API Specification for Vietnam Cards and Alternate Payment Methods

Version: 2.2

Description

This document introduces the **OpenAPI specification** which describes the REST APIs for HSBC's ASP Omni Collection Vietnam Cards and Alternate Payment Methods.

The target audience of this document are Developers, Business Analysts and other Project Team Members.

Update Log

- [Dec 21, 2021] **v2.2** Revised several content sections
- [Oct 12, 2021] **v2.1** Added new possible value in field `payment_option`
- [Oct 10, 2020] **v2.0** Renew All API design for supporting multiple payment gateways

- ~~[Sep 30, 2020] **v1.9** Added optional request field `description` in Payment Page Redirect API~~
- ~~[Nov 8, 2019] **v1.8** Updated `API Base URL` of both Sandbox and Production~~
- ~~[Sep 20, 2019] **v1.7** Updated `Disclaimer`~~
- ~~[Sep 4, 2019] **v1.6**~~
  - ~~Enhanced Section `API Connectivity`~~
  - ~~Added Content Section `REFERENCE`~~
- ~~[Aug 12, 2019] **v1.5** Updated all API endpoints~~
- ~~[Jul 26, 2019] **v1.4** Modified possible value of response field `respCode` and `status` in Payment Status Enquiry API~~
- ~~[Jul 22, 2019] **v1.3**~~
  - ~~Changed Max Length of field `txnRef` and `rfdRef`~~
  - ~~Removed response object `system` in Order Cancellation API and Refund API~~
- ~~[Jul 19, 2019] **v1.2**~~
  - ~~Changed object name `description` to `descriptions`~~
  - ~~Added Payment Channel `Pay on Delivery` in content section~~
- ~~[Jul 10, 2019] **v1.1**~~
  - ~~Changed field name~~
    `refund_id` ~~to~~ `rfdRef`
    `refundAmt` ~~to~~ `rfdAmount`
    `refundedAmt` ~~to~~ `rfdAmount`
  - ~~Added new field `customer_phone` and `customer_address` in pay_rqt_customer_Obj~~
  - ~~Added new possible value in all `payment_option` fields~~
  - ~~Added new possible value in field `respCode` at cancel_rpn_txn_Obj~~
- ~~[Jun 17, 2019] **v1.0** Initial Version~~

How to Read this Document

This document walks through the API listing the key functions by section: API Usage Flow, API Connectivity, and API Operation. There is also a FAQ and a list of Schema Definitions used by API operations.

This document has links to subsequent sections. For example, when you visit the section API Operation, it has links to the data model or schemas containing the data and status codes definitions.

Use Cases for this API

HSBC Omni Collect provides a wide range of payment solutions which allow online merchants to process different online and offline payments through the online secured payment gateway. The payment gateway supports implementations with websites or mobile applications.

Our solution also offer choices between different Payment Gateway Partners depending on the merchant's business needs. Please contact our team to learn more. To present any proprietary terminology or service provided by one specific Payment Gateway Partner, the content will be highlighted in a coloured `Block Quote` as in the example below:

> Gateway 1
> **INFORMATION:**
> Indicate the corresponding service or item is eligible for Payment Gateway #1

> Gateway 2
> **INFORMATION:**
> Indicate the corresponding service or item is eligible for Payment Gateway #2

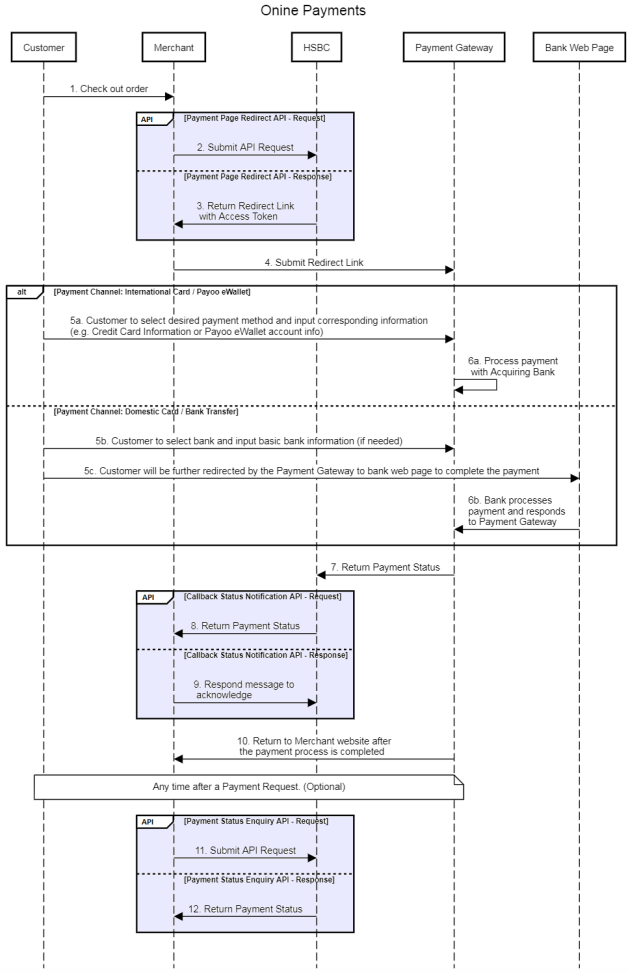
Using our API services, you can allow your eCommerce / mCommerce website or Mobile Application to accept payments including the following payment channels:

Payment Channel	API Model
International Credit / Debit Card Payments	Online Payment
Vietnamese Domestic Cards / Bank account	Online Payment plus Redirect to Bank Website
Payoo eWallet	Online Payment
QR Code Payment	QR Code Payment or Display QR Code on Merchant Website
Pay at Convenience Store	Offline Payment
Pay on Delivery	Offline Payment

Online Payments

Online Payments include Credit / Debit Card Payment, Online Bank Transfer and eWallent Payment.

API Use Case



1. The Customer conducts a checkout process on merchant's website.
2. The Merchant submits a Payment Page Redirect API request to HSBC.
3. HSBC returns a JSON response which embeds the redirect link of the Online Payment Gateway with an access token inside field `redirectLink`.
4. The Merchant submits the redirect link. It redirects the Merchant website to the Online Payment Gateway.

> **NOTE:**
> The Merchant can restrict the display to one particular payment option in the payment gateway by passing corresponding value in field `payment_option` during API request.

5. **For International Card or Payoo eWallet:** the Customer selects a desired payment method and inputs corresponding information inside the Payment Gateway. *(e.g. Credit Card information or Payoo eWallet account info)*
   For Domestic Card or Bank Transfer: the Customer selects the Domestic Card option and chooses a bank from the list of available banks. Depending upon the selected option, the Customer may need to supply more information such as Bank Card Number, Bank Account Number, or Holder Name inside the Payment Gateway. In order to complete the payment, the customer's browser is redirected to the Bank Website for more input.
6. The Payment is processed between the issuing and acquiring bank systems. The Payment Gateway collects the final payment result after the payment process is completed.
7. HSBC receives payment status as soon as it's updated from the backend system.
8. HSBC triggers a Callback Payment Notification API and send payment status back to the Merchant.

> **NOTE:**
> The Merchant can define the URL to catch the Notification in request field `notificationUrl` in Payment Page Redirect API.

9. The Merchant responds to the API with an acknowledge. Failure to return a proper response triggers the Notification resend mechanism.
10. The browser redirects back to the merchant website when the customer presses "Return to shopping website".

> **NOTE:**
> In the Payment Page Redirect API, the Merchant can define the redirect back URL using the request field `redirectUrl`.

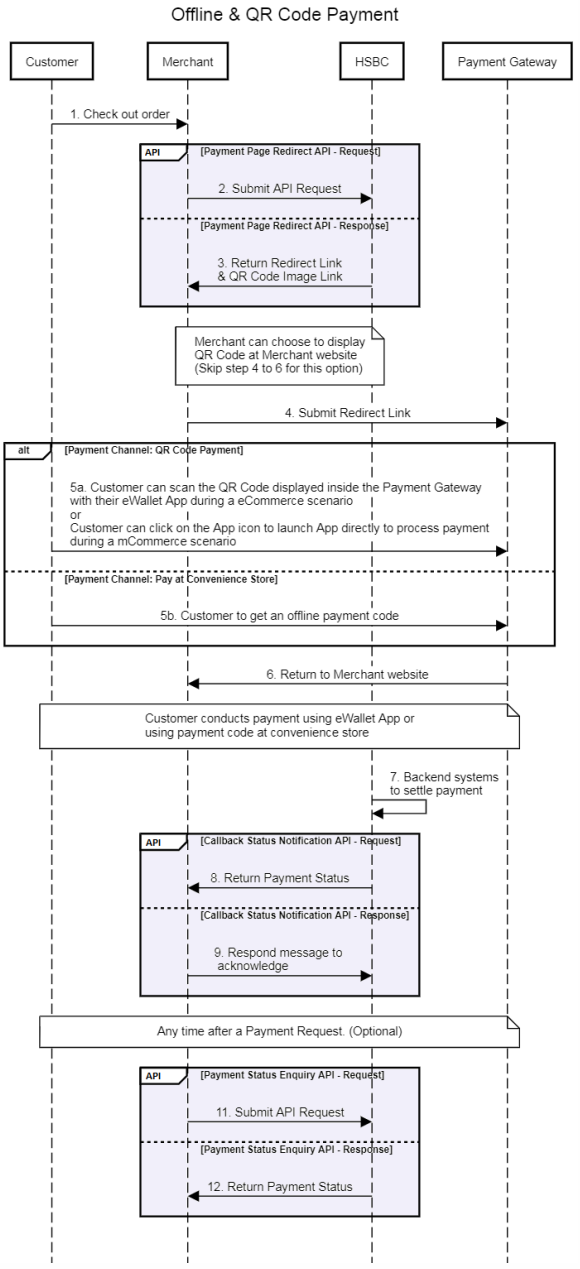
11. The Merchant can optionally submit a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant receives no acknowledge message returned after a certain period of time.
12. HSBC will return the latest payment status according to the transaction reference number the Merchant provided.

