amazon payment services

iOS Merchant Integration Guide

Document Version: 3.92

Oct, 2020

Table of Contents

Contact Us

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

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 iOS and includes information on how to integrate it with your mobile application.

2.1. Intended Audience

This document was created for iOS developers that integrate the Amazon Payment Services iOS 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 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.

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 that allows you to simulate 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 to test your integration before the going live to assure your application integration.

4. About the Software

4.1. Supported Platforms

IOS 8+

4.2. Localization

The Amazon Payment Services iOS 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 iOS SDK, the Merchant has the ability to process a **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. Amazon Payment Services iOS mobile SDK

The Amazon Payment Services iOS mobile SDK allows merchants to securely integrate payment functionality into a native iOS 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 iOS Mobile SDK. In turn, this gives the merchants' customers a smooth, pleasing user experience thanks to in-app payment functions through the native applications.

5.1. Download the iOS Mobile SDK

To download the iOS Mobile SDK, click here.

5.2. Create an iOS 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 with Amazon Payment Services through our iOS mobile SDK.

NOTE!

- A unique authentication token should 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.
- The creation and initiation of a mobile SDK token happens on the merchant's server side.

5.3. iOS mobile SDK token URLs

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

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

5.4. Parameters Submission Type

REST POST request using JSON.

5.4.1. iOS Mobile SDK Token Request Parameters

Include the following parameter in the request you send to Amazon Payment Services:

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			zx0IPmPy5jp1vAz8Kpg7
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	-		ffffffff-a9fa0b44-7b2729e70033c587
signature	Alphanumeric	Yes	A string hashed using the Secure Hash Algorithm. (More details are available in our Merchant Integration Guide).	200			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];

5.4.2. iOS Mobile SDK Token Response Parameters

The parameters will be returned in Amazon Payment Services' response:

Response Parameters						
Parameter Name	**Type**	**Man-datory**	**Description**	**Length**	**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	The ID of the Merchant.	20		CycHZxVj
language	Alpha	Yes	The checkout page and messages language.	2	- en - ar	
device_id	Alphanumeric	Yes	The ID of the used device for this payment.	100		ffffffff-a9fa-0b44-7b2729e70033c5 87
sdk_token	Alphanumeric	Yes	An SDK authentication token to enable using the iOS Mobile SDK.	100		Dwp78q3
signature	Alphanumeric	Yes	A string hashed using the Secure Hash Algorithm. (More details are vailable in our Merchant Integration Guide).	200		7cad05f021 2ed933c9a5 d5dffa31661 acf2c827a
status	Numeric	No	A two-digit numeric value that indicates the status of the transaction.	2	(Please refer to section Statuses).	

response_code	Numeric	No	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
response_message	Alphanumeric	No	Message description of the response code. It returns according to the request language.	150		Insufficient Funds

NOTE!

