# API Specification of HSBC Omni Collect in Japan

API Base URL

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

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

Schemes: https

Version: 1.2

## Purpose of this document

This document provide the audience with **OpenAPI specification** for describing REST APIs of HSBC Omni Collect in Japan.

The target audience of this document is the Developer, Business Analyst and other related Project Team Member (who has the basic technical know-how of Web technology such as REST or JSON) of HSBC's client (i.e. the Merchant)

## Update Log

- [Mar 19, 2021] **v1.2** Updated API Use Case of Content Section Credit Card
- [Jan 25, 2021] **v1.1** Content Section Revised
- [Now 31, 2020] **v1.0** Initial Version

# How to Read this Document

This document walks through the API usage and lists the key idea by section like API Usage Flow, API Connectivity and API Operation. There is also a FAQ and list of Schema Definitions used by API operation.

# Features and Use Cases

HSBC Omni Collect offers a wide range of online payment solutions which allows online merchants to process Credit / Debit Card and Code Payments. The payment platform supports implementations with websites or mobile applications.

---

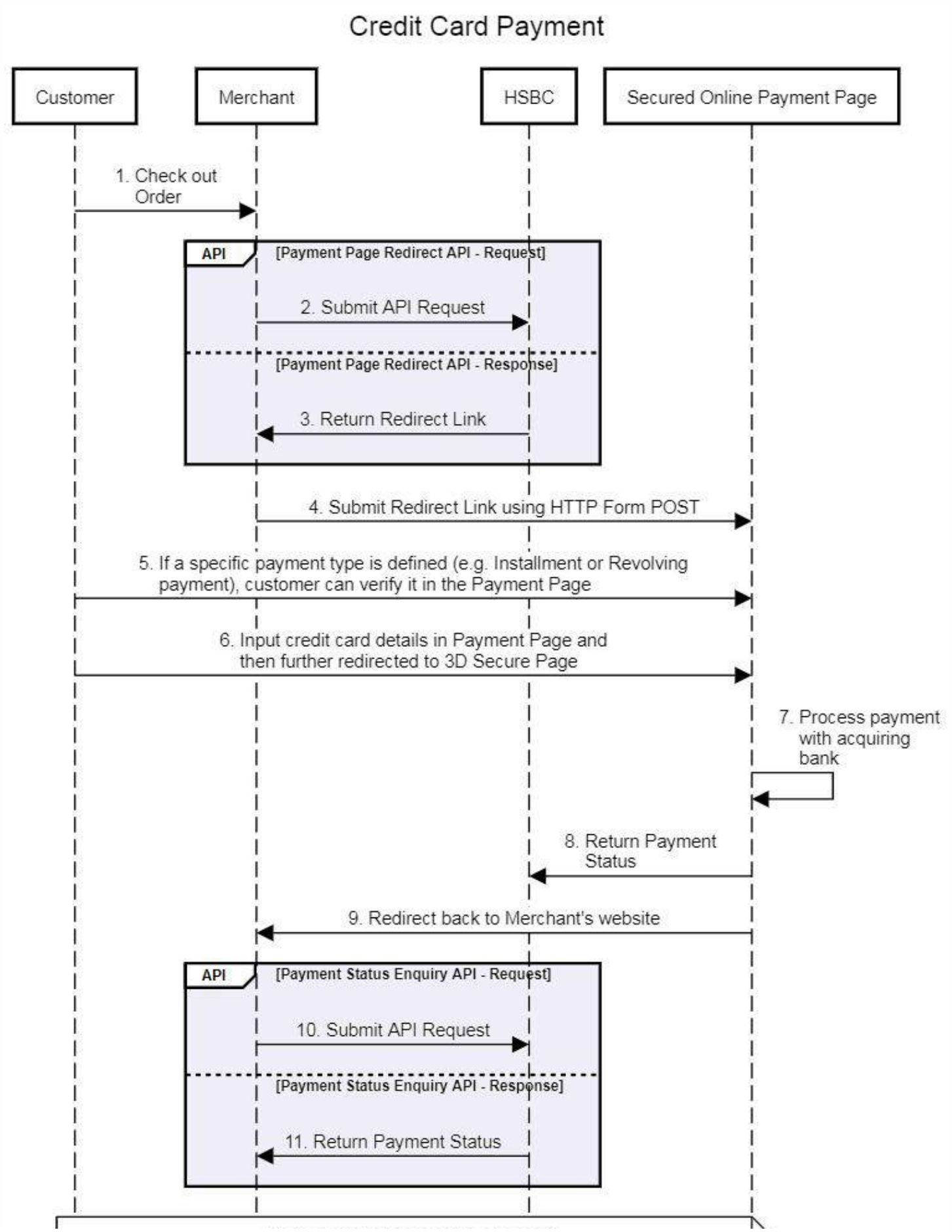
## Credit Card / Debit Card Payments

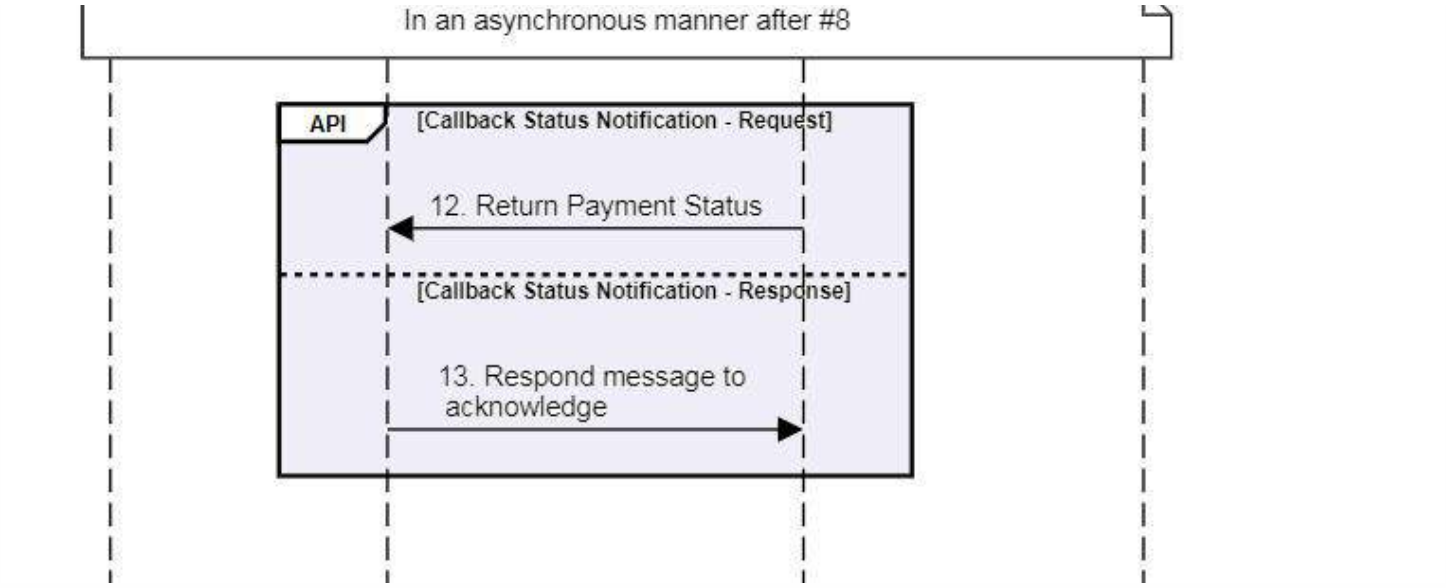
HSBC Omni Collect for Japan currently supports the following card companies:

| **List of supported Card Brands / Companies** | | |
|---|---|---|
| AEON | NC日商連 | ポケットカード |
| American Express | SAISON | 三井住友 |
| APLUS | UC | 三菱UFJニコス (DC) |
| Cedyna (CF) | UCS | 三菱UFJニコス (NICOS) |
| Cedyna (OMC) | Visa | 三菱UFJニコス (UFJ) |
| DINERS | エポス | 京王パスポート |
| JACCS | オリコ | 日専連 |
| JCB | すみしんLIFE | 東急TOP |
| LIFE | トヨタファイナンス | 楽天カード |
| Mastercard | | |