Offline Payments & QR Code Payment

Offline Payments are pending between request and payment. The Customer is allowed to make payments via cash, cards, QR code, bank transfers, or any other means at a convenience store or using pay on delivery.

A **QR code payment** is a coantactless payment method whereby a payment is performed by scanning a QR code from a mobile app. This is an alternative to doing electronic funds transfer at point of sale using a payment terminal.

API Use Case



1. The Customer conducts a checkout process in merchant's website.
2. The Merchant submits a Payment Page Redirect API request to HSBC.
3. The redirect link of the Payment Gateway is returned inside the API response field `redirectLink`.

> **NOTE:**
> For a QR Code Payment scenario, the Merchant can choose to display the QR Code image directly on their website rather than on the Payment Gateway - a link to the payment QR Code image is returned in the API response field `qrCodeLink`.

4. The Merchant submits the redirect link. It redirects the Merchant website to the Online Payment Gateway.
5. **For QR Code Payment:** during a eCommerce scenario, the Customer scans the QR Code displayed inside the Payment Gateway with their eWallet App. Alternatively, the Customer clicks the App icon to launch the App on their mobile device to directly process the payment.
   For Pay at Convenience Store: the Customer gets an offline payment code that he/she can present and pay at a convenience store or bank counter at a later time.
   For Pay on Delivery: the Customer clicks the option to confirm and then waits until shipper delivers the ordered goods before making the payment.
6. The browser redirects back to the merchant website once the customer presses "Return to shopping website".

> **NOTE:**
> The Merchant can define the redirect back URL in the request field `redirectUrl` in the Payment Page Redirect API.

7. The HSBC backend system receives the payment status as soon as the payment process is completed at the acquiring bank.
8. HSBC triggers a Status Notification API and send payment status back to the Merchant.

> **NOTE:**
> The Merchant can define the redirect back URL in the request field `notificationUrl` in the Payment Page Redirect API.

9. The Merchant responds to the API to acknowledge receipt. Failure to return a correct response triggers a Notification resend mechanism.
10. Optionally, the Merchant can submit a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant finds no acknowledge message is returned after a certain period of time.
11. HSBC will return the latest payment status according to the transaction reference number that the Merchant provided.

Check Status Feature

Omni Collect provides a feature for the merchant to check the status of every payment transaction. To implement the Check Status feature, please see the Payment Enquiry API.

Cancel & Refund

The Merchant can request an Order Cancellation & Refund API to either cancel an existing order whose payment transaction is yet to be settled, or refund a settled transaction (Settled on both issuing and acquiring bank).

HSBC accepts Full Refund and multiple Partial Refund. Every refund is a new transaction and is returned in an array object in the Payment Enquiry API response message.

Order Confirmation

Regarding the previous API use case flow, the final step is to redirect the Payment Page back to the Merchant website. The Merchant can build a dynamic Order Confirmation Page with payment details retrieved from the asynchronous Callback Payment Notification API.

How to Connect

API Connectivity refers to all measures and their components that establishes connection between HSBC, the API Provider and Merchant, the API Consumer.

	Definition	Components
API Authentication	HTTP BASIC Authentication	• Username • Password
	Locate API Gateway Policy of the corresponding user	• Client ID • Client Secret
User Identification	A Merchant Profile	• Merchant ID • Merchant Profile
Connection Security	HTTPS Connection (TLS 1.2) and Network Whitelisting	• SSL Certificate • Network Whitelist
Message Security	Digital Signing and Data Encryption	• A pair of Private Key & Public Key Certificate (PKI Model) • JWS Key ID • JWE Key ID

API Gateway URL

You need to include this before each API endpoint to make API calls.

Production
https://cmb-api.hsbc.com.hk/glcm-mobilecoll-mcvn-ea-merchantservices-prod-proxy/v1

Sandbox
https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcvn-ea-merchantservices-cert-proxy/v1

API Authentication

Username & Password		
Purpose	All APIs are authorized using `Basic Authorization`	
Components	• Username	• Password
Where to get it?	Delivered by HSBC via secure email during onboarding procedure	
Implementation	In HTTP header: `Authorization: Basic [Base64-encoded Credential]`	

Client ID & Client Secret		
Purpose	API Gateway locates the corresponding policy of the specific API consumer	
Components	• Client ID	• Client Secret
Where to get it?	Delivered by HSBC via secure email during onboarding procedure	
Implementation	In HTTP header: `x-hsbc-client-id: [Client ID]`	In HTTP header: `x-hsbc-client-secret: [Client Secret]`

User Identification

Merchant Profile & Merchant ID		
Purpose	• Merchant Profile contains all necessary information from a Merchant in order to enable payment service.	• Merchant ID is used for Merchant identification in each API call.
Components	• Merchant Profile	• Merchant ID
Where to get it?	• Set up by HSBC team after collect information from Merchant	• Delivered by HSBC via secure email during onboarding procedure
Implementation	*nil*	In HTTP header: `x-hsbc-msg-encrypt-id: [Merchant ID]+[JWS ID]+[JWE ID]`

Connection Security

SSL Certificate & Network Whitelist			
Purpose	• Request HSBC API over HTTPS connection (TLS 1.2)	• Accept Callback API request over HTTPS connection (TLS 1.2)	
Components	• Public SSL Certificate issued by HSBC	• Merchant's web server or domain whose HTTPS connection is enabled	• Network Whitelist on HSBC system
Where to get it?	• Downloaded automatically by Browsers or API Tools, if any problem found, please contact HSBC	*nil*	*nil*

SSL Certificate & Network Whitelist			
Implementation	*nil*	*nil*	• Merchant's domain URL will be configured in HSBC's network whitelist by HSBC team

Message Security - Data Encryption and Signing

In addition to the Transport Layer Security, HSBC adopts additional security - Data Encryption on the message being passed across the session. This serves as a type of locked briefcase containing the data (the API message) within the HTTPS "tunnel". In other words, the communication has double protection.

> **DID YOU KNOW?**
> Javascript Object Signing and Encryption (**JOSE™**), is a framework that secures information transferred between parties. To achieve this, the JOSE framework provides a collection of specifications, including JSON Web Signature (**JWS™**) and JSON Web Encryption (**JWE™**).

HSBC uses JWS to sign message payloads, and JWE to encrypt the signed message. These are created by using the Private Key & Public Key Certificate (PKI Model).

Private Key & Public Key Certificate (PKI Model)		
Purpose	• Digitally sign a API request message • Decrypt a API response message	• Encrypt the signed API request message • Verify a signed API response message
Components	• Private Key issued by Merchant	• Public Key Certificate issued by HSBC
Where to get it?	• Created by any Public Key Infrastructure (PKI) toolkits, such as Keytool™ and OpenSSL™. Technical detail is in here	• Exchanged with HSBC with the Public Key Certificate issued by Merchant
Implementation	Please see the technical detail in here	

> **NOTE:**
> Technically, an X.509 certificate can serve as a SSL Certificate as well as a Public Key Certificate for Data Encryption. However, for segregation of certificate usage, HSBC recommends that the Merchant uses a different X.509 Certificate for Data Encryption. Moreover, the Public Key Certificate does not have to be CA-signed. However, if the Merchant decides to enhance security, a CA-Signed Certificate is acceptable.

keyID of JWS™ & JWE™		
Purpose	• The unique identifier to bind Merchant's Private Key in order to create a JWS object - a signed Message Payload	• The unique identifier to bind HSBC's Public Key Certificate in order to create a JWE object - an encrypted JWS object
Components	• keyID of JWS™	• keyID of JWE™
Where to get it?	• Mutual agreed between Merchant and HSBC	• Mutual agreed between Merchant and HSBC
Implementation	Define in program coding, see demo in here	

> **NOTE:**
> For security purposes, `HSBC's Public Key Certificate` and its associated `keyID` is renewed **every** year and a Certificate Renewal process is triggered. More detail is covered in the section Key Renewal

How to Sign and Encrypt Outgoing Message

Every message sent to HSBC must be signed and encrypted. From the Merchant's perspective, an **Outgoing Message** means:

- the Request Message of a Service API, or
- the Respond Message of a Callback API.

To help you understand how to construct a Signed and Encrypted Message, let's take the Java program below as an example. Don't worry if you are not familiar with Java, the idea is to let you know the steps and the required components:

> **NOTE:** These Java codes are for demonstration only - it's not *plug and play*.

```
private JWSObject signMessage(String messagePayload, KeyStore ks, String keyAlias, String keyPw)
  throws UnrecoverableKeyException, KeyStoreException, NoSuchAlgorithmException, JOSEException {
#1  Payload payload = new Payload(messagePayload);

#2  JWSHeader header = new JWSHeader
              .Builder(JWSAlgorithm.RS256)
              .keyID("0001")
              .customParam("iat", Instant.now().getEpochSecond()).build();
#3  JWSObject jwsObject = new JWSObject(header, payload);

#4  PrivateKey privateKey = (PrivateKey) ks.getKey(keyAlias, keyPw.toCharArray());
    JWSSigner signer = new RSASSASigner(privateKey);
#5  jwsObject.sign(signer);

    return jwsObject;
}
```

1. Prepare your **Message Payload**, that is, the plain `json` request message.
2. Create a **JWS Header** where the parameters are as follows:

```
{
  "alg": "RS256",      //Signing Algorithm is RS256
  "kid": "0001",       //Put your own Key ID value, "0001" is just an example
  "iat": "1625587913"  //Issued At - the time this request is sent, in Unix Time format
}
```

3. Create a **JWS Object** by combining JWS Header and Message Payload.
4. Retrieve your **Private Key** as the signer.
5. Create a **Signed JWS Object** by signing it with the Private Key.

