AKGPay
Payment Gateway
Payout 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.

IP Whitelisting :

Before integrating the API, ensure that your IP address is whitelisted with us. This is a crucial step to ensure secure and successful communication between your server and our API.

For any further assistance or inquiries, please do not hesitate to contact our support team.

Note – All the API requests should be made over HTTPS instead of HTTP (all calls made over plain HTTP will fail).

Payout Types and Transaction Creation:

This API is used to check the available types of payouts, including NEFT, UPI, IMPS, and RTGS.
When creating a transaction, use the corresponding ID instead of the payout name in the request body as specified below:

1. IMPS
2. NEFT
3. RTGS
4. UPI

Step 2 : Encrypt all mandatory JSON Data

```
{
        "orderId": " 478784747332",
        "name": " Test ",
        "nickname": "test",
        "mobile": "88888******",
        "email": "test@gmail.com",
        "accountNo": "50****86******48",
        "ifscCode": "H***00*****91",
        "transactionAmmount": 1.00,
        "transactionBankTransferMode": "IMPS"

    }
```

Encrypted Data:

L6L3Zp4+oqICFYVX6q4TmyZLKiEdurwY8XNkOjwoYuKZyyBI4R9H+3Who0Y74/R23YMwosPYQXBpS74tDh
S2zTWWMnlI4w5fX+sLDKt0CmfV7RBKkvgq5ifpbfiibJRX6xnZlz1QqjQfh82nTrarg69KbMvVHKMOVHYAKHefsA2FOP
ZhxsX/w4Q70vvhlIYL7tSKAWQ iQmDiWf7zcMsN+zdOfAa

Step 3:

```
curl --location 'https://v1.akgpay.com/payoutTransferApi' \
--header 'Content-Type: application/json' \
{
"appId": "APP315796524644",
"data":
"L6L3Zp4+oqICFYVX6q4TmyZLKiEdurwY8XNkOjwoYuKZyyBI4R9H+3Who0Y74/R23YMwosPYQXBpS74tDhS2zTWWMnlI4w
5fX+sLDKt0CmfV7RBKkvgq5ifpbfiibJRX6xnZlz1QqjQfh82nTrarg69KbMvVHKMOVHYAKHefsA2FOPZhxsX/w4Q70vvhlIYL7tSK
AWQ/1ZhqrMY99Y6IKu6srOWshevRf1bG9XTNG1k "
}
```

Response :

```
{
 "orderId": " 478784747332",
"transactionStatus": "PENDING",
"statusCode": " 001",
"createDate": "2025-01-01"
 }
```

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);
SecretKe      KeySpec secretKeySpec = new SecretKeySpec(tmp.getEncoded(), "AES");  Cipher
y tmp =       cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
factory.g     cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivspec);
enerateS      byte[] cipherText = new byte[encryptedData.length - 16];
ecret(sp      System.arraycopy(encryptedData, 16, cipherText, 0, cipherText.length);
ec);          byte[] decryptedText = cipher.doFinal(cipherText);
S             return new String(decryptedText, "UTF-8");
e
c
r
e
t
      } catch (Exception e) {
              // Handle the exception properly e.printStackTrace();
        return null;
      }
    }
    //Decryption logic ends
    }
    //class ends
```

Step 4: Payin Status API

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

Encrypted orderId

FKUH1+D3020kCgOg9zlQmWQ0u/vEq7a++d5HkMGRR11Q5wdVRTPpPNcU78YSj9JLN80Bv/sF0gPLd67fNKpAbhJHMAeQG7nI5WFQIHDaBgvcd8mUr

```
curl --location 'https://v1.akgpay.com/payoutStatusApi' \
--header 'Content-Type: application/json' \
--data
{
"appId": "APP315796524644",
"data":
"FKUH1+D3020kCgOg9zlQmWQ0u/vEq7a++d5HkMGRR11Q5wdVRTPpPNcU78YSj9JLN80Bv/sF0gPLd67fNKpAbhJHMAeQG7nI5WFQIHDaBgvcd8mUr"
}
```

Step 7: Status And Status Code

Status	Status Code
SUCCESS	000
FAILD	007
PENDING	001
SETNTOBANK	002