Furthermore, online credit card transaction in Japan is usually required additional security from issuer Bank, called 3D Secure. This process will ask the genuine credit card holder to enter Internet PIN or One Time PIN(OTP) that usually sent to Credit Card Holder mobile phone.

API Use Case


Credit Card Payment
Customer
Merchant
HSBC
Secured Online Payment Page
1. Check out Order
API
[Payment Page Redirect API - Request]
2. Submit API Request
[Payment Page Redirect API - Response]
3. Return Redirect Link
4. Submit Redirect Link using HTTP Form POST
5. If a specific payment type is defined (e.g. Installment or Revolving payment), customer can verify it in the Payment Page
6. Input credit card details in Payment Page and then further redirected to 3D Secure Page
7. Process payment with acquiring bank
8. Return Payment Status
9. Redirect back to Merchant's website
API
[Payment Status Enquiry API - Request]
10. Submit API Request
[Payment Status Enquiry API - Response]
11. Return Payment Status




1. Customer conducts checkout process in merchant's website.
2. Merchant submits Payment Page Redirect API request to HSBC.
3. HSBC returns JSON response which embeds the access link (in HTML Form Submit format) of the Secured Online Payment Page in the field `redirectLink`. More details will be covered in here.
4. Merchant submits the redirect link using HTML Form POST. It will redirect Merchant website to the Secure Online Payment Page.
5. Customer can verify the payment type (one-time, installment or a revolving payment) on the payment page. Merchant can associate an installment or revolving plan on step #2. See more details in here.
6. Customer input Credit Card details in the Payment Page and then further redirected to 3D Secure (3DS) Page for input One-Time password.
7. Payment page will connect securely to bank and backend systems to process the payment.
8. HSBC will receive payment status once it