Next, **Encrypt** the Signed JWS Object:

```
private JWEObject getEncryptedJWEObject(JWSObject jwsObject, RSAPublicKey key)
  throws JOSEException {
#1  Payload jwepayload = new Payload(jwsObject.serialize());

#2  JWEHeader jweheader = new JWEHeader.Builder(JWEAlgorithm.RSA_OAEP_256, EncryptionMethod.A128GCM).
#3  JWEObject jweObject = new JWEObject(jweheader, jwepayload);

#4  JWEEncrypter encrypter = new RSAEncrypter(key);
#5  jweObject.encrypt(encrypter);

    return jweObject;
}
```

1. Prepare your **JWE Payload**, that is, the `Signed JWS Object`.
2. Create the **JWE Header**. The algorithm used to encrypt the message body is `A128GCM` while the algorithm used to encrypt the encryption key is `RSA_OAEP_256`. **JWE keyID** is `0002`.
3. Create the **JWE Object** by combining JWE Header and JWE Payload.
4. Retrieve the **HSBC's Public Key** as the encrypter.

5. Create the **Encrypted JWE Object** by encrypting it with HSBC's Public Key.

You are now ready to put the Encrypted JWE Object in the message body *(you may need to first serialize it into String format, depends on your program code design)* of any API call.

How to Decrypt Message and Verify Signature of an Incoming Message

Every message sent from HSBC must be decrypted and verified. From the Merchant's perspective, an **Incoming Message** means:

- the Respond Message of a Service API, or
- the Request Message of a Callback API.

Let's look into the following example to see how to decrypt a response message from HSBC:

```
private String decryptMessage(String respMsgPayload, KeyStoreFactory keyStore)
  throws KeyStoreException, NoSuchAlgorithmException, CertificateException, IOException,
         java.text.ParseException, UnrecoverableKeyException, JOSEException {
#1  JWEObject jweObject = JWEObject.parse(respMsgPayload);

#2  PrivateKey privateKey = (PrivateKey) keyStore.getPrivateKey("merchant_private_key_alias");

    JWEDecrypter decrypter = new RSADecrypter(privateKey);
#3  jweObject.decrypt(decrypter);

#4  String signedMessage = jweObject.getPayload().toString();
    return signedMessage;
}
```

1. Create an **Encrypted JWE Object** by parsing the encrypted response message payload.
2. Retrieve the **Private Key** as the decrypter.
3. Decrypt the JWE Object using your Private Key.
4. Get the **Signed Message** from the decrypted JWE Object.

You are now able to extract the plain `json` message, but first you **must** verify the signature to guarantee data integrity.

```
private String verifySignature(String signedMessage, KeyStore ks, String keyAlias)
  throws KeyStoreException, JOSEException, ParseException {
#1  JWSObject jwsObject = JWSObject.parse(signedMessage);

    Certificate certificate = ks.getCertificate(keyAlias);
#2  JWSVerifier verifier = new RSASSAVerifier((RSAPublicKey) certificate.getPublicKey());

#3  if (!jwsObject.verify(verifier)) {
      throw new ValidationException("Invalid Signature");
    }
#4  return jwsObject.getPayload().toString();
}
```

1. Create a **JWS Object** by parsing the `Signed Message`.
2. Retrieve the **HSBC's Public Key** as the verifier.
3. Verify the signed JWS Object. Invoke error handling if an invalid signature is found *(depends on your code design)*.
4. Get the plain `json` message for further actions.

Summary

Components \ Steps	Message Signing	Message Encryption	Message Decryption	Verify Signature
JWS Object	Signing Algorithm: `RS256`			
JWE Object		JWE Algorithm: `RSA_OAEP_256` Encryption Method: `A128GCM`		
KeyID	`0002`	`0002`		
Merchant's Private Key	Used as `Signer`		Used as `Decrypter`	
HSBC's Public Key		Used as `Encrypter`		Used as `Verifier`

How to Make an API Request

An API request can be submitted without Message Encryption, in case you want to:

- learn about the basic API Call;
- test API connectivity before spending substantial development effort on Message Encryption.

Data encryption is a required data security imposed by HSBC standards. The Merchant has to invoke the encryption logic before moving to Production and must be fully tested during the testing phase.

Make Your API Request with Plain Messages

> ! **NOTE:**
> In the Sandbox Environment you can skip message encryption. However, this is for testing purpose only.

Submit an example API request using cURL™
cURL™ is a simple command-line tool that enables you to make any HTTP request. Merchant can choose any other GUI tool such as Postman™ and SoapUI™.

Step 1. Run this command on your platform:

POST GET

```
#1 curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcvn-ea-merchant
#2  -H "message_encrypt: false"
#3  -H "Authorization: Basic eW9icl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#4  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#5  -H "x-HSBC-client-secret: 1bb456a541dc416d8601B5F9583C606"
#6  -H "x-HSBC-msg-encrypt-id: 4229854900001+0001+0002"
#7  -H "Content-Type: application/json"
#8  -d "{ \"txnRef\": \"PAY-QJZV956664\", \"merId\": \"4229854900001\"}"
```

1. Submit the `POST` request to the API URL endpoint.
2. Set the secret header `message_encrypt: false` to indicate this API request is without message encryption. This header is only applicable in Sandbox environment.
3. Put the Basic Authorization in HTTP header `Authorization`.
4. Put the Client ID in HTTP header `x-HSBC-client-id`.
5. Put the Client Secret in HTTP header `x-HSBC-client-secret`.
6. Put the Merchant ID, the JWS ID and the JWE ID in HTTP header `x-HSBC-msg-encrypt-id` respectively.
7. Set the `Content-Type` to JSON format.
8. Plain `json` message payload.

Step 2. Receive the response message in plain `json` format.

Making API Request with Message Encryption

Step 1. Run this cURL™ command on your platform:

POST GET

```
#1 curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcvn-ea-merchant
#2  -H "Authorization: Basic eW9icl9ic2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#3  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#4  -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
#5  -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
#6  -H "Content-Type: application/json"
#7  -d "eyJraWQiOiIwMDAxIiwiZW5jIjoiQTEyOEdDTSIsImFsZyI6IlJTQS1PQUVQLTI1NiJ9.W4nobHoVXUMOXGM5I-wG
```

1. Submit the `POST` request to the API URL endpoint. Any `{id}` adhered in the URL must be encrypted.
2. Put the Basic Authorization in HTTP header `Authorization`.
3. Put the Client ID in HTTP header `x-HSBC-client-id`.
4. Put the Client Secret in HTTP header `x-HSBC-client-secret`.
5. Put the Merchant ID, the JWS ID and the JWE ID in HTTP header `x-HSBC-msg-encrypt-id` respectively.
6. Set the `Content-Type` to JSON format.
7. The Encrypted Message Payload.

> **NOTE:**
> Data Encryption invokes compulsory prerequisites, such as JOSE library and program coding, please make sure the section on Message Security has been gone through thoroughly.

Step 2. For a successful request (HTTP Status Code 200), an encrypted response message is returned, otherwise, a plain `json` with failure message is returned.

Data Type Overview

Data Type Control:

Data Type	Allowed Characters	Definition & Important Notice
String *(For general field)*	AlphaNumeric and Symbols	General field means field which is **NOT** a critical field. HSBC system will execute characters checking upon all string fields we received in order to tackle security vulnerability, such as Cross-site Scripting. Yet, we recommend you to try use AlphaNumeric only for most cases.

String *(For critical field)*	`0-9` `a-z` `A-Z` `-` `_` `.`	Critical field is used to be either a key or search criteria in HSBC backend system and hence tight restriction is applied to the allowed characters.
		
		Moreover, the starting and ending space of the string value will be trimmed before stored in HSBC system. For example, string `" example 12 34 "` will be trimmed to `"example 12 34"`.
		
		List of Critical Fields:
		`txnRef`
		`product_id`

Integer	`0-9`	Instead of having Max Length check for String, integer range will be checked, e.g. `0 ≤ x ≤ 9999`

Field Mandatory Control:

Field Mandatory Type	Definition & Important Notice
Mandatory	Annotated with required tag in field definition section. Field & value must be present in the request with valid `JSON` format.
Optional	Annotated with optional tag in field definition section. If you don't want to pass fields that are optional, your handler should not pass neither empty strings `{"example":""}` nor blank value `{"example":" "}`.
Conditional	Annotated with conditional tag in field definition section. Required under a specific condition whose logic is always provided in the field definition if it is a Conditional Field.

Time Zone Control:

Aspect	Format	Definition & Important Notice
In Request Message	`yyyy-MM-dd'T'HH:mm:ssZ`	Time zone is expected to be `GMT+7` (Vietnam local time). Merchant is required to perform any necessary time zone conversion before submit request if needed.
In Response Message	`yyyy-MM-dd'T'HH:mm:ss±hh:mm`	Timezone returned in `api_gw` object is generated from HSBC API Gateway which located in Cloud and hence is calculated in `GMT+0`. On the other hand, time field in `response` object will be returned together with timezone information. For more details, please read each field definition carefully.

FAQ

SSL Connection Questions

Where can I find the HSBC SSL server certificates?

