ARB PAYMENT GATEWAY REST API MOF JS WIDGET INTEGRATION

Version 1.0

Publishing Date: March 2023

REST API MOF JS Widget Integration Guide

مصرف الراجحي
Al Rajhi Bank

Disclosure and Confidentiality

Disclosure Agreement

This document is a private document and Bank should disclose this document fully or partially, to third party/merchants only when the required NDA (Non-disclosure Agreement) is signed.

For the purposes of this rule, proprietary and classified information shall include, but not be limited to, the following:

- All information concerning the technical standards and procedures used in implementing and operating the Service and the Network.
- Information not previously disclosed to the public concerning any member or Service that is under consideration for inclusion in the operation of the Network.

Version History

The following table displays the version history of this document:

Version No.	Created or Updated By	Reason for Change	Created or Updated on
1.0	Al Rajhi Bank	Document created	March 2023

Table of Contents

Chapter 1 ABOUT THIS DOCUMENT

This document covers the purpose of the document, target audience and conventions used.

Purpose of the Document

This reference document is being shared to the AL Rajhi Bank merchants who are integrated in a secure and mandated manner.

The expectation is that the merchant's system integrator or auditor can refer to this document while performing integration as well as post integration. This document contains the technical integration details including message formats to be used in communicating to the Payment Gateway irrespective of the merchant platform being used. The document also shares the best practices and recommendations the merchant should follow during the integration with Payment Gateway.

The Payment Gateway follows industry standards and norms as prescribed by MasterCard and Visa International in conformity with Payment Card Industry – Data Security Standards commonly referred to as PCI – DSS.

Chapter 2 MERCHANT PRE-REQUISITES

This chapter covers the Hardware and Software pre-requisites for the Merchant integration. This pre-requisite for this document user is to be familiar with languages such as JSP, ASP .Net, PHP and Java.

Hardware Pre-requisites

Merchants can use their existing hardware for transaction processing via ARB Payment Gateway. Merchants may have a variety of arrangements for hosting their websites and thus have relevant security mandates for internet access controls and checks. This may include utilization of a Proxy Server which presents informed challenges. It is recommended that the merchant use a Public IP during the integration testing for transaction processing to the ARB Payment Gateway.

Software Pre-requisite

The merchant should have the requisite software for connecting to the ARB Payment Gateway depending on the merchant application environment. The merchant may use combinations of OS/Web Server/Application server whilst setting up and operating the website. Standard Software options are listed below, this list is for reference use only.

Operating Systems: Windows 2000/2003/2008 Server, Sun Solaris, IBM AIX
Web/Application Servers-Web/Application Server that support JSP & ASP .Net.

The current version with all required patches is recommended to ensure success. Software Installation – Basic software that are required for Web/Application server should be installed at the merchant site. (Java/JDK for JSP integration is essential; similarly, .NET frame work is essential for ASP.NET integration)

Chapter 3 MERCHANT INTEGRATION PROCESS

This Chapter covers integration process of the Merchant with ARB Payment Gateway Plug-in.

Integration Pre-requisites

Below are the pre-requisites for merchants to integrate with ARB Payment gateway. ARB will share the below parameters to merchant manually for REST API based integration.

Parameter	Sample Value	Description
Id	IPAYlCR6qZF7q6w	Tranportal ID - Unique ID for each terminal.
Password	q@a68O$27@JLkcK	Tranportal Password - Used for authentication of the terminal.
Key	309897493722309897493734	Resource Key - Request data needs to be encrypted with this key.
End point URL	https://securepayments.alrajhibank.com.sa/pg/payment/hosted.htm	End point URL - Merchant needs to send re- quest to this URL.

Integration Process

Steps to be followed by Merchant Integration Team:

1. Frame the request of plain trandata with request parameters as specified in the document.
2. Encrypt the plain request with the resource key using the AES algorithm encryption logic specified in the document.
3. Frame the final URL using given end point URL and encrypted request to connect to ARB Payment Gateway.

