- activity_cc_payment.xml
- activity_cc_response.xml
- activity_init_secure_conn

Every file is available for both English and Arabic alignments; layout and layout-ar.

NOTE!

The files Hierarchy and Content might change in our SDK’s future version

- Make sure to import InputTextLayout from com.google.android.material.textfield.TextInputLayout as we are usign androidx starting from release #1.6

Customization example:

The following image is the standard design and layout of the Android mobile SDK payment:

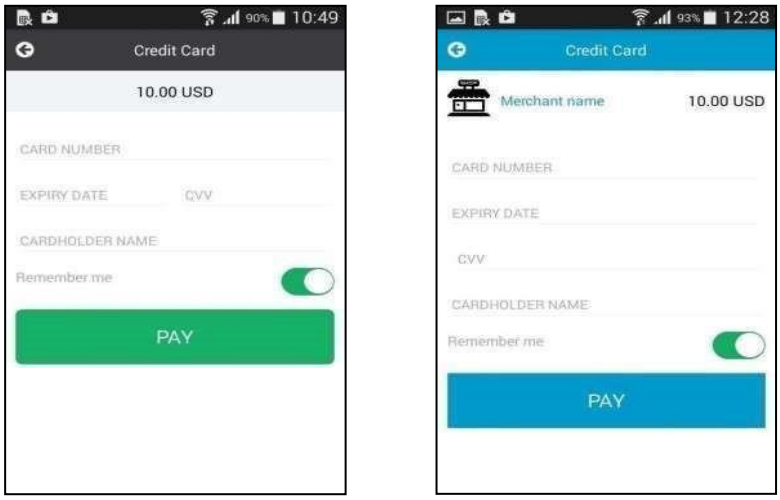

Figure 2: Standard vs. Customized Mobile SDK Payment Page Design

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="@dimen/pf_payment_type_header_height" android:background="@color/pf_light_gray"
android:gravity="center_horizontal|center_vertical" android:textColor="@android:color/black"
android:textSize="@dimen/pf_15_txt_size" />
```

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

```
<LinearLayout android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal" android:padding="10dp"
android:background="@android:color/white">

<ImageView android:layout_width="40dp" android:layout_height="40dp"
android:src="@drawable/merchant_logo"/>

<TextView android:fontFamily="sans-serif-medium"
android:text="Merchant name" android:layout_margin="10dp"
android:textColor="@android:color/holo_blue_dark"
android:layout_width="wrap_content"
android:layout_gravity="center_vertical"
android:layout_height="wrap_content"
android:src="@drawable/merchant_logo"/>

<TextView android:id="@+id/amountTV"
android:layout_width="match_parent"
android:layout_height="@dimen/pf_payment_type_header_height"
android:gravity="right|center_vertical" android:textColor="@android:color/black"
android:text="100 UDS" android:textSize="@dimen/pf_15_txt_size" />

</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, and “TextView“ that contains the Merchant's name.
- 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!

- make sure to retest your custom design, for example, by showing the error messages on fields and applying the changes to the Arabic layout. (Refer to the points mentioned under the Customizing the Mobile SDK Payment Layout section).

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

7.4 Android Mobile SDK Operations

The **Android mobile SDK** allows your application to process authorization and purchase operations.

7.4.1 Request Parameters

Include the following parameters in the request you will send to the Amazon Payment Services SDK:

Request Parameters							
Parameter Name	**Type**	**Mandatory**	**Description**	**Length**	**Special Characters**	**Possible/ Expected Values**	**Example**
command	Alpha	Yes	Command.	20		- AUTHORIZATION - PURCHASE	
merchant_reference	Alphanumeric	Yes	The Merchant's unique order number.	40	- _ .		XYZ9239-yu898
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 transaction's amount in ISO code 3.	3			AED
language	Alpha	Yes	The checkout page and messages language.	2		- en - ar	
customer_email	Alphanumeric	Yes	The customer's email.	254	_ - . @		customer@domain.com
sdk_token	Alphanumeric	Yes	An SDK authentication token to enable using the Android mobile SDK.	100			Dwp78q3

token_name	Alphanumeric	No	The Token received from the Tokenization process.	100	. @ - _		Op9Vmp
payment_option	Alpha	No	Payment option.	10		- MASTERCARD - VISA - AMEX - MADA (for Purchase operations and eci Ecommerce only). Click here to download MADA branding document. - MEEZA (for Purchase operations and ECOMMERCE eci only)	
eci	Alpha	No	E-commerce indicator.	16		9. ECOMMERCE	
order_description	Alphanumeric	No	A description of the order.	150	# ' / . _ - : $ **Space**		iPhone 6S
customer_ip	Alphanumeric	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	+ - (		0096279 7219966
					) **Space**		

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	- _ .		XYZ9239 yu898
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	. ; / _ - , ' @		JohnSmit h
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	. ; / _ - , ' @		JohnSmit h
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	. ; / _ - , ' @		JohnSmit h
merchant_ extra3	Alphanumeric	No	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	. ; / _ - , ' @		JohnSmit h
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	. ; / _ - , ' @		JohnSmit h

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.