The Merchant developer can export SSL server certificates installed in your browser. To achieve this, visit the domain of the corresponding API endpoint in your browser. For example, to get the SSL certificate of sandbox environment, use the domain name **https://devcluster.api.p2g.netd2.HSBC.com.hk/**

However, in production, we provide a certificate and require TLS 1.2 implementation.

Message Encryption Questions

What certificates do I need to work with Message Encryption in HSBC's sandbox and production environments?

A self-sign certificate is acceptable. However, if the Merchant decides to enhance security, a CA-Signed Certificate is also acceptable.

Javascript Object Signing and Encryption (JOSE) Framework Questions

Where can I get more information about JOSE Framework?

If you want to fully understand the framework, you can read here for more details.

Please note these urls or websites do not belong to HSBC, use them at your own discretion. By clicking these urls or websites signifies you accept these terms and conditions.

Where can I download JOSE libraries for development?

For your reference, you may find the following JOSE libraries of different programming languages.

- Ruby
- Python
- PHP
- Java
- Node
- .NET

Please note these urls or websites do not belong to HSBC, use them at your own discretion. By clicking these urls or websites signifies you accept these terms and conditions.

Payments

Contains resource collections for conventional payments, enquiry, notification, etc.

Payments

Create a Payment Link

`POST` `/payment/pageRedirect`

DESCRIPTION

This API returns a URL link that redirects Merchant's browser to the Secured Online Payment Page. Customer can input all other necessary information (such as Credit Card details) in that page to complete the payment.

A payment QR Code is also available and returned for a particular payment gateway.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json

REQUEST BODY

paymentReqtModel — *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: application/json

Request Example

```
{
  "transaction": {
    "txnRef": "ORD-438UL748T6"
  },
  "system": {
    "redirectUrl": "https://www.example.com/redirectBacktoMerchantSite",
    "notificationUrl": "https://www.example.com/receiveNotification"
  },
  "payment": {
    "country": "VN",
    "currency": "VND",
    "payment_option": "all",
    "amount": 10200000,
    "expiry": "2020-05-31T14:10:25Z"
  },
  "customer": {
    "name": "Bùi Khánh An",
    "email": "customer.name@example.com",
    "phone": "091-1111111",
    "address": "xxxxxxx"
  },
  "order": {
    "shippingDate": "2019-01-01",
    "shippingDays": 5,
    "description": "Proceed check out for your order #ORD-438UL748T6",
    "descriptions": [
      {
        "product_name": "Product Item 1",
        "product_id": "PRO-ASDF-1234",
        "unitAmt": 1500000,
        "unit": 2,
        "subAmt": 3000000
      },
      {
        "product_name": "Product Item 2",
        "product_id": "PRO-JHGF-9876",
        "unitAmt": 2400000,
        "unit": 3,
        "subAmt": 7200000
      }
    ]
  },
  "other": {
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```

RESPONSES

200 OK paymentRespModel	Successful operation. *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*
400 Bad Request commonRespObj	Missing or invalid Parameters.
403 Forbidden	Authorization credentials are missing or invalid.
404 Not Found	Empty resource/resource not found.
500 Internal Server Error	The request failed due to an internal error.

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "ORD-438UL748T6"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "sysDatetime": "2019-01-05T15:20:45+07:00",
      "qr_code": "https://qr-gw-sb.payoo.vn:8712/QRLink_GatewayWCFService/REST/GetQRCode2?
QRCodeKey=xxxx",
      "redirectLink": "https://newsandbox.payoo.com.vn/v2/paynow/detail?_token=xxxx"
    }
  }
}
```

Response Example (400 Bad Request)

```
{
  "messageId": "89817674-daOO-4883",
  "returnCode": "400",
  "returnReason": "Error Message Here",
  "sentTime": "2016-11-15T10:00:00.000Z",
  "responseTime": "2016-11-15T10:00:00.000Z"
}
```

Payments

Get Transaction Information

`GET` `/payment/transaction/{txnRef}`

DESCRIPTION

Merchant can optionally initiate payment status enquiry at any time after a payment request is submitted. This is used when Merchant wants to check payment status any time after a payment request or find no acknowledge message returned after a certain period of time. HSBC Mobile Collection will return the latest transaction status according to the transaction reference number Merchant provides.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
txnRef: string required in path	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

RESPONSES

200 OK enquiryRespModel	Successful operation. *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*
400 Bad Request commonRespObj	Missing or invalid Parameters.
403 Forbidden	Authorization credentials are missing or invalid.
404 Not Found	Empty resource/resource not found.
500 Internal Server Error	The request failed due to an internal error.

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "transaction": {
      "txnRef": "ORD-438UL748T6",
      "error_code": "53"
    },
    "payment": {
      "status": "4",
      "payment_option": "1",
      "amount": 500000,
      "payment_datetime": "2019-12-12T14:10:25+07:00",
      "bank_name": "VISA",
      "mcn": "411111******1111",
      "payment_code": "11815866"
    },
    "refunds": [
      {
        "id": "CT201801302",
        "status": "0",
        "amount": 5000,
        "datetime": "2018-12-12T14:10:25+07:00"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


```
{
  "messageId": "89817674-daOO-4883",
  "returnCode": "400",
  "returnReason": "Error Message Here",
  "sentTime": "2016-11-15T10:00:00.000Z",
  "responseTime": "2016-11-15T10:00:00.000Z"
}
```


Payments

Cancel or Refund an Order

`POST` `/payment/refund`

DESCRIPTION

This API can either cancel an unsettled order or send a refund request for a settled transaction. It supports both full and partial refund.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json

REQUEST BODY

refundReqtModel	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

RESPONSES

200 OK refundRespModel	Successful operation. *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*
400 Bad Request commonRespObj	Missing or invalid Parameters.
403 Forbidden	Authorization credentials are missing or invalid.
404 Not Found	Empty resource/resource not found.
500 Internal Server Error	The request failed due to an internal error.

Request Content-Types: application/json

Request Example


```
{
  "txnRef": "ORD-438UL748T6",
  "rfdRef": "RFD-438UL748T6",
  "reason": "Item is damaged",
  "amount": 550000
}
```


Response Content-Types: application/json

Response Example (200 OK)


```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "transaction": {
      "txnRef": "ORD-438UL748T6",
      "amount": 550000
    },
    "refund": {
```

```
    "id": "CT201801302",
    "status": "0",
    "amount": 550000
  }
 }
}
```

Response Example (400 Bad Request)

```
{
  "messageId": "89817674-daOO-4883",
  "returnCode": "400",
  "returnReason": "Error Message Here",
  "sentTime": "2016-11-15T10:00:00.000Z",
  "responseTime": "2016-11-15T10:00:00.000Z"
}
```

Payments

Payment Status Notification

POST `/<Callback URL predefined by Merchant>`

DESCRIPTION

Payment status will be returned to Merchant by asynchronous callback once Mobile Collection receives a payment request. After Mobile Collection payment platform completes reconciliation with bank and receives payment result, Mobile Collection will push the result back to Merchant by calling this API.

> **Implementation**
> This is a Callback API. HSBC will trigger this API call and defines the interface with OpenAPI standard. Merchant is required to provide implementation.

> **Retry Mechanism**
> If no success response is received, up to 4 retries will be triggered in every 2 minutes. Maximum 5 calls including the 1st attempt.

> **Endpoint Definition**
> Field `notificationUrl` from Payment Page Redirect API will be used as URL endpoint of the corresponding transaction.

> **Exception Handling**
> Only success case will be returned. Merchant can submit a Payment Status Enquiry API request if found no acknowledge message returned after a certain period of time.

REQUEST PARAMETERS

Content-Type: string required in header — text/plain

REQUEST BODY

statusRtnReqtModel — *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: text/plain

Request Example

```
{
  "transaction": {
    "txnRef": "ORD-438UL748T6"
  },
  "merchant": {
    "merId": "723"
  },
  "payment": {
    "status": "PAYMENT_RECEIVED",
    "amount": 500000,
    "payment_option": "INTERNATIONAL_CARD",
    "payment_code": "11815866"
  },
  "other": {
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```

RESPONSES

200 OK — Successful operation.

statusRtnRespModel — *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "status": "SUCCESS"
}
```

Schema Definitions

commonRespObj: object

PROPERTIES

messageId: string range: (up to 36 chars) required
System generated unique message ID only for HSBC internal reference use

returnCode: string range: (up to 3 chars) required
System Return Code.

- This checking is on API Operational level, in other words, it checks upon Authorization, Connectivity and JSON Message Structure.

