



AKGPay
Payment Gateway
Seamless(S2S)

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:

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

Step 1: Encrypt all mandatory JSON Data

UPI Transaction Param:

1. PaymentType= QR and mopCode =QR
2. PaymentType= INTENT and mopCode =INTENT
3. PaymentType= COLLECT and mopCode =COLLECT , custVpa=user Vpa

Card Transaction Param:

1. PaymentType= CC/DC and mopCode =VISA/MASTER etc. ,cardNumber= 444*****6464 , cardExp= 23/2024 , cvv= 123

NetBanking Transaction Param:

1. PaymentType= NB and mopCode =BankCode

```
{
  "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",
  "paymentTypeCode": "UP",
  "mopCode": "INTENT",
  "returnUrl": "https://www.google.com", // Webhook Url user getting real time webhook
  "transactionType": "SALE",
  "productDesc": "Test Transaction"
}
```

```
{
  "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",
  "paymentTypeCode": "UP",
  "mopCode": "COLLECT",
  "custVpa": "user@okaxis",
  "returnUrl": "https://www.google.com", // Webhook Url user getting real time webhook
  "transactionType": "SALE",
  "productDesc": "Test Transaction"
}
```

Encrypted Data:

L6L3Zp4+oqlCFYVX6q4TmyZLKieDurwY8XNkOjwoYuKZyyBI4R9H+3Who0Y74/R23YMwosPYQXBpS74tDhS2zTWWMnlI4w5fX+sLDKt0CmfV7RBKkygq5ifpbfiibJRX6xnZl1QqjQfh82nTrarg69KbMvVHKMOVHYAKHefsA2FOPZhxsX/w4Q70vvhlIYL7tSKAWQ/1ZhqrMY99Y6IKu6srOWshevRf1bG9XTNG1k+N6pK+7KagSl/8A4e3F9Fjv3tscwtHDvMkRgoGgeg+h+MAibfx1R2aqBDhoLo1VnF1TWKzdknpCmAg2xG8bm0KTcvAqhrJQs4Bn6pkA3B3LAsp9Np3cEkOQ6/VDE+Z3RDMO5Gabzjcv9TSIpULOYclDqX3EYL2yymBn1WixhsYiOj81Bmf3TqBS8plGG+LclE3bUxmUMdY8IAQy29D+sPYHlAx7mmiiDbKLtW1+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/directPay ' \
--header 'Content-Type: application/json' \
{
  "appId": "APP315796524644",
  "data":
    "L6L3Zp4+oqICFYVX6q4TmyZLKieDurwY8XNkOjwoYuKZyyBI4R9H+3Who0Y74/R23YMwosPYQXBpS74tDhS2zTWWMnlI4w
    5fX+sLDKt0CmfV7RBKkvgq5ifpbfiibJRX6xnZlZ1QqjQfh82nTrarg69KbMvVHKMOVHYAKHefsA2FOPZhxsX/w4Q70vvhlIYL7tSK
    AWQ/1ZhqrMY99Y6lKu6srOWshevRf1bG9XTNG1k+N6pK+7KagSl/8A4e3F9Fjv3tscwtHDvMkRgoGgeg+h+MAibfx1R2aqBDho
    Lo1VnF1TWKzdknpCmAg2xG8bm0KTcvAqhrJQs4Bn6pkA3B3Lasp9Np3cEkOQ6/VDE+Z3RDM05Gabzjcv9TSIpUJOYcldqX3EY
    L2yymBn1WixhsYiOj81Bmf3TqBS8plGG+LclE3bUxmUMdY8lAQy29D+sPYHlAx7mmiiDbKLtw1+l3tn/hEjX1YWmoyXtSXfL4G
    zFHK0QUabVONMm+LL6wEISSKl2cOPYRQW14LUAreF8aJRzCahdiSkgWPxwVb/dQl9bwsGNXhFNhaE25TdWfjZsceFjb0hxomy
    Sng1+gCEsYjKQX+HTX98IUf7BZol8Ouo1iQmDiWf7zcMsN+zdOfAa"
}
```

Step 4: bank response

```
{
  "statusCode": 200,
  "error": null,
  "data": {
    "txnStatus": "SENTTOBANK",
    "responseCode": "002",
    "responseMessage": "Payment request pending at bank",
    "orderId": "78555575",
    "txnId": "TXN08817270432",
    "intentUrl":
      "upi://pay?pa=yespay.bipsbib@bankltd&pn=BOX+INDIA+PRIVATE+LIMITED&am=1000.0&tn=422314&tr=4
      22314&mcc=4816"
  }
}
```

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/sF0gPLd67fNKpAbhJ
HMAeQG7nI5WFQIHdaBgvc8mUr

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

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

```
--data
```

```
{  
  "appId": "APP315796524644",  
  "data":  
    "FKUH1+D3020kCgOg9zlQmWQ0u/vEq7a++d5HkMGRR11Q5wdVRTPpPNcU78YSj9JLN80Bv/sF0gPLd67fNKpAbhJHMAeQG7  
    nI5WFQIHdaBgvc8mUr"  
}
```

Step 7: Status And Status Code

Status	Status Code
SUCCESS	000
FAILD	007
PENDING	001
SETNTOBANK	002