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

6. Integrate the Amazon Payment Services iOS mobile SDK

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

6.1. Using the iOS mobile SDK

6.1.1. Payment Process

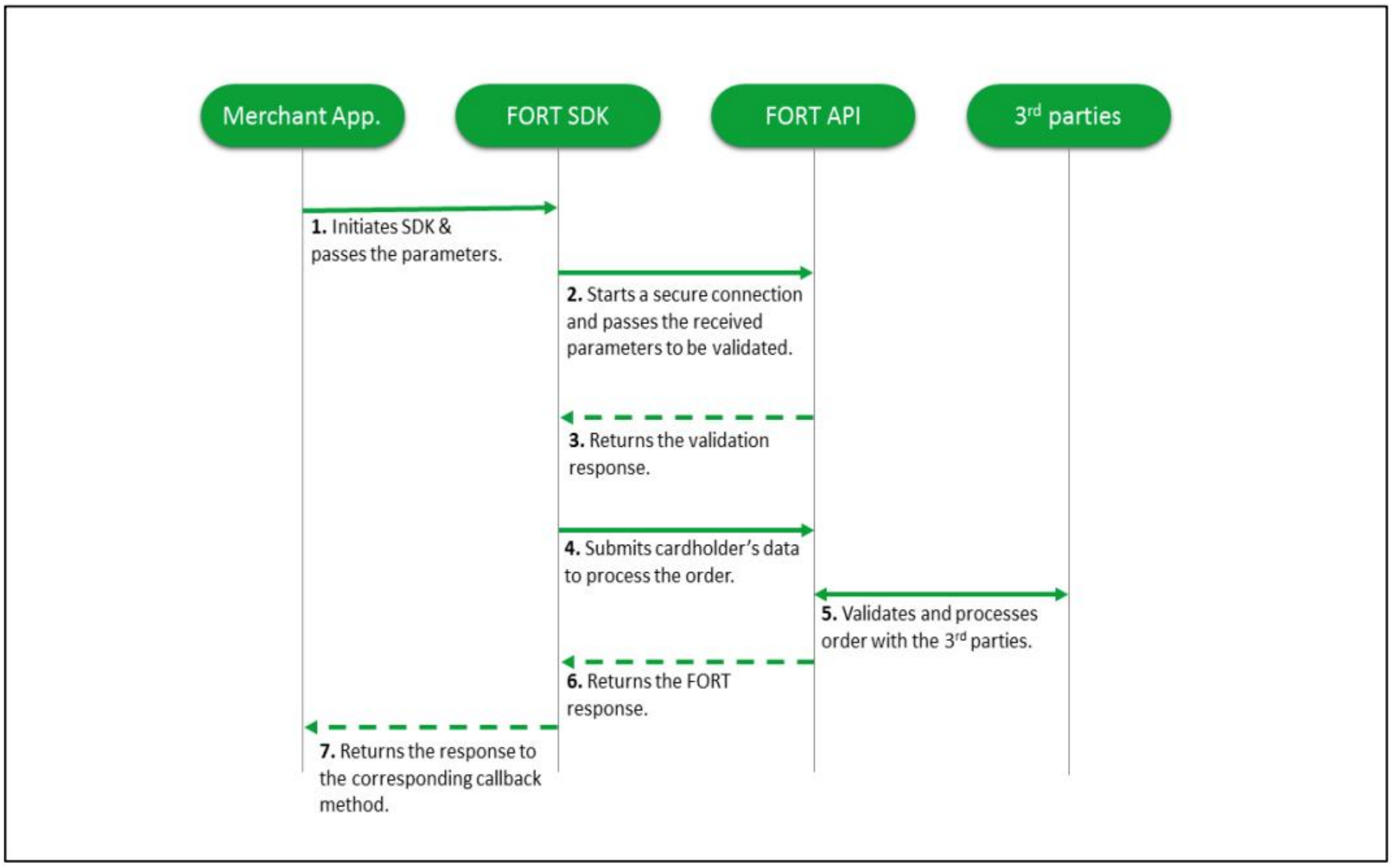

Merchant App.
FORT SDK
FORT API
3rd parties
1. Initiates SDK & passes the parameters.
2. Starts a secure connection and passes the received parameters to be validated.
3. Returns the validation response.
4. Submits cardholder's data to process the order.
5. Validates and processes order with the 3rd parties.
6. Returns the FORT response.
7. Returns the response to the corresponding callback method.

Figure 1: Payment Workflow

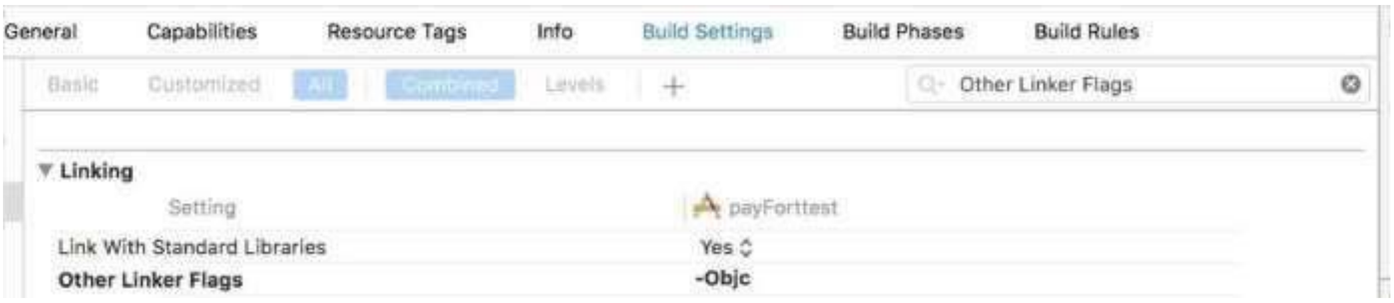
Workflow Description:

1 The merchant's application initiates the Amazon Payment Services iOS mobile SDK and passes the parameters to the iOS mobile SDK.
2 The iOS 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 iOS 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 iOS Mobile SDK returns the response to the corresponding callback method.