ARB Payment Gateway Transaction Flow

Payment Gateway uses AES encryption algorithm to encrypt transaction data which includes all sensitive information.

Transaction Flow for Bank Hosted Setup

- Customer completes the purchase and proceeds to checkout with payment options.
- Merchant will be redirected to Bank's payment page, wherein customer inputs the card details on Bank's page.
- Card details are validated by ARB PG.
- Payment Gateway will interface with respective schemes like MasterCard, VISA to process the 3D secure authentication.
- Once respective scheme/ACS complete the authentication.
- Once the authentication step is completed, Payment gateway completes the transaction authorization and provides the response to the customer.

Bank Hosted Integration

Mode of communication would be HTTP URL Redirection.

The below are the parameters need to be set in merchant request:

Transaction Action Type: Describes the received transaction type (ex: 1 – Purchase)

Recipient/Response URL: Stores the response URL (ex:

http://www.merchantdemo.com/RAW/jsp/HostedPaymentResultHTTP.jsp)

Error URL: Stores the response URL (ex:

http://www.merchantdemo.com/RAW/jsp/HostedPaymentResultHTTP.jsp*)*

- **Amount:** Stores the currency value of the selected product/entity (ex: SR. 100.00).
- **Currency:** Always 682.
- **Track ID:** Allows you to map track ID to every transaction (ex: 112312312313)
- **Tranportal ID:** Unique ID for a terminal.
- **Tranportal Password:** Used for authentication of the terminal.
- **Bill Details:** Contains billing details of the MOF Agencies.
- Payor ID Type
- Payor ID Number

- Below are the sample JSON request and response, Request from Merchant to ARB Payment gateway:

```
[{

//Mandatory Parameters "id":"IPAYlCR6qZF7q6w",
“trandata”:”<encrypted trandata> ”,
"responseURL":"https://merchantpage/PaymentResult.jsp",
"errorURL":"https://merchantpage/PaymentResult.jsp"
}]
```

- Trandata will contain below parameters encrypted with AES algorithm with CBC Mode, PKCS5Pad- ding with initialization vector value PGKEYENCDECIVSPC under Resource Key

Plain Trandata:

```
[{ “amt”:”12.00”,
”action”:”1”, ”password”:”q@a68O$27@JLkcK”,
”id”:”IPAYlCR6qZF7q6w”,
”currencyCode”:”682”,
”trackId”:”12345656789”,
”responseURL”:”https://merchantpage/PaymentResult.jsp”,
”errorURL”:”https://merchantpage/PaymentResult.jsp”, ”udf1”:”udf1text”,
”udf2”:”udf2text”,
”udf3”:”JSWIDGET”,
”udf4”:”udf4text”,
”udf5”:”udf5text”,
"payorIDType":"NAT",
"payorIDNumber":"1234567891",
"billDetails":
“[{\"issuerAgencyId\":\"789456\",\"billingAccountId\":\"123456\",\"billing
Cycle\":\"123456\",\"dueAmount\":\"10.00\",\"paidAmount\":\"10.00\",\"bill
ReferenceInfo\":\"123456\",\”agencyCode\”:\”1234\”},{\"issuerAgencyId\":\"
789456\",\"billingAccountId\":\"123456\",\"billingCycle\":\"123456\",\"due
Amount\":\"10.00\",\"paidAmount\":\"10.00\",\"billReferenceInfo\":\"123456
\",\”agencyCode\”:\”1234\”},{\"issuerAgencyId\":\"789456\",\"billingAccoun
tId\":\"123456\",\"billingCycle\":\"123456\",\"dueAmount\":\"10.00\",\"pai
dAmount\":\"10.00\",\"billReferenceInfo\":\"123456\",\”agencyCode\”:\”1234
\”}]”
}]
```

Request from Merchant to ARB Payment gateway

S. No	Fields	M/C/O	Field Type	Description
1	trandata	M	Alphanum	All the below request parameters to be encrypted and pass the encrypted value in trandata.
2	id	M	Alphanum	Tranportal ID. Unique ID for a terminal.
3	responseURL	M	Alphanum	The merchant Response URL
4	errorURL	M	Alphanum	The merchant Error URL which contains initial level Failures