Possible Value	Definition
200	Successful operation
400	Bad Request (With detail message in field `returnReason`)
500	Internal Error. **Important Notices:** If any tier comes before the API Cloud Foundry is unavailable, such as the API Gateway, there will be no json respond message returned. Furthermore, the respond message of 500 will be ignored by some common HTTP libraries, in such case, the respond message body can be considered as a hint for troubleshooting during development and testing phase.

returnReason: string range: (up to 200 chars) required
Corresponding Text message of returnCode

Corr. Return Code	Return Message Sample	Definition

Example

```
{
  "messageId": "89817674-daOO-4883",
  "returnCode": "200",
  "returnReason": "Successful operation",
  "sentTime": "2016-11-15T10:00:00.000Z",
  "responseTime": "2016-11-15T10:00:00.000Z"
}
```

Corr. Return Code	Return Message Sample	Definition
200	Successful operation	A successful API operation in terms of Authorization, Connectivity and valid JSON Message Structure. Any checking failure on Business Logic level will be still considered a successful API operation yet the Business Logic checking result will be returned in `response` object.
400	Client ID - Merchant ID mapping is not correct/updated!	The binding of Client ID, Merchant ID and Merchant Public Certificate is incorrect or not up-to-date.
400	object has missing required properties `field name`	Fail to pass JSON Field Mandatory Check.
400	instance type `data type` does not match any allowed primitive type	Fail to pass JSON Field Type Check.
400	string `field value` is too long	Fail to pass JSON Field Max Length Check
400	instance failed to match at least one required schema among `no. of conditional field`	Fail to pass JSON Conditional Field Check.
500	java.net.ConnectException: Connection refused: connect	**Notices:** Message can be varied depended on the dependent system *(which across the entire system pipeline)* which returns this message. Yet, all reasons can be concluded into Internal Error or System Unavailable.

sentTime: string range: (up to 27 chars) required
Time of request received by HSBC system from client, only for HSBC internal reference use

responseTime: string range: (up to 27 chars) required
Time of HSBC system provides response to client, only for HSBC internal reference use

paymentReqtModel: object

PROPERTIES

transaction: pay_rqt_txn_Obj required
system: pay_rqt_system_Obj required
payment: pay_rqt_payment_Obj required
customer: pay_rqt_customer_Obj optional

Gateway 1
NOTICE:
Object `customer` is only available for Payment Gateway #1

order: pay_rqt_order_Obj required
other: pay_rqt_other_Obj optional

Example


```
{
  "transaction": {
    "txnRef": "ORD-438UL748T6"
  },
  "system": {
    "redirectUrl": "https://www.example.com/redirectBacktoMerchantSite",
    "notificationUrl": "https://www.example.com/receiveNotification"
  },
  "payment": {
    "country": "VN",
    "currency": "VND",
    "payment_option": "all",
    "amount": 10200000,
    "expiry": "2020-05-31T14:10:25Z"
  },
  "customer": {
    "name": "Bùi Khánh An",
    "email": "customer.name@example.com",
    "phone": "091-1111111",
    "address": "xxxxxxx"
  },
  "order": {
    "shippingDate": "2019-01-01",
    "shippingDays": 5,
    "description": "Proceed check out for your order #ORD-438UL748T6",
    "descriptions": [
      {
        "product_name": "Product Item 1",
        "product_id": "PRO-ASDF-1234",
        "unitAmt": 1500000,
        "unit": 2,
        "subAmt": 3000000
      },
      {
        "product_name": "Product Item 2",
        "product_id": "PRO-JHGF-9876",
        "unitAmt": 2400000,
        "unit": 3,
        "subAmt": 7200000
      }
    ]
  },
  "other": {
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```


pay_rqt_txn_Obj: object

PROPERTIES

txnRef: string (Critical Field) range: (up to 30 chars) required
Unique Transaction Reference generated by Merchant

Example


```
{
  "txnRef": "ORD-438UL748T6"
}
```


pay_rqt_system_Obj: object

PROPERTIES

redirectUrl: string range: (up to 200 chars) required
Define front-end URL for redirecting customer back to merchant site after completing the payment

notificationUrl: string range: (up to 200 chars) required
Define back-end URL for receiving payment result notification from HSBC after payment completed

Example


```
{
  "redirectUrl": "https://www.example.com/redirectBacktoMerchantSite",
  "notificationUrl": "https://www.example.com/receiveNotification"
}
```


pay_rqt_payment_Obj: object

PROPERTIES

country: string enum: [VN] range: (up to 2 chars) required
Country Code (Format: `ISO alpha-2`)

Possible Value	Definition
VN	Vietnam

currency: string enum: [VND] range: (up to 3 chars) required

Example


```
{
  "country": "VN",
  "currency": "VND",
  "payment_option": "all",
  "amount": 10200000,
  "expiry": "2020-05-31T14:10:25Z"
}
```

Payment Currency (Format: `ISO 4217 Alpha`)

Possible Value	Definition
VND	Vietnamese dong

payment_option: string enum: [all, payoo-account, bank-payment, cc, pay-later, pod, pay-transfer, qrcode] range: (up to 64 chars) conditional
To restrict customer payment methods shown in the secured online Payment Page

Possible Value	Definition
all	All available options
payoo-account	Payoo e-wallet
bank-payment	Vietnamese domestic cards/bank accounts
cc	International cards
pay-later	Pay at convenience store
pod	Pay on Delivery
pay-transfer	Pay with transfer
qrcode	Pay with E-wallet app or Bank app

Gateway 1
NOTICE:
`payment_option` is only available and required in Payment Gateway #1

amount: integer range: 1 ≤ x ≤ 99999999999999999999 required
Payment Amount

- It must be in Vietnam Dong. Noted: Do not use comma or dot. For example: Input `100000` instead of `100.000`

expiry: string range: (up to 20 chars) conditional
The period of payment. This is the time that Partner will keep available goods/services for buyer.

- This time must be later than current time
- Format: `yyyy-MM-dd'T'HH:mm:ssZ`

Gateway 1
NOTICE:
`expiry` is only available and required in Payment Gateway #1

pay_rqt_customer_Obj: object

PROPERTIES

name: string range: (up to 128 chars) optional
Customer's Name

email: string range: (up to 64 chars) optional
Customer's Email

phone: string range: (up to 32 chars) optional
Customer's Phone Number

address: string range: (up to 128 chars) optional
Customer's Address

Example

```
{
  "name": "Bùi Khánh An",
  "email": "customer.name@example.com",
  "phone": "091-1111111",
  "address": "xxxxxxx"
}
```

pay_rqt_order_Obj: object

PROPERTIES

shippingDate: string range: (up to 10 chars) conditional
This is the date that shopping website intends to deliver goods/services to the buyer. Format: `yyyy-MM-dd`

Gateway 1
NOTICE:
`shippingDate` is only available and required in Payment Gateway #1.

shippingDays: integer range: 0 ≤ x ≤ 999 conditional
This is the total shipping days. It may equal 0. In that case the order status will change to "Shipping" immediately after it has been paid.

Gateway 1
NOTICE:
`shippingDays` is only available and required in Payment Gateway #1.

description: string range: (up to 100 chars) optional
A brief Order Description to be displayed in the Settlement Report

descriptions: Array< descriptionObj > range: (up to 50 objects) required
An array of detailed Product Descriptions

Example

```
{
  "shippingDate": "2019-01-01",
  "shippingDays": 5,
  "description": "Proceed check out for your order #ORD-438UL748T6",
  "descriptions": [
    {
      "product_name": "Product Item 1",
      "product_id": "PRO-ASDF-1234",
      "unitAmt": 1500000,
      "unit": 2,
      "subAmt": 3000000
    },
    {
      "product_name": "Product Item 2",
      "product_id": "PRO-JHGF-9876",
      "unitAmt": 2400000,
      "unit": 3,
      "subAmt": 7200000
    }
  ]
}
```

descriptionObj: object

PROPERTIES

product_name: string range: (up to 200 chars) required
Product Item Name / Description

product_id: string (Critical Field) range: (up to 50 chars) required
Product Numner / ID

unitAmt: integer range: 1 ≤ x ≤ 99999999999999999999 required
Unit Amount of each item

- It must be in Vietnam Dong. Noted: Do not use comma or dot. For example: Input `100000` instead of `100.000`

unit: integer range: 1 ≤ x ≤ 9999 required
No. of Unit

subAmt: integer range: 1 ≤ x ≤ 99999999999999999999 required
Sub Amount of the Sum of one particular item with mulitple orders. Namly, Unit Amount x Unit

- It must be in Vietnam Dong. Noted: Do not use comma or dot. For example: Input `100000` instead of `100.000`

Example

```
{
  "product_name": "Product Item 1",
  "product_id": "PRO-ASDF-1234",
  "unitAmt": 1500000,
  "unit": 40,
  "subAmt": 60000000
}
```

pay_rqt_other_Obj: object

PROPERTIES

Example

udfs: Array< udfsObj > range: (up to 50 objects) optional
Array of User Defined Fields

```
{
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoker"
    }
  ]
}
```

udfsObj: object

PROPERTIES

definition: string range: (up to 1024 chars) optional
Merchant Defined Definition

value: string range: (up to 2048 chars) optional
Merchant Defined Value

Example

```
{
  "definition": "Special Notes from Customer",
  "value": "Customer is a non-smoker"
}
```

paymentRespModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required
PROPERTIES
transaction: pay_rpn_txn_Obj required
system: pay_rpn_system_Obj required

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "ORD-438UL748T6"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "sysDatetime": "2019-01-05T15:20:45+07:00",
      "qr_code": "https://qr-gw-sb.payoo.vn:8712/QRLink_GatewayWCFService/REST/GetQRCode2?
QRCodeKey=xxxx",
      "redirectLink": "https://newsandbox.payoo.com.vn/v2/paynow/detail?_token=xxxx"
    }
  }
}
```

pay_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Returning back Transaction Reference

Example

```
{
  "txnRef": "ORD-438UL748T6"
}
```

pay_rpn_system_Obj: object

PROPERTIES

sysCode: string range: (up to 6 chars) required
System Return Code

Possible Value	Definition
000000	Request Successful
900030	Duplicate Transaction Reference
999999	*(Other system failure. Error message may be varied.)* **Example:** Invalid start ship date (Verify field `shippingDate`) Invalid validity date (Verify field `payment_expiry`)

sysMsg: string range: (up to 128 chars) required
Corresponding Text Message of Process Return Code

sysDatetime: string range: (up to 25 chars) optional
Time of sending out this request / response

- Server system time. A `GMT+7` timezone information is appended to the end of the timestamp to indicate this time is a Vietnam local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

qr_code: string range: (up to 1024 chars) optional
Return Payment QR Code Image Link or Base64-encoded Image