6.2. Include the SDK in your Xcode Project

- Extract the folder found in section 5.1
- Drag the PayFortSDK.framework & PayFortSDK.bundle to Frameworks in Project Navigator.
- Create a new group Frameworks if it does not exist.
  - Choose Create groups for any added folders.
  - Make Sure to select Copy files if needed



Link With Standard Libraries: Yes
Other Linker Flags: -Objc

❼ Build Settings Tab.

- Make Sure to select Copy files if needed.
- Set -ObjC in the Other Linker Flags in the Target

For Swift Projects Don't forget to add the

#import <PayFortSDK/PayFortSDK.h> to the Bridging-Header.h

NOTE!
Ensure linked once in the Linked Framework and Libraries or just drag the PayFortSDK.framework to Embedded Binaries in the general tab in the project settings.

NOTE!
In Xcode, secondary-click your project's .plist file and select Open As -> Source Code. Insert the following XML snippet into the body of your file just before the final, same as below:
</dict>element

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key><true/>
</dict>
```

NOTE!
To make the application not disconnected when go to background make sure to add this code:

Objective C:

```
(void)applicationDidEnterBackground:(UIApplication *)application {
 __block UIBackgroundTaskIdentifier backgroundTask;
backgroundTask = [application
beginBackgroundTaskWithExpirationHandler: ^ { [application
endBackgroundTask:backgroundTask];
 backgroundTask = UIBackgroundTaskInvalid; }]; }
```

```
Swift :
func applicationDidEnterBackground(_ application: UIApplication)
            {
            var bgTask: UIBackgroundTaskIdentifier = 0 bgTask =
            application.beginBackgroundTask(expirationHandler:
            { application.endBackgroundTask(bgTask) bgTask =
            UIBackgroundTaskInvalid
            })
}
```

6.3. Change present style

To change the present style from (Full Screen) to (Default) use the following property.

NOTE!
The default type is full screen when you set the value to false, it's will appear as OS default**.**

- Objective C

```
PayFort.presentAsDefault = YES;
```

- Swift

```
PayFort.presentAsDefault = YES;
```

6.4. Installation

1. Import the Amazon Payment Services library.
   `#import <PayFortSDK/PayFortSDK.h>`
2. Initialize PayFortController with targeted environment, You set the target environment by setting one
   the two ENUM KPayFortEnviromentSandBox or KPayFortEnviromentProduction

- Objective C

```
PayFortController *payFort = [[PayFortControlleralloc]initWithEnviroment:KPayFortEnviroment
SandBox];
```

- Swift

```
Let payFort=PayFortController.init(enviroment:KPayFortEnviromentSandBox)
```

3. Set Dictionary contain all keys and values for SDK

- Objective C

```
NSMutableDictionary *request = [[NSMutableDictionary alloc]init];
[request setValue:@"10000" forKey:@"amount"];

[request setValue:@"AUTHORIZATION" forKey:@"command"];
[request setValue:@"USD" forKey:@"currency"];
[request setValue:@ "email@domain.com" forKey:@"customer_email"];
```

```
[request setValue:@"en" forKey:@"language"];
[request setValue:@"112233682686" forKey:@"merchant_reference"]; [request setValue:`SDK TOKEN GOES HERE`
forKey:@"sdk_token"]; [request setValue:@"" forKey:@"payment_option"];
[request setValue:@"gr66zzwW9" forKey:@“token_name"];
```

- Swift

```
let request = NSMutableDictionary.init() request.setValue("1000", forKey: "amount")
request.setValue("AUTHORIZATION", forKey: "command") request.setValue("USD",
forKey: "currency") request.setValue("email@domain.com", forKey:
"customer_email") request.setValue("en", forKey: "language")
request.setValue("112233682686", forKey: "merchant_reference")
request.setValue("token" , forKey: "sdk_token")
```

4. Call Amazon Payment Services and response callback

- Objective C