7.4.2 Response Parameters

The following parameters will be returned in the Android SDK response:

Response Parameters					
Parameter Name	**Type**	**Description**	**Length**	**Possible/ Expected Values**	**Example**
command	Alpha	Command.	20	- AUTHORIZATION - PURCHASE	
merchant_reference	Alphanumeric	The Merchant's unique order number.	40		XYZ2939 -yu898
amount	Numeric	The transaction's value. *The amount parameter is returned by our system according to the predefined allowed decimal points per currency.	10		10000
currency	Alpha	The currency of the transaction's amount in ISO code 3.	3		AED
customer_email	Alphanumeric	The customer's email.	254		customer @domain .com
fort_id	Numeric	The order's unique reference returned by our system.	20		1492954 3540008 4008
sdk_token	Alphanumeric	An SDK authentication token to enable using the Android Mobile SDK.	100		Dwp78q3

token_name	Alphanumeric	The Token received from the Tokenization process.	100		Op9Vmp

payment_option	Alpha	Payment option.	10	- MASTERCARD - VISA - AMEX - MADA (for Purchase operations and eci Ecommerce only). Click here to download MADA branding document. - MEEZA (for Purchase operations and ECOMMERCE eci only)	
eci	Alpha	E-commerce indicator.	16	- ECOMMERCE	
authorization_code	Alphanumeric	The authorization code returned from the 3rd party.	100	-	P100000 0000000 372136
order_description	Alphanumeric	It holds the description of the order.	150	-	iPhone 6S
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	Alphanumeric	It holds the customer's IP address.	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	Alphanumeric	The customer's phone number.	19	-	0096279 7219966
settlement_reference	Alphanumeric	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	Alphanumeric	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	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	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	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	-	JohnSmith

merchant_extra4	Alphanumeric	Extra data sent by merchant. Will be received and sent back as	250	-	JohnSmith
		received. Will not be displayed in any report.		-	
merchant_extra5	Alphanumeric	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	-	JohnSmith

NOTE!

Every parameter the Merchant sends in the Request should be received by the Merchant in the Response - even the optional ones.

7.5 Amazon Payment Services Transaction Feedback

7.5.1 Overview

The Amazon Payment Services transaction feedback system provides merchants with two types of notifications:

1. Direct Transaction Feedback, Amazon Payment Services will send merchants HTTPs notifications that inform merchants of the transaction's final status whenever a transaction is processed.

2. Notification Transaction Feedback, Amazon Payment Services will send merchants HTTPs notifications that inform merchants of the transaction's final status whenever a transaction status is updated.

7.5.2 Registering Transaction Feedback URLs

1 Log in to your back-office account.

2 Select the active channel under Integration Settings → Technical Settings.

3 Enter your Direct Transaction Feedback URL and Notification Transaction Feedback URL.

4 Click "Save Changes" button.

7.5.3 Transaction Feedback Implementation

The Transaction Feedback URL is required to send the merchant the response parameters after processing the transaction on the merchant's server side.

For the Direct Transaction Feedback, it sends the immediate payments response in all cases, like if the user closed the browser before getting redirected to the redirection URL due to a drop in the internet connection or he closed the browser during the redirection, the merchant will create an endpoint which accepts the notifications received from Amazon Payment Services as POST Method.

For the Notification Transaction Feedback, it's required to provide the merchant the transaction final status update whenever received, like if the transaction was pending due to the unavailability for any party, the final update will be pushed to the Notification Feedback URL as POST Method.
Beyond whatever your Transaction Feedback URL does with the data received, it must also return a 2xx (like 200, 201, etc…) or 302 HTTP status code to update the Amazon Payment Services system that the notification was received. If your URL does not return 2xx or 302, Amazon Payment Services will continue to retry the notification for 10 times with 10 seconds in between until it's properly acknowledged.

Beyond whatever your Transaction Feedback URL does with the data received, it must also return a 2xx (like 200, 201, etc…) or 302 HTTP status code to update Amazon Payment Services that the notification was received. If your URL does not return 2xx or 302, Amazon Payment Services will continue to retry the notification for 10, times with 10 seconds in between until it's properly acknowledged.

NOTE!

- You can check the Direct and Notification Feedback logs in your Amazon Payment Services back office Account to check the details related to the submission like the Transaction Feedback URL which was triggered, The response which our Amazon Payment Services system pushed, The response Code and Status retuned from your Transaction Feedback URL.

- The specifics of the data will differ based upon the financial operation that has been processed. Please refer to the Amazon Payment Services integration guide for more details.

- If you want to change the submission type to JSON or XML, you can contact us on integration@payfort.com.

If you want to change the grace period or the time interval between the retries

7.6 Sample Code

```
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) {

try {
          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) {
                    //TODO: handle me
                    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) {
//TODO: handle me
                    Log.e("Failure ", fortResponseMap.toString());
                }

            });
        } catch (Exception e) {
            Log.e("execute Payment", "call FortSdk", e);
        }

    }



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