> Gateway 1
> **INFORMATION:**
> `qr_code` is only available in Payment Gateway #1.

redirectLink: string range: (up to 1024 chars) optional
Return redirect URL of the Online Payment Gateway

Example

```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful",
  "sysDatetime": "2019-01-05T15:20:45+07:00",
  "qr_code": "https://qr-gw-sb.payoo.vn:8712/QRLink_GatewayWCFService/REST/GetQRCode2?
QRCodeKey=xxxx",
  "redirectLink": "https://newsandbox.payoo.com.vn/v2/paynow/detail?_token=xxxx"
}
```

enquiryRespModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required
PROPERTIES
system: enq_rpn_sys_Obj required
transaction: enq_rpn_txn_Obj required
payment: enq_rpn_payment_Obj required
refunds: Array< enq_rpn_refund_Obj > optional

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "transaction": {
      "txnRef": "ORD-438UL748T6",
      "error_code": "53"
    },
```

```
    "payment": {
      "status": "4",
      "payment_option": "1",
      "amount": 500000,
      "payment_datetime": "2019-12-12T14:10:25+07:00",
      "bank_name": "VISA",
      "mcn": "411111******1111",
      "payment_code": "11815866"
    },
    "refunds": [
      {
        "id": "CT201801302",
        "status": "0",
        "amount": 5000,
        "datetime": "2018-12-12T14:10:25+07:00"
      }
    ]
  }
}
```


enq_rpn_sys_Obj: object

PROPERTIES

sysCode: string range: (up to 6 chars) required
System Return Code

Possible Value	Definition
000000	Request Successful
100010	Transaction is Pending
900010	Transaction Record Not Found
999999	Request Fail

sysMsg: string range: (up to 128 chars) required
System Return Status. This is the corresponding message of System Return Code.

Example


```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful"
}
```


enq_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Returning Unique Transaction Reference

error_code: string range: (up to 10 chars) conditional
Error Code. Please see details in here.

Gateway 2
NOTICE:
`error_code` is only available in Payment Gateway #2.

Example


```
{
  "txnRef": "ORD-438UL748T6",
  "error_code": "53"
}
```


enq_rpn_payment_Obj: object

PROPERTIES

status: string range: (up to 10 chars) required
Payment Status

Gateway 1
INFORMATION:
Please see definition in here.

Gateway 2
INFORMATION:
Please see definition in here.

payment_option: string range: (up to 10 chars) optional
Returning Payment Method

Gateway 1
INFORMATION:
`payment_option` is only available in Payment Gateway #1 and returned if the payment status is not pending.

Possible Value	Definition
1	Payoo E-wallet
2	International card
3	Domestic card
4	Pay at convenience store
5	QR Code payment (Pay with E-wallet app or Bank app)
7	Pay on Delivery
8	Pay Transfer

amount: integer range: 1 ≤ x ≤ 99999999999999999 required
Payment Amount

- It must be in Vietnam Dong. Noted: Do not use comma or dot. For example: Input `100000` instead of `100.000`

payment_datetime: string range: (up to 25 chars) optional
Returning Transaction time for the inward credit payment

- Bank system local time. A `GMT+7` timezone information is appended to the end of the timestamp to indicate this time is a Vietname local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

Gateway 1
INFORMATION:
`payment_option` is only available in Payment Gateway #1.

bank_name: string range: (up to 128 chars) optional
Bank Name / Brand Name

Gateway 1
INFORMATION:
`bank_name` is only available in Payment Gateway #1.

mcn: string range: (up to 32 chars) optional
Masked Credit Card Number

- First 6 and last 4 digits of credit card number

Example


```
{
  "status": "4",
  "payment_option": "1",
  "amount": 500000,
  "payment_datetime": "2019-12-12T14:10:25+07:00",
  "bank_name": "VISA",
  "mcn": "411111******1111",
  "payment_code": "11815866"
}
```

Gateway 1

INFORMATION:
`mcn` is only available in Payment Gateway #1.

payment_code: string range: (up to 16 chars) optional
Returning Payment Code / Billing Code

Gateway 1

INFORMATION:
`payment_code` is only available in Payment Gateway #1.

enq_rpn_refund_Obj: object

PROPERTIES

id: string range: (up to 64 chars) required
Refund ID

status: string range: (up to 10 chars) required
Refund status

Gateway 1

INFORMATION:
Please see definition in here.

Gateway 2

INFORMATION:
Please see definition in here.

amount: integer range: 1 ≤ x ≤ 999999999999 required
Returning Refund Amount

datetime: string range: (up to 25 chars) required
Time of sending out this request

- Server system time. A `GMT+7` timezone information is appended to the end of the timestamp to indicate this time is a Vietnam local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

Example

```
{
  "id": "CT201801302",
  "status": "0",
  "amount": 5000,
  "datetime": "2018-12-12T14:10:25+07:00"
}
```

refundReqtModel: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Merchant to pass Transaction Reference that refers to one specific transaction

rfdRef: string range: (up to 30 chars) optional
Merchant can optionally assign an unique Refund Reference Number for every refund transaction. The number will then be displayed in the field `id` under `refund` entity, otherwise the `id` will be assigned by payment gateway.

reason: string range: (up to 256 chars) required
Reason of Refund

amount: integer range: 1 ≤ x ≤ 9999999999999999999 required
Refund Amount

- It must be in Vietnam Dong. Noted: Do not use comma or dot. For example: Input `100000` instead of `100.000`

NOTICE:
Submitting a full amount will be considered a payment cancellation or a full refund. (Depending on settlement status)

Submitting a partial amount will be considered a partial refund. (Payment cancellation will not be proceeded even eligible)

Example

```
{
  "txnRef": "ORD-438UL748T6",
  "rfdRef": "RFD-438UL748T6",
  "reason": "Item is damaged",
  "amount": 550000
}
```

refundRespModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required

PROPERTIES

system: refund_rpn_sys_Obj required
transaction: refund_rpn_txn_Obj required
refund: refund_rpn_refund_Obj optional
Returned only if successful

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "transaction": {
      "txnRef": "ORD-438UL748T6",
      "amount": 550000
    },
    "refund": {
      "id": "CT201801302",
      "status": "0",
      "amount": 550000
    }
  }
}
```

refund_rpn_sys_Obj: object

PROPERTIES

sysCode: string range: (up to 6 chars) required
System Return Code

Possible Value	Definition
000000	Request Successful
900010	Transaction Record Not Found
900040	Duplicate Refund Transaction Reference
999999	System Error

sysMsg: string range: (up to 128 chars) required
System Return Status

Example

```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful"
}
```

refund_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Return Transaction Reference

amount: integer range: 1 ≤ x ≤ 99999999999999999999 required
Original Transaction Amount

- It must be in Vietnam Dong. Noted: Do not use comma or dot. For example: Input `100000` instead of `100.000`

Example

```
{
  "txnRef": "ORD-438UL748T6",
  "amount": 550000
}
```

refund_rpn_refund_Obj: object

PROPERTIES

id: string range: (up to 64 chars) required
ID of refund transaction used in Settlement Report

status: string range: (up to 10 chars) required
Refund Status

Gateway 1
INFORMATION:
Please see definition in here.

Gateway 2
INFORMATION:
Please see definition in here.

amount: integer range: 1 ≤ x ≤ 99999999999999999999 required
Refunded Amount

Example

```
{
  "id": "CT201801302",
  "status": "0",
  "amount": 550000
}
```

statusRtnReqtModel: object

PROPERTIES

transaction: notif_rqt_txn_Obj required
merchant: notif_rqt_merchant_Obj required
payment: notif_rqt_payment_Obj required
other: notif_rqt_other_Obj optional

Example

```
{
  "transaction": {
    "txnRef": "ORD-438UL748T6"
  },
  "merchant": {
    "merId": "723"
  },
  "payment": {
    "status": "PAYMENT_RECEIVED",
    "amount": 500000,
    "payment_option": "INTERNATIONAL_CARD",
    "payment_code": "11815866"
  },
  "other": {
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```

notif_rqt_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Returning Transaction Reference

Example

```
{
  "txnRef": "ORD-438UL748T6"
}
```

notif_rqt_merchant_Obj: object

PROPERTIES

merId: string range: (up to 20 chars) required
Returning Merchant ID

Example

```
{
  "merId": "723"
}
```

notif_rqt_payment_Obj: object

PROPERTIES

status: string range: (up to 32 chars) required
Returning Payment Status

Gateway 1
INFORMATION:
Please see definition in here.

Gateway 2
INFORMATION:
Please see definition in here.

amount: integer range: 1 ≤ x ≤ 99999999999999999999 required
Returning Payment Amount

payment_option: string range: (up to 64 chars) conditional
Returning Payment Method

Gateway 1
INFORMATION:
`payment_option` is only available in Payment Gateway #1.

Possible Value	Definition
E_WALLET	Payoo E-Wallet account

Example

```
{
  "status": "PAYMENT_RECEIVED",
  "amount": 500000,
  "payment_option": "INTERNATIONAL_CARD",
  "payment_code": "11815866"
}
```

Possible Value	Definition
INTERNATIONAL_CARD	International Credit Card/Debit Card
INTERNAL_CARD	Vietnamese's Domestic card/bank account
POD	Pay on Delivery
IN-STORE	Pay in store
PAY-TRANSFER	Pay with transfer
QRCODE	Pay with E-wallet app or Bank app

payment_code: string range: (up to 16 chars) optional
Returning Offline Payment Code / Billing Code

Category 1

INFORMATION:
`payment_code` is only available in Payment Gateway #1 and returned if it is an offline payment.

notif_rqt_other_Obj: object

PROPERTIES

udfs: Array< udfsObj > range: (up to 50 objects) optional
Array of User Defined Fields

Example

