AKGPay
Payment Gateway
Non Seamless

1. INTRODUCTION:

As we all already know how APIs work and are integrated. With APIs, applications talk to each other without any user knowledge or intervention. AKGPay provides a simple interface listing information of interest to perform transactions.

This interface can be accessed by Url << https://onboarding.akgpay.com-- >>. AKGPay payment gateway links merchants to process online payments by providing all major payment methods.

2. PRE-REQUISITE:

Credentials – Merchant-related credentials that are required for dashboard login, payment request generation, etc.

SNo	Name	Value	Provided by
1	App ID	4639100322102911	akgpay
2	Secret Key	a26a088d0e854e95	akgpay

Note:

a. Kindly consider whole document as Case-sensitive.
b. App Id C Secret Key will remain constant and can never be changed or modified, in any case.
c. Password can be reset at Merchant's end and other credentials can be modified too.

Step 2 : Encrypt all mandatory JSON Data

```
{
"appId": "APP315796524644", // App_Id provided by AKG
"orderId": "m559455454556", // OrderId unique generated by merchant
"amount": "100",
"custEmail": "user@gmail.com", // Correct Formate
"custPhone": "1234567890",
"currencyCode": "356",
"returnUrl": "https://www.google.com", // Webhook Url user getting real time webhook
"transactionType": "SALE",
"productDesc": "Test Transaction"
}
```

Encrypted Data:

L6L3Zp4+oqICFYVX6q4TmyZLKiEdurwY8XNkOjwoYuKZyyBI4R9H+3Who0Y74/R23YMwosPYQXBpS74tDhS2zTWWMnlI4w5fX+sLDKt0CmfV7RBKkvgq5ifpbfiibJRX6xnZlz1QqjQfh82nTrarg69KbMvVHKMOVHYAKHefsA2FOPZhxsX/w4Q70vvhlIYL7tSKAWQ/1ZhqrMY99Y6IKu6srOWshevRf1bG9XTNG1k+N6pK+7KagSl/8A4e3F9Fjv3tscwtHDvMkRgoGgeg+h+MAibfx1R2aqBDhoLo1VnF1TWKzdknpCmAg2xG8bm0KTcvAqhrJQs4Bn6pkA3B3LAsp9Np3cEkOQ6/VDE+Z3RDMO5Gabzjcv9TSIpUlOYcldqX3EYL2yymBn1WixhsYiOj81Bmf3TqBS8plGG+LcIE3bUxmUMdY8lAQy29D+sPYHlAxC7mmiiDbKLtw1+l3tn/hEjX1YWmoyXtSXfL4GzFHK0qUabVONMm+LL6wEISSKl2cOPYRQW14LUAreF8aJRzCahdiSkgWPxwVb/dQl9bwsGNXhFNhaE25TdWfJZsceFJb0hxomySng1+gCEsYjKQX+HTX98IUF7BZol8Ouo1iQmDiWf7zcMsN+zdOfAa

Encryption And Decryption Logic:

```
import java.security.SecureRandom;
import java.security.spec.KeySpec;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

public class Encryptor {
```

```
private static final int KEY_LENGTH = 256; private static final int ITERATION_COUNT = 65536;

//Encryption Logic starts here
public static String encrypt(String strToEncrypt, String appId, String secretKey) {

try {
            SecureRandom secureRandom = new SecureRandom();
            byte[] iv = new byte[16];
            secureRandom.nextBytes(iv);
            IvParameterSpec ivspec = new IvParameterSpec(iv);
            SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            KeySpec spec = new PBEKeySpec(appId.toCharArray(), secretKey.getBytes(), ITERATION_COUNT, KEY_LENGTH);
            SecretKey tmp = factory.generateSecret(spec);
            SecretKeySpec secretKeySpec = new SecretKeySpec(tmp.getEncoded(), "AES");
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivspec);
            byte[] cipherText = cipher.doFinal(strToEncrypt.getBytes("UTF-8"));
            byte[] encryptedData = new byte[iv.length + cipherText.length];
            System.arraycopy(iv, 0, encryptedData, 0, iv.length);
            System.arraycopy(cipherText, 0, encryptedData, iv.length, cipherText.length);
            return Base64.getEncoder().encodeToString(encryptedData);
  } catch (Exception e) {
     // Handle the exception properly e.printStackTrace();
            return null;
  }
}
//Encryption logic ends



//Decryption logic start here

public static String decrypt(String strToDecrypt, String appId, String secretKey) {

try {
            byte[] encryptedData = Base64.getDecoder().decode(strToDecrypt);
            byte[] iv = new byte[16];
            System.arraycopy(encryptedData, 0, iv, 0, iv.length);
            IvParameterSpec ivspec = new IvParameterSpec(iv);
            SecretKeyFactory factory = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
            KeySpec spec = new PBEKeySpec(appId.toCharArray(), secretKey.getBytes(), ITERATION_COUNT, KEY_LENGTH);
            SecretKey tmp = factory.generateSecret(spec);
            SecretKeySpec secretKeySpec = new SecretKeySpec(tmp.getEncoded(), "AES");
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivspec);
            byte[] cipherText = new byte[encryptedData.length - 16];
            System.arraycopy(encryptedData, 16, cipherText, 0, cipherText.length);
            byte[] decryptedText = cipher.doFinal(cipherText);
            return new String(decryptedText, "UTF-8");
  } catch (Exception e) {
            // Handle the exception properly e.printStackTrace();
    return null;
  }
 }
//Decryption logic ends
}
//class ends
```