```
[payFort callPayFortWithRequest:request currentViewController:self
                                Success:^(NSDictionary *requestDic, NSDictionary *responeDic) {
                                  NSLog(@"Success");
                                  NSLog(@"responeDic=%@",responeDic);
                               }
                               Canceled:^(NSDictionary *requestDic, NSDictionary *responeDic) {
                                 NSLog(@"Canceled");
                                 NSLog(@"responeDic=%@",responeDic);
                               }
                               Faild:^(NSDictionary *requestDic, NSDictionary *responeDic, NSString *message) {
NSLog(@"Faild");
                                 NSLog(@"responeDic=%@",responeDic);
                               }];
```

- Swift

```
 PayFort.callPayFort(withRequest: request, currentViewController: self,
           success: { (requestDic, responeDic) in    print("success")
},
                                 canceled: { (requestDic, responeDic) in
print("canceled")
        },                                  faild: { (requestDic, responeDic,
message) in print("faild")
        })
```

6.5. iOS SDK response

By default the response will be dictionary to show the sent data in addition to the status, response message and response code.

The response will be ready in the registered call back handler with success, failed and cancelled. You can view the response by log the result as the followings:

- Objective

```
[payFort callPayFortWithRequest:request currentViewController:self
                              Success:^(NSDictionary *requestDic, NSDictionary *responeDic) {
                                NSLog(@"Success");
                                NSLog(@"requestDic=%@",requestDic);
                                NSLog(@"responeDic=%@",responeDic);
                            }
                             Canceled:^(NSDictionary *requestDic, NSDictionary *responeDic) {
                                NSLog(@"Canceled");
                                NSLog(@"requestDic=%@",requestDic);
                                NSLog(@"responeDic=%@",responeDic);
                            }
                             Faild:^(NSDictionary *requestDic, NSDictionary *responeDic, NSString *message) {
NSLog(@"Faild");
                                NSLog(@"requestDic=%@",requestDic);
                                NSLog(@"responeDic=%@",responeDic);
                                NSLog(@"message=%@",message);
                            }];
```

- Swift

```
PayFort.callPayFort(withRequest: request,                  currentViewController: self,
success: { (requestDic, responeDic) in                         print("success")
print("responeDic=\(responeDic)")                       print("responeDic=\(responeDic)")
},
canceled: { (requestDic, responeDic) in
print("canceled")           print("requestDic=\(requestDic)")
print("responeDic=\(responeDic)")
       },                                  faild: { (requestDic, responeDic,
message) in
print("faild")       print("requestDic=\(requestDic)") print("responeDic=\(responeDic)")
            print("message=\(message)")
})
```

Also, there is an option to show response view directly in elegant view that show response results either its success or failed. By activating the following option:

- Objective C

```
PayFort.IsShowResponsePage =
YES;
```

- Swift

```
PayFort.IsShowResponsePage = true;
```

6.6. Hide Amazon Payment Services loading prompt

There is an option to hide the loading prompt when the iOS SDK initializes the connection request. You can disable the loading prompt by using following option:

- Objective C

```
PayFort.HideLoading = YES;
```

- Swift

```
PayFort.HideLoading =
true;
```

6.7. Customizing the payment UI

You have the option to design a custom UI theme for the payment view by doing the followings:

- Create your nibFile .xib and set the name of Arabic xib same name with English one with suffix -ar.
- Link the xib with PayFortView and bind all the IBOutlets in interface section