```
{
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoker"
    }
  ]
}
```

statusRtnRespModel: object

PROPERTIES

status: string range: (up to 30 chars) required
Return Message

Example

```
{
  "status": "SUCCESS"
}
```

Lifecycle of Cryptographic Keys

This section highlights the Lifecycle of cryptographic keys in the following stages:

1. Generate keys pair (Private Key and Public Key Certificate)
2. ***Optional:*** *Export CSR (Certificate Signing Request) and sign using a CA (Certificate Authority)*

> **DID YOU KNOW?**
> In public key infrastructure (PKI) systems, a certificate signing request is a message sent from an applicant to a certificate authority in order to apply for a digital identity certificate. It usually contains the public key for which the certificate should be issued.

3. Exchange Certificate with HSBC
4. Certificate and Keys Maintenance
5. Certificate and Keys Renewal Process

The Key Renewal Process Command line tool **Java Keytool™** is used in the demonstration. The tool can generate public key / private key pairs and store them into a Java KeyStore. The Keytool executable is distributed with the **Java SDK (or JRE)™**, so if you have an SDK installed you will also have the Keytool executable. The Merchant is free to choose any other tool to generate and manage keys, such as **OpenSSL™**.

Key Generation and Certificate Exchange with HSBC

1. Create a new keys pair (Private Key and Public Key Certificate) with a new or existing Keystore.

```
keytool -genkey
    -alias merchant_key_pair
    -keyalg RSA
    -keystore merchant_keystore.jks
    -keysize 2048
    -validity 3650
    -storepass <your keystore password>
```

- **-genkey** - command to generate keys pair.
- **-alias** - define the alias name (or unique identifier) of the keys pair stored inside the keystore.
- **-keyalg** - key algorithm, it must be `RSA` regarding to HSBC standard. If `RSA` is taken, the default hashing algorithm will be `SHA-256`.
- **-keystore** - file name of the keystore. If the file already exists in your system location, the key will be created inside your existing keystore, otherwise, a new keystore with the defined name will be created.

> **DID YOU KNOW?**
> Keystore is a password-protected repository of keys and certificates. A file with extension `jks` means it is a Java Keystore which is originally supported and executable with Java™.
>
> There are several keystore formats in the industry like `PKCS12` with file extension `p12` which is executable with Microsoft Windows™, merchant can always pick the one most fit their application.

- **-keysize** - key size, it must be `2048` regarding to HSBC standard.
- **-validity** - the validity period of the private key and its associated certificate. The unit is `day`, 3650 means 10 years.
- **-storepass** - password of the keystore.

1.1. Provide the `Distinguished Name` information after running the command:

```
Information required for CSR generation
-----------------------------------------------------------
What is your first and last name?
  [Unknown]:  MERCHANT INFO
What is the name of your organizational unit?
  [Unknown]:  MERCHANT INFO
What is the name of your organization?
  [Unknown]:  MERCHANT INFO
What is the name of your City or Locality?
  [Unknown]:  HK
What is the name of your State or Province?
  [Unknown]:  HK
What is the two-letter country code for this unit?
  [Unknown]:  HK
Is CN=XXX, OU=XXX, O=XXX, L=HK, ST=HK, C=HK correct? (type "yes" or "no")
  [no]:  yes

Enter key password for <merchant_key_pair>
        (RETURN if same as keystore password):
Re-enter new password:
```

> **NOTE:**
> The Private Key password and Keystore password can be identical, however to be more secure, the Merchant should set them differently.

2. **Optional:** Export CSR and get signed with CA. This step can be skipped if the Merchant decides to work with a Self-Signed Certificate.

```
keytool -certreq
    -alias merchant_key_pair
    -keyalg RSA
    -file merchant_csr.csr
    -keystore merchant_keystore.jks
```

- **-certreq** - command to generate and export CSR.
- **-alias** - the name of the associated keys pair.
- **-keyalg** - key algorithm, it must be `RSA` regarding to HSBC standard.
- **-file** - file name of the CSR. This will be generated at the location where the command is run.
- **-keystore** - specify the keystore which you are working on.

2.1. Select and purchase a plan at Certificate Authority and then submit the CSR accordingly. After a signed Certificate is issued by CA, import the Certificate back to the Merchant's keystore.

```
keytool -import
    -alias merchant_signed_cert_0001
    -trustcacerts -file CA_signed_cert.p7b
    -keystore merchant_keystore.jks
```

- **-import** - command to import object into a specific keystore.
- **-alias** - define the alias name (or unique identifier) of the signed Certificate.
- **-trustcacerts -file** - specify the file name of the signed Certificate in Merchant's local file system.

> **NOTE:**
> `PKCS#7` is one of the common formats that contains certificates and has a file extension of `.p7b` or `.p7c`. The certificate format may be varied depending on the policy of the issuing CA.

- **-keystore** - specify the keystore which you are working on.

3. Export the Certificate and send it to HSBC for key exchange.

> **DID YOU KNOW:**
> A Certificate or Public Key Certificate is an electronic document that contains a public key and additional information that prove the ownership and maintains integrity of the public key. It is essential for the sender to ensure the key is not altered by any chance during delivery.

```
keytool -export
    -alias merchant_key_pair
    -file merchant_cert_0001.cer
    -keystore merchant_keystore.jks
```

- **-export** - command to export object from a specific keystore.
- **-alias** - the name of the associated keys pair.

> **NOTE:**
> If the Merchant associates the original keys pair `merchant_key_pair`, the exported Certificate is without CA-signed, and hence, Self-Signed. However, if the Merchant associates the imported Certificate `merchant_signed_cert_0001` mentioned in step #2, the exported Certificate is CA-signed.

- **-file** - specify the file name of the Certificate where the file will be exported to Merchant's local file system.

> **NOTE:**
> The default Certificate file encoding is binary. HSBC accepts both binary and base64 encoding. To export a printable base64 encoding file, please attach an extra parameter `-rfc` in the command.
> e.g. `-file merchant_cert_0001.crt -rfc`.

- **-keystore** - specify the keystore which you are working on.

4. Import HSBC's Certificate into the merchant's Keystore.

```
keytool -import
    -alias hsbc_cert_0002
    -file hsbc_cert_0002.cer
    -keystore merchant_keystore.jks
```

- **-import** - command to import object into a specific keystore.
- **-alias** - define the alias name of HSBC's Certificate in your keystore.
- **-file** - specify the file name of HSBC's Certificate in Merchant's local file system.
- **-keystore** - specify the keystore which you are working on.

5. **Optional:** List keystore objects. Merchant is suggested to verify that all required objects are properly maintained. 2 - 3 entries should be found in your Java Keystore: *(Entries may be varied if other key repository format is used)*

Alias name	Corresponding Object	Remark
merchant_key_pair	• Merchant's Private Key • Merchant's Public Certificate (Self-Signed)	These two objects appear to be one entry in a JAVA Keystore. Merchant can still export them separately into two objects (files) on your local file system depending on your application design.
merchant_signed_cert_0001	• Merchant's Public Certificate (CA-Signed)	Not exist if Merchant skips step #2
hsbc_cert_0002	• HSBC's Public Certificate	

```
keytool -list -v -keystore merchant_keystore.jks

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

Alias name: merchant_key_pair
Creation date: Jan 1, 2020
Entry type: PrivateKeyEntry

<Other Information>

*******************************************
*******************************************

Alias name: merchant_signed_cert_0001
Creation date: Jan 1, 2020
Entry type: trustedCertEntry

<Other Information>

*******************************************
*******************************************

Alias name: hsbc_cert_0002
Creation date: Jan 1, 2020
Entry type: trustedCertEntry