Step 3:

```
curl --location 'https://v1.akgpay.com/payment/paymentInit' \

--header 'Content-Type: application/json' \

{
"appId": "APP315796524644",
"data":
"L6L3Zp4+oqICFYVX6q4TmyZLKiEdurwY8XNkOjwoYuKZyyBI4R9H+3Who0Y74/R23YMwosPYQXBpS74tDhS2zTWWMnlI4w
5fX+sLDKt0CmfV7RBKkvgq5ifpbfiibJRX6xnZlz1QqjQfh82nTrarg69KbMvVHKMOVHYAKHefsA2FOPZhxsX/w4Q70vvhlIYL7tSK
AWQ/1ZhqrMY99Y6IKu6srOWshevRf1bG9XTNG1k+N6pK+7KagSl/8A4e3F9Fjv3tscwtHDvMkRgoGgeg+h+MAibfx1R2aqBDho
Lo1VnF1TWKzdknpCmAg2xG8bm0KTcvAqhrJQs4Bn6pkA3B3LAsp9Np3cEkOQ6/VDE+Z3RDMO5Gabzjcv9TSIpUlOYcldqX3EY
L2yymBn1WixhsYiOj81Bmf3TqBS8plGG+LcIE3bUxmUMdY8lAQy29D+sPYHlAxC7mmiiDbKLtw1+l3tn/hEjX1YWmoyXtSXfL4G
zFHK0qUabVONMm+LL6wEISSKl2cOPYRQW14LUAreF8aJRzCahdiSkgWPxwVb/dQl9bwsGNXhFNhaE25TdWfJZsceFJb0hxomy
Sng1+gCEsYjKQX+HTX98IUF7BZol8Ouo1iQmDiWf7zcMsN+zdOfAa"
}
```

Step 4: Checkout Page

```
{
"statusCode": 200,
"error": null,
"data": {
"appId": "APP315796524644",
"paymentLink": "https://checkout.akgpay.com/pay/d8SPx+grkEkGFhxNr4+HNgnJ/TdOGgEp155J4hlVFvw="
}
}
```

Step 5: Web Hook Callback Response on Return URL

```
{
"txnId":  "TXN315796524644",
"orderId":  "m559455454556",
"appId": "APP315796524644",
"amount": "100",
"custEmail": "user@gmail.com",
"custPhone": "9876543210",
"currencyCode": "356",
"paymentTypeCode": "UP",
"mopCode": "INTENT",
"transactionType": "SALE",
"productDesc": "Test Transaction",
"status": "SUCCESS",
"statusCode": "000",
"responseMessage": "SUCCESS Txn",
"utr": "57878578478784",
"cardNumber": "",
"custVpa": "user@okaxis"
}
```

Step 6: Payin Status API

```
{
"orderId": "7837878443823"
}
```

Encrypted orderId

FKUH1+D3020kCgOg9zlQmWQ0u/vEq7a++d5HkMGRR11Q5wdVRTPpPNcU78YSj9JLN80Bv/sF0gPLd67fNKpAbhJHMAeQG7nI5WFQIHDaBgvcd8mUr

```
curl --location 'https://v1.akgpay.com/payment/status' \

--header 'Content-Type: application/json' \
```

--data

```
{
"appId": "APP315796524644",
"data":
"FKUH1+D3020kCgOg9zlQmWQ0u/vEq7a++d5HkMGRR11Q5wdVRTPpPNcU78YSj9JLN80Bv/sF0gPLd67fNKpAbhJHMAeQG7
nI5WFQIHDaBgvcd8mUr"
}
```

Step 7: Status And Status Code

Status	**Status Code**
SUCCESS	000
FAILD	007
PENDING	001
SETNTOBANK	002