OMPAY Payment Gateway Integration Document

Version 1.0.1

2025

المكتب ٤٠١، الطابق الرابع المبنى ٩٤/١، القطعة ٢٥٠، الشارع ٣١٩، غلا، بوشر ، مسقط، صندوق البريد: ١٢٥٧، الرمز البريدي: ١٣٣، سلطنة عُمان.

Office 401, 4th Floor, Building 1/94, Block 250, Street 319, Ghala, Bousher, Muscat, P.O. Box: 1257, Postal Code: 133, Sultanate of Oman.

VAT Certificate Number: 13564020, Tax Card Number: 13599622, CR Number: 1408762

رقم شهادة ضريبة القيمة المضافة: ١٣٥٦٤٠٢٠، رقم البطاقة الضريبية: ١٣٥٩٩٦٢٢، رقم السجل التجاري: ١٤٠٨٧٦٢

Version History:

Version	Publishing Date	Changes
1.0.0	July 2025	Initital Draft
1.0.1	September 2025	Minor changes

Table of Contents

1. Key Notes

Overview

To integrate the OMPAY payment gateway into your website or application, follow the steps outlined in this document. This guide covers obtaining your credentials, creating orders, integrating the standard, checkout form and pay by link generation, handling payment responses, and verifying payment signatures

Obtain Client ID and Secret Key

To authenticate your requests and enable communication with the OMPAY payment gateway, you need to obtain a unique **Client ID** and **Secret Key**.

Steps to Obtain Credentials:

- Log in to the OMPAY merchant portal. Under **Payment Gateway** navigate to **Payment Checkout Page** to create your **Client ID** and **Secret Key.**
- Both the **Client ID** and **Secret Key** must be securely updated in your website backend to authenticate API requests.

Integration Endpoints

- **UAT URL**: https://api.uat.gateway.ompay.com
- **PROD URL**: - https://api.gateway.ompay.com

Note: *The integration endpoint mentioned above refers to the API environment.*

2. Steps to Integrate the Standard Checkout Form.

2.1. Create an order on the server.
2.2. Integrate checkout on client-side.
2.3. Handle Payment Success and Failure.
2.4. Verify payment signature.
2.5. Verify payment status.
2.6. Generate Signature

2.1. Create an Order on the Server

An order should be created for every payment. The orderId received in the response should be passed to the checkout to ensure the request's security against tampering.

How to create an Order:

To create an Order, use the provided API sample code by integrating it into your server. This will enable you to generate an order seamlessly.

API SAMPLE CODE

```
POST: `{{domain}}/nac/api/v1/pg/orders/create-checkout`

curl -X POST {{domain}}/nac/api/v1/pg/orders/create-checkout \
-U [CLIENT_ID]:[CLIENT_SECRET] \
-H 'Content-Type: application/json' \
-d '{
   "amount": 500.00,
   "currency": "INR",
   "uiMode": "checkout",
   "receiptId": "66497326b1577421a0e99fb3",
   "description": "Ecommerce Transaction",
   "customerFields": {
      "email": "xyz@email.com",
      "phone": 9999999999,
      "name": "User name"
   },
   "curn": "curn_id",
   "redirectType": “redirect”
}'
```

Sample Success Response

```
{
  "orderId": "wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27",
  "amount": 500.00,
  "receiptId": "66497326b1577421a0e99fb3",
  "status": "success",
  "resCode": 200,
  "errMessage": ""
}
```

Sample Failure Response

```
{
  "receiptId": "66497326b1577421a0e99fb3",
  "status": "failure",
  "resCode": 400,
  "errMessage": "Order failed"
}
```

Request Parameters

amount *mandatory*	**Description**: The total amount of the transaction in the specified currency. (up to 2 decimals) **Type**: Numeric **Example**: 500.00
currency *mandatory*	**Description:** The currency used for the transactions. **Type:** String **Example:** "INR"
uiMode *mandatory*	**Description:** UI mode for the transaction. **Type:** String **Example:** "checkout"
receiptId *optional*	**Description:** Unique identifier for the transaction. (up to 40 characters) **Type:** String **Example:** "57Kr4ec2qdQMLJ0QVFYZnu1N5y49x0CZz5PlHhp"
description *optional*	**Description:** Additional Information or notes regarding the transaction. **Type:** String **Example:** "Sample Transactions"
curn *optional*	**Description:** Custom identifier or metadata for the order. **Type:** String **Example:** "123456789"

redirectType *mandatory*	**Description:** Redirection mode after the transaction. **Type:** String **Example:** “post” / “redirect”
phone *mandatory*	**Description:** Customer’s phone. **Type:** String **Example:** “9912364456”
email *mandatory*	**Description:** Customer’s email. **Type:** String **Example:** “example@gmail.com”
name *mandatory*	**Description:** Customer’s name. **Type:** String **Example:** “Customer name”

2.2. Integrate Checkout on Client-Side

To integrate the payment checkout process within your website or application, follow the steps below:

HTML Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Add necessary headers -->
</head>
<body>
  <button id="pay-button">Pay</button>