<Other Information>

*******************************************
*******************************************
```

Certificates and Keys Maintenance

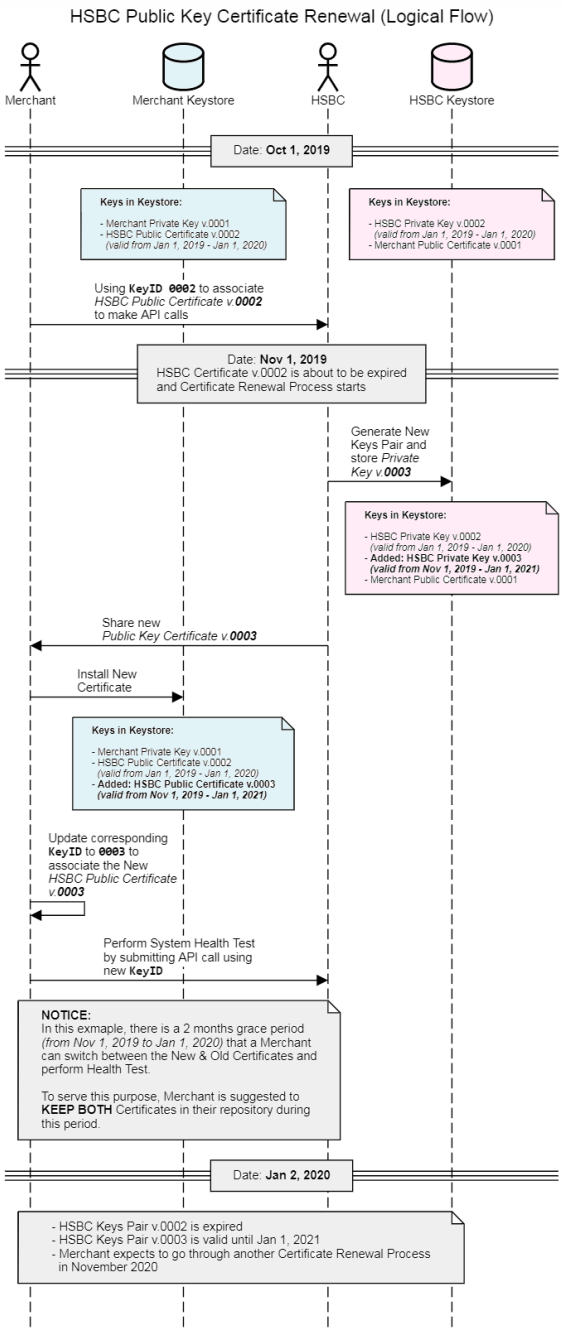
Here are some recommendations to Merchant of how to properly maintain certificates and keys:

Component	Storage	Validity
Merchant's Private Key	Private Key should be maintained and handled with the most secure approach that a Merchant can apply. The most common and yet secure enough approach is: • **key password** - Do not save the password in plain text or hard-coded in application. Recommend to encrypt it by any Password Encryption Tools • **key storage** - Store inside password-protected key repository, such as `JKS` or `PKCS12` keystore. Keystore password should also be encrypted.	No restriction on the Validity Period. However, if Merchant suspects there is any chance that the key is leaked or for any other security reason, a new Private Key and its associated Public Key Certificate should be generated.
Merchant's Public Key Certificate	Since Public Key Certificate is publicly distributed, a comparative moderate secure storage approach is acceptable. Merchant can store the physical file in any system's file system or store all keys and certificates in one single key repository for a centralised key management.	For a self-signed Certificate, the same condition has been mentioned as above. However, the validity period of a CA-signed Certificate is depended on the purchase plan of the issuing CA. The most common standard is 1 to 2 years.
HSBC's Public Key Certificate	Same as the above	1 Year **NOTE:** Technically, the validity period is usually 1 Year plus 1 to 2 months more. The spare period is a buffer for a merchant to switch a "to-be-expired" Certificate to the new one during the Certificate Renewal Process. More technical detail will be covered in later section.

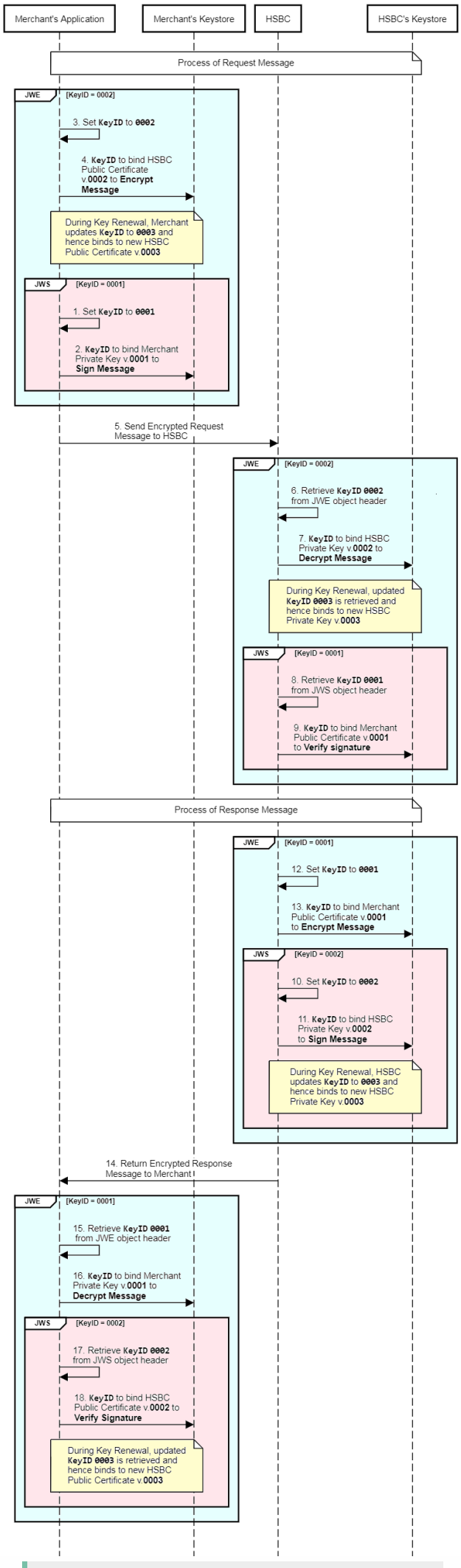
Certificates and Keys Renewal

Every Public Key Certificate has an expiration date. When either the Merchant's or HSBC's Certificate is about to expire, a key renewal process takes place. Please see the Key Renewal Process Flow below:

> ! **SOME RULES YOU SHOULD KNOW:**
> - **Keys Repository:** This is a mock-up for demonstration purpose only.
> - **Keys Name:** Using a `Key Name` `KeyID` naming convention makes for a simpler demonstration. The suggested identifier of one key should be the alias name inside a key repository.
> - **KeyID Value:** HSBC uses the naming convention `0001`, `0002`, `0003`... `n + 1`, each time the HSBC certificate is renewed, the `KeyID` value is `n + 1`.
> - **KeyID Binding:** The binding between the `KeyID` and the corresponding `Keys Pair` in the merchant's system can make use of any key/value logic, such as a Database table. In our example below, KeyID `000X` binds to `Private Key v.000X` and `Public Certificate v.000X`, etc.
> - **Validity Date:** All dates are made-up for demonstration purposes only.



Below is the technical flow showing how `Certificates`, `Alias Names` and `KeyIDs` work together during a normal process or a key renewal process:



> **NOTE:**
> All examples above concern the HSBC Certificate Renewal. Whenever the Merchant needs to renew their Certificate, they need to switch role and steps to follow those of HSBC's.

Error Code of Enquiry

Possible Value	Definition
00	Successful transaction
22	Customer entered wrong OTP on PG
V01	Wrong check_sum
V02	Customer entered wrong OTP on PG
V03	OTP expired
21	Customer entered wrong password (PIN code)
685	Customer entered wrong password (PIN code)
16	Customer does not have available balance for payment
W04	The timeout connection (including the case where the user does not operate on the PG web server will be redirected back to the return_url after 3 minutes).
V04	Error when query system in VIETTEL
V05	Transaction Not Verified (Call to Partner Transaction Confirmation API failed)
V06	Customer cancelled payment
S_MAINTAIN	PG on maintenance
99	Unknown error
M01	Partner code is not registered (contact Viettel for checking)
M02	Not set up account to receive money for partners (contact Viettel technical)
M03	The payment method is not suitable (contact Viettel technical)
M04	QR images are not valid or image value is not readable
813	Error connecting to PG

Enquiry Status of PG #1

Possible Value	Definition	Remark
-1	Unpaid	• Customer has not completed payment in the online payment page.
0	Processing	• When the Payment Expiry Date is arrived. Merchant can define this date at request field `payment_expiry` in Payment Page Redirect API
2	Shipping	• When the Shipping Date is arrived. Merchant can define this date at request field `shippingDate` in Payment Page Redirect API • Merchant is suggested to set `shippingDate` right after `payment_expiry`
4	Finished	• When the *Delivery Completion Date* is arrived. In another words, that is Shipping Date `shippingDate` plus Total Shipping Days `shippingDays`. • Or the corresponding settled transaction is fully or partially refunded.
1	Suspended	• Updated by backend system due to fraud screening.
3	Cancelled	• The transaction is cancelled. It would only happen before Delivery Completion Date.

Enquiry Status of PG #2

Possible Value	Definition
-1	No transaction occurred
0	transaction pending
1	successful transaction
2	transaction failed
3	the transaction is unclear

Refund Status of PG #1

Possible Value	Definition
0	Success
9	Invalid refund amount.
11	Transaction is not in "processing" status.
12	Transaction in "Cancelled" status.
83	Refund amount less than minimum transfer.
91	Receiver's Status is locked.
92	Refund amount total greater than maximum limit per day.
94	Status of Order is "Pending" (In case, you should contact with Payoo admin)
96	Merchant does not permission to doing a partial refund.
97	Refund amount greater than money total of Order.
98	Status of Order is "Cancelled"
99	Merchant's refund Id is exist.
1000	System error.

Refund Status of PG #2

Possible Value	Definition
00	Successful cancel/refund
M01	Cannot find partner's info. Need to check configures.
M05	Refunding configures is not suitable. Need to check info again.
218	Error of not able to send transaction code of partner
KG3	Cannot find payment transaction corresponded to data that partner sent
27	Transaction not from partner
KG8	Refund amount not valid
176	This payment transaction status is not suitable to continue cancelling/refunding. Cancel/refund might have been successful or not determined or expired… (need to base on error_msg)
485	Payment transaction failed so is not allowed to cancel/refund
655	Not enough data to process refund to customer
457	Not enough data to process refund to customer
17	ViettelPay/Bankplus account of customer does not have enough condition to receive refunds. Contact Viettel.
159	Customer's type of Bankplus account is not supported for refunding
813	Error in cancelling/refunding process in Viettel. Try again later.
927	Error in cancelling/refunding process in Viettel. Try again later.

Notification Status of PG #1

Possible Value	Definition
PAYMENT_PROCESSING	The payment is still in process and it does not have the final result
PAYMENT_RECEIVED	Merchant order has been paid. Merchant can start the delivery procedure

Notification Status of PG #2

Possible Value	Definition
1	successful transaction

Download Swagger

Click here to download Swagger 2.0 file in YAML format.

Disclaimer

IMPORTANT NOTICE