Detailed description of Plain Trandata request parameters

S. No	Fields	M/ C/ O	MAX	Field Type	Occurrence	Description
1	amt	M	12	Numeric	1	Transaction amount
2	action	M	10	Numeric	1	It defines the transactions actions Purchase: 1
3	password	M	15	Alphanum	1	Tranportal password. Used for authentication of the terminal.
4	id	M	15	Alphanum	1	Tranportal ID. Unique ID for a terminal.
5	currencycode	M	3	Numeric	1	3-digit currency code of KSA. Ex:682
6	trackid	M	255	Numeric	1	Merchant unique reference no
7	udf1	-O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a trans-action request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.

S. No	Fields	M/ C/ O	MAX	Field Type	Occurrence	Description
8	udf2	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
9	udf3	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should pass the value as "JSWIDGET".
10	udf4	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
11	udf5	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
12	udf6	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the Transaction response. Merchant should ensure that field is left blank when no data needs to be passed.

S. No	Fields	M/ C/ O	MAX	Field Type	Occurrence	Description
13	udf7	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
14	udf8	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
15	udf9	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
16	udf10	O	255	Alphanum	1	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
17	langid	O	50	Alphabetic	1	Language ID

S. No	Fields	M/ C/ O	MAX	Field Type	Occurrence	Description
18	payorIDType	M		Alphanumeric	1	• Payor ID Type Ex: NAT • NAT – National ID • IQA – IQAMA ID • BIS – Business ID • C700 – 700 Code • UOI – Unified Organization ID • GCC – GCC Passport Number • PAS – Passport Number • BDN – Border Number • FCN – Family Card Number
19	PayorIDNumber	M		Numeric	1	• Payor ID Number Ex: 1234567891 • For NAT, id number value should start with 1 and value length should be 10 dig- its • For IQA, id number value should start with 2 and value length should be 10 dig- its • For BIS, id number length should be 10 digits • For C700, id number length should be 10 digits • For the other id types, the id number should be between 6 and 15
20	billDetails	M		String	1-20	Contains Bill details of the Agencies in JSON Array Format
21	responseURL	M		Alphanum	1	The merchant Response URL
22	errorURL	M		Alphanum	1	The merchant Error URL which contains initial level Failures

Bill Details Description:

S.No	Fields	M/C/O	Field Type	Length	Occurence	Owner
1	issuerAgencyId	M	String	18	1	Agency
2	billingAccountId	M	String	6-20	1	Agency
3	billingCycle	O	Numeric	1-12	1	Agency
4	dueAmount	M	Money		1	Agency
5	paidAmount	M	Money		1	Agency
6	billReferenceInfo	O	Character	[Max 80]	1	Agency
7	agencyCode	M	String	4	1	Agency

Note – The Field Name should be same as mentioned in the formats, any difference (including spell mistake) would lead to error in processing the transaction.

- ARB Payment gateway internally validates the request and gives payment ID and payment page URL in the response in case of successful validation, if failure then error code and description will be provided. The below response will be in plain format and there is no encryption for the below. Merchant can directly parse the response-based status and result fields as mentioned below.

Success:

```
[{
"status": "1",
"result":"100201931620827468:https://securepayments.alrajhibank.com.sa/pg/paymentpage.htm", //Payment ID:Paymentpage URL
"error": null, "errorText": null }] Failure:
[{
"status": "2",
"error":" IPAY0100124",
"errorText": "Problem occurred while validating transaction data", "result": null
}]
```

Initial Response from PG to Merchant

S. No	Fields	M/C/O	Field Type	Description
1	status	M	Numeric	If the request validation success, then status will be '1'. If the validation failed, then status will be '2'
2	result	C	Alphanum	It contains payment ID and Payment URL if the validation success else this will be null
3	error	C	Alphanum	If validation failed, then Payment gateway will provide the respective error code
4	errorText	C	Alphanum	If validation failed, then Payment gateway will provide the respective error description