```
IBOutlet UILabel *titleLbl;
IBOutlet UIButton *BackBtn;
IBOutlet UILabel *PriceLbl;
IBOutlet JVFloatLabeledTextField *CardNameTxt;
IBOutlet JVFloatLabeledTextField *CardNumberTxt;
IBOutlet JVFloatLabeledTextField *CVCNumberTxt;
IBOutlet JVFloatLabeledTextField *ExpDateTxt;
IBOutlet UILabel *cardNumberErrorlbl;
IBOutlet UILabel *cVCNumberErrorlbl;
IBOutlet UILabel *expDateErrorlbl;
IBOutlet UISwitch *savedCardSwitch;
IBOutlet UIButton *paymentBtn;
IBOutlet UILabel *saveCardLbl;
IBOutlet UIImageView
*imageCard;
```

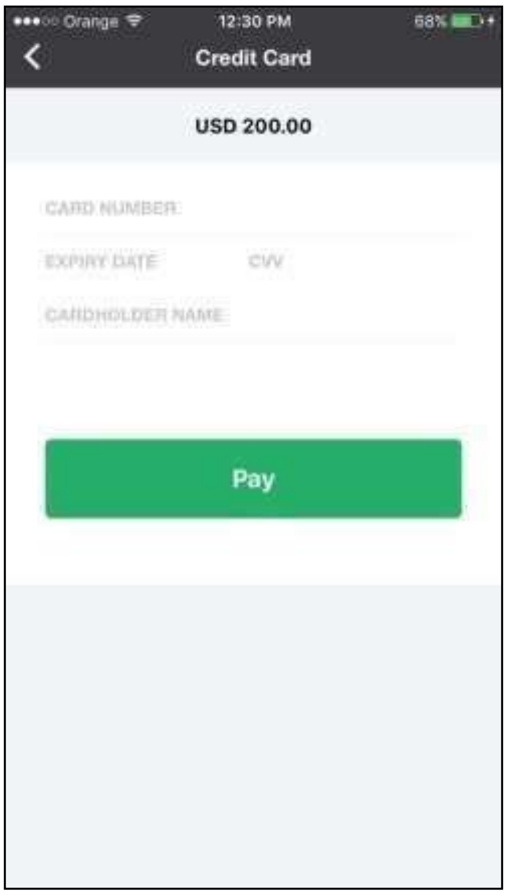
- Assign new created xib file to Amazon Payment Services controller.

```
[payFort setPayFortCustomViewNib:@"PayFortView2"];
```

NOTE!
If you call Arabic view and the Arabic view not existed the application will crash.
Don’t forget to set the custom view field in the identity inspector

Customization example:

The following image is the standard design and layout of the iOS SDK payment page:



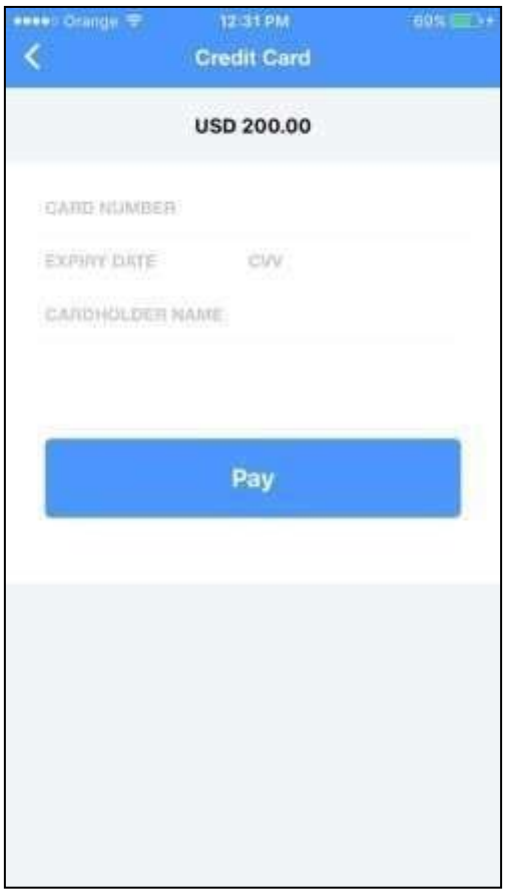


Figure 2: Standard vs. Customized Mobile SDK Payment Page Design

6.8. iOS Mobile SDK operations

The iOS Mobile SDK allows the merchant's application to process authorization and purchase operations.

6.8.1. Request Parameters

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

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	- _		XYZ9239yu898

amount	Numeric	Yes	*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	Alphanu meric	Yes	An SDK token to enable using the iOS Mobile SDK.	100			Dwp78q3
token_name	Alphanumeric	No	The Token received from the Tokenization process	100	. @ - _		Op9Vmp
payment_option	Alpha	No	Payment option.	10		- VISA - MASTERCARD - 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		ECOMMERCE	
order_description	Alphanumeric	No	It holds the description of the order	150	# ' / . _		iPhone 6-S

					- : $ **Space**		
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' sname.	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	250	; / _		JohnSmith

			received and sent back as received. Will not be displayed in any report.		- , ' @		
merchant_ extra2	Alphan umeric	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	Alphan umeric	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	Alphan umeric	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	Alphan umeric	No	Extra data sent by merchant. Will be received and sent back as received. Will not be displayed in any report.	250	; / _ - , ' @		JohnSmit h

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.

6.8.2. Response Parameters

The following parameters will be returned in Amazon Payment Services response:

		Response Parameters			
Parameter Name	**Type**	**Description**	**Length**	**Possible/ Expected Values**	**Example**
command	Alpha	Command.	20	- AUTHORIZATION - PURCHASE	
merchant_reference	Alphanu meric	The Merchant's unique order number.	40		XYZ2939yu898
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		1443796866848
sdk_token	Alphanumeric	An SDK token to enable using the Amazon Payment Services mobile SDK.	100		Dwp78q3
token_name	Alphanumeric	The Token received from the Tokenization process.	100		Op9Vmp
payment_option	Alpha	Payment option.	10	- VISA - MASTERCARD - 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	-	P1000000000000372136

order_description	Alphanumeric	It holds the description of the order.	150	-	iPhone 6-S
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.	5	- (Please refer to section Messages).	
status	Numeric	A two-digit numeric value that indicates the status of the transaction.	2	(Please refer to section Statuses).	
customer_ip	Alphanumeric	It holds the customer's IP address.	45		192.178.1.10
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
customer_name	Alpha	The customer's name.	40		John Smith
phone_number	Alphanumeric	The customer's phone number.	19		00962797219966
settlement_referenc e	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		XYZ9239yu898
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 received. Will not be displayed in any report.	250		JohnSmith
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

6.9. Amazon Payment Services transaction feedback

6.9.1. Overview

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

1. Direct Transaction Feedback, Amazon Payment Services sends merchants HTTPs notifications that inform merchants of the transaction's final status whenever a transaction is processed.
2. Notification Transaction Feedback, Amazon Payment Services sends merchants HTTPs notifications that informs the merchant of the transaction's final status whenever a transaction status is updated.

6.9.2. 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.

6.9.3. 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' 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

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.
- 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 please contact us on integration@payfort.com

6.10. Sample Code

6.10.1. Initialize the Mobile SDK

- Objective C:

```
PayFortController *payFort = [[PayFortController
alloc]initWithEnviroment:KPayFortEnviromentSandBox];

//if you need to switch on the Payfort Response page payFort.IsShowResponsePage = YES;

//Generate the request dictionary as follow
NSMutableDictionary *requestDictionary = [[NSMutableDictionary alloc]init];
[requestDictionary setValue:@"10000" forKey:@"amount"];
[requestDictionary setValue:@"AUTHORIZATION" forKey:@"command"];
[requestDictionary setValue:@"USD" forKey:@"currency"];
[requestDictionary setValue:@"email@domain.com" forKey:@"customer_email"];
[requestDictionary setValue:@"en" forKey:@"language"];
[requestDictionary setValue:@"112233682686" forKey:@"merchant_reference"];
[requestDictionary setValue:@"" forKey:@"payment_option"];
[requestDictionary setValue:@"gr66zzwW9" forKey:@"token_name"];

[payFort callPayFortWithRequest:requestDictionary currentViewController:self
Success:^(NSDictionary *requestDic, NSDictionary *responeDic) {

} Canceled:^(NSDictionary *requestDic, NSDictionary *responeDic) {
} Faild:^(NSDictionary *requestDic, NSDictionary *responeDic, NSString *message) {

}];
```

- Swift :

```
let payFort = PayFortController.init(enviroment: KPayFortEnviromentSandBox)

//if you need to switch on the Payfort Response page paycontroller.isShowResponsePage = true
let request = NSMutableDictionary.init() request.setValue("1000", forKey: "amount")
request.setValue("AUTHORIZATION", forKey: "command")
request.setValue("USD", forKey: "currency")
request.setValue("email@domain.com", forKey: "customer_email") request.setValue("en", forKey: "language")
request.setValue("112233682686", forKey: "merchant_reference")
request.setValue("gr66zzwW9" , forKey: "token_name") request.setValue("" , forKey: "payment_option")
payFort.callPayFort(withRequest: request, currentViewController: self, success: { (requestDic, responeDic) in
 },
canceled: { (requestDic, responeDic) in
 },
faild: { (requestDic, responeDic, message) in

})
}
```