  <script src="{{domain}}/nac/public/checkout.js"></script>
  <script>
    var options = {
      "orderId": " wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27",
      "redirectUrl": "https://example.com/receipt",
      "clientId": "[CLIENT_ID]"
    };
    checkout(options);
  </script>
</body>
</html>
```

Explanation

- **Pay Button:** Replace with an element on your webpage that initiates the payment process.
- **Script Inclusion:** Include the checkout.js script from the provided domain to access necessary payment functionalities.

Request Parameters

orderId *mandatory*	**Description:** Pass the orderId received from the server upon creating the order. **Type:** String **Example:** "wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27"
redirectUrl *mandatory*	**Description:** Specify the URL to which the user is redirected after the payment process completes. **Type:** String **Example:** "https://example.com/receipt"
clientId *mandatory*	**Description:** Pass the ClientID **Type:** String

2.3. Handle Payment Success and Failure

Upon payment success or failure, the customer will be directed to your website page. As part of the onboarding process, please provide us with your webhook URL. Upon payment completion, we will invoke your designated endpoint and transmit the following details:

Success Response

```
{
  "orderId": " wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27",
  "paymentId": "PAY0642d0df582e4d8fa96d796e79b52c99",
  "status": "success",
  "receiptId": "66497326b1577421a0e99fb3",
  "amount": 500.00,
```

```
  "signature": "37bedbdf0381d1df56677487217e4aa02040320f396691da3e783eb0a0591c",
  "timestamp": "2025-04-10T13:42:54.563Z",
  "paymentDetails": {
     "paymentMethod": "card",
     "cardNetwork": "visa",
     "cardType": "credit"
  }
}
```

Failure Response

```
{
  "orderId": " wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27",
  "paymentId": "PAY0642d0df582e4d8fa96d796e79b52c99",
  "status": "failure",
  "receiptId": "66497326b1577421a0e99fb3",
  "amount": 500.00,
  "signature": " b57b027aae98d1d307bea6257fba9ce2b832e8530274c6b458f972084a0305",
  "timestamp": "2025-04-10T13:42:54.563Z",
  "paymentDetails": {
     "paymentMethod": "card",
     "cardNetwork": "visa",
     "cardType": "credit"
  }
}
```

2.4. Verify Payment Signature using JavaScript

To verify the signature, create a signature on your server using the following format:

1. Create a signature on your server.
    a. **orderId**: Retrieve the orderId from your server.
    b. **paymentId**: This field will be available in the success/failure response.
    c. **clientSecret**: You will receive this information during the onboarding process.

2. Use the SHA256 algorithm, as given below.

```
const crypto = require('crypto');

function generateHMAC(key, data) {
   const hmac = crypto.createHmac('sha256', key);
   hmac.update(data);
   return hmac.digest('hex');
}

const secretKey = '<clientSecret>';
```

```
const message = 'orderId|paymentId;
const generatedSignature = generateHMAC(secretKey, message);

if (generatedSignature == signature) {
   // Payment signature is validated. You can now update the transaction status.
}
```

3. If the signature you generate on your server matches the **signature** returned to you, the payment received is from an authentic source.

2.5. Verify Payment Signature using Java

To verify the signature in Java use below format,

1. Create a signature in your server.
    a. **orderId**: Retrieve the orderId from your server.
    b. **paymentId**: This field will be available in the success/failure response.
    c. **clientSecret**: You will receive this information during the onboarding process.

2. Use the SHA256 algorithm, as given below.

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;

public class HMACGenerator {

   public static void main(String[] args) {
      String secretKey = "<client Secret>";
      String message = "<orderId>" + "|" + "<paymentId>";

      String generatedSignature = generateHMAC(secretKey, message);
      System.out.println(generatedSignature);

      // if generatedSignature == signature
      // payment signature is validated.
   }


   public static String generateHMAC(String key, String data) {
```

```
        try {
            Mac hmac = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(), "HmacSHA256");
            hmac.init(secretKeySpec);
            byte[] hmacBytes = hmac.doFinal(data.getBytes());
            return bytesToHex(hmacBytes); // Convert bytes to hexadecimal string
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
            return null;
        }
    }

    // Utility method to convert byte array to hexadecimal string
    public static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    }
}
```

3. If the signature you generate on your server matches the **signature** returned to you, the payment received is from an authentic source.

2.6. Verify Payment Status

To verify the successful completion of a payment transaction and update your system accordingly, use the following method provided by the payment gateway:

The '**orderId**' parameter serves as a unique identifier that links the transaction to be checked for its payment status.

SAMPLE API CALL

```
GET: {{domain}}/nac/api/v1/pg/orders/check-status?orderId={orderId}
```

Sample Request

```
curl -X GET {{domain}}/nac/api/v1/pg/orders/check-status?orderId=wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27 \
```

```
-U [CLIENT_ID]:[CLIENT_SECRET] \
-H 'Content-Type: application/json'
```

Query Parameter: **orderId**: Unique identifier for the specific transaction that you want to check the status for.

EXPLANATION:

1. This parameter is crucial in identifying the transaction for which you want to retrieve the payment status. It is generated and provided by the payment gateway upon creating an order for a transaction.

SAMPLE RESPONSE

```
{
 "orderId": "wb-e883b28d-0cd3-455d-9903-ef5d3d8ddf27",
 "status": "success",
 "paymentId":"PAY0642d0df582e4d8fa96d796e79b52c99",
 "receiptId": "66497326b1577421a0e99fb3",
 "amount": 500.00,
 "signature": "37bedbdf0381d1df56677487217e4aa02040320f396691da3e783eb0a0591c",
 "timestamp": "2025-04-10T13:42:54.563Z",
 "paymentDetails": {
      "paymentMethod": "card",
      "cardNetwork": "visa",
      "cardType": "credit"
   }
}
```

3.Test Card Details:

❖ For Local Debit card

Card Number: 4393570006367857

Expiry Date: 03/28

CVV: 152

❖ 2.For Credit & International card

Card Number: 4012001037490006

Expiry Date: 12/25

CVV: 123