- If success, Merchant needs to frame the payment page URL like the below sample

After Initial Response from ARB PG, merchant needs to frame the payment JSON data like the below sample.

```
JSONObject json = new JSONObject();


json.put("contextPath","https://securepayments.alrajhibank.com.sa/pg");
json.put("paymentId",PaymentID);
json.put("merchantResponseURL","https://securepayments.alrajhibank.com.sa/ac
sWidgetDemo/jsp/fss/HostedPaymentResult.jsp");
```

1. Merchant should add the below mentioned code in their webpage. Payment page will get loaded in this div component.

```
<div id=”payment-widget-container” class=”col-6 payment-widget-container”>


</div>
```

2. Merchant should set payment values and customization values in merchant webpage as below mentioned format.

Sample request:

```
<script>
const reqBody={


"contextPath" : "https://securepayments.alrajhibank.com.sa/pg", // Fixed
Value


"paymentId" : "600202014526378912",      // Dynamic value for each
transaction as received from the First API call
```

```
"merchantResponseURL":"https://securepayments.alrajhi-
bank.com.sa/acsWidgetDemo/jsp/fss/HostedPaymentResult.jsp" //Merchant
Response Page URL
}
</script>
```

1. Merchant should import the js plug-in url as mentioned below once the above parameters is set.

```
<script src="https://securepayments.alrajhibank.com.sa/pg/payment-
widget/js/functions.js"></script>

<script src="https://securepayments.alrajhibank.com.sa/pg/payment-
widget/js/tom-select.complete.min.js"></script>

<script type="text/javascript">payload();</script>
```

ARB PG server will process the request and displays the 'Payment Page' with the following mandatory field to customer:

- Card Holder Name
- Card Number
- CVV
- Card Expiry Month
- Card Expiry Year
- Payor ID Type (National ID Card Type)
- Payor ID Number (National ID Number)

After customer provides all the mandatory details and clicks submit in the 'Payment Page', ARB PG will process the payment and Merchant/ARB PG displays the response to the customer based on Merchant requirement.

- ARB Payment gateway will provide the final response in URL redirection. Below is the sample response from ARB PG to merchant

//Redirection Parameters

- "paymentId":"100201935166676976",
- "trandata":"<encrypted trandata>",
- "error":"",
- "errorText":""

Merchant needs to decrypt the above parameter

Plain Trandata:

```
[{ “paymentId”:”100201935166676976”, ”result”:”CAPTURED”,
”transId”:201935166561122, ”ref”:”935110000001”,
 ”date”:”1217”,
 ”trackId”:”1003383844”,
 ”udf1”:””,
”udf2”:””, ”udf3”:”8870091137”, ”udf4”:”FC”,
”udf5”:”Tidal5”,
”amt”:”70.0”,
”authRespCode”,”00”, "authCode":"000000", "cardType":"Visa"
}]
```


Detailed description of response from ARB payment gateway to Merchant:

S. No	Fields	M/C/O	Field Type	Description
1	paymentId	M	Numeric	Unique payment Id generated by PG and merchant can use this ID to match the response from PG
2	trandata	C	Alphanum	All the below response parameters are encrypted and sent as trandata
3	error	C	Alphanum	If any error, PG will send the error code
4	errorText	C	Alphanum	If any error, PG will send the error description

Detailed description of Plain trandata parameters

S. No	Fields	M/C/O	Field Type	Description
1	paymentId	M	Numeric	Unique ID generated by payment gateway.
2	result	M	Alphanum	Transaction status
3	ref	M	Numeric	Transaction reference number (RRN)
4	transId	M	Numeric	Unique transaction Id generated by Payment gateway and merchant can use this id for initiating supported transactions (Void, refund and inquiry)
5	date	M	Numeric	Transaction date and time
6	trackId	M	Numeric	Merchant unique reference no

S. No	Fields	M/C/O	Field Type	Description
7	udf1	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
8	udf2	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
9	udf3	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
10	udf4	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
11	udf5	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
12	udf6	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction re- quest and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
13	udf7	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction re- quest and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
14	udf8	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
15	udf9	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.

S. No	Fields	M/C/O	Field Type	Description
16	udf10	O	Alphanum	The user (merchant) defines these fields. The field data is passed along with a transaction request and then returned in the transaction response. Merchant should ensure that field is left blank when no data needs to be passed.
17	amt	M	Numeric	Transaction amount
18	authRespCode	M	Numeric	Auth response code provided by PG
19	auth	M	Numeric	6 digit authorization code
20	cardType	M	Alphabetic	Card Brand name Value will be "Visa" or "MasterCard" or "Mada".

Encryption And Decryption Code

Sample Encryption and Decryption Code for JAVA

```
public static String encryptAES(String key,String encryptString) throws
Exception{


String AES_IV = "PGKEYENCDECIVSPC";

Byte [] encryptedText=null; IvParameterSpec ivspec=null; SecretKeySpec
skeySpec=null; Cipher cipher=null;

Byte [] text=null;

String s=null;

try {

ivspec = new IvParameterSpec(AES_IV.getBytes("UTF-8")); skeySpec = new
SecretKeySpec(key.getBytes("UTF-8"), "AES"); cipher =
Cipher.getInstance("AES/CBC/PKCS5Padding"); cipher.init(Cipher.ENCRYPT_MODE,
skeySpec,ivspec);

text = encryptString.getBytes("UTF-8"); encryptedText =
cipher.doFinal(text);

s = byteArrayToHexString(encryptedText);

} catch (Exception e) { e.printStackTrace();
```

```
}
finally
{
encryptedText=null; ivspec=null; skeySpec=null; cipher=null; text=null;
}
return s.toUpperCase();
}


public static String decryptAES(String key,String encryptedString) throws
Exception{
String AES_IV = "PGKEYENCDECIVSPC";
SecretKeySpec skeySpec=null; IvParameterSpec ivspec=null; Cipher cipher
=null;


Byte [] textDecrypted=null; Try {
Byte [] b = hexStringToByteArray(encryptedString); skeySpec = new
SecretKeySpec(key.getBytes("UTF-8"),"AES"); ivspec = new
IvParameterSpec(AES_IV.getBytes("UTF-8")); cipher =
Cipher.getInstance("AES/CBC/PKCS5Padding"); cipher.init(Cipher.DECRYPT_MODE,
skeySpec,ivspec); textDecrypted = cipher.doFinal(b);


} catch (Exception e) { e.printStackTrace();


}
{


}



skeySpec=null; ivspec=null; cipher =null;
return(new String(textDecrypted));
}
```

Sample Encryption and Decryption Code For JAVASCRIPT

```
function aesEncrypt(trandata,key)
{
var iv = "PGKEYENCDECIVSPC";
var rkEncryptionIv = aesjs.utils.utf8.toBytes(iv); var enckey=
aesjs.utils.utf8.toBytes(key);
var aesCtr = new aesjs.ModeOfOperation.cbc(enckey, rkEncryptionIv); var
textBytes = aesjs.utils.utf8.toBytes(trandata);
var encryptedBytes = aesCtr.encrypt(aesjs.padding.pkcs7.pad(text    Bytes));
var encryptedHex = aesjs.utils.hex.fromBytes(encryptedBytes);
return encryptedHex;
}
function AESdecryption(encryptedHex,key)
{
var iv = "PGKEYENCDECIVSPC";
var enckey= aesjs.utils.utf8.toBytes(key);
var rkEncryptionIv = aesjs.utils.utf8.toBytes(iv);
var encryptedBytes = aesjs.utils.hex.toBytes(encryptedHex);
var aesCbc = new aesjs.ModeOfOperation.cbc(enckey, rkEncryptionIv); var
decryptedBytes = aesCbc.decrypt(encryptedBytes);
var decryptedText = aesjs.utils.utf8.fromBytes(decryptedBytes); return
decryptedText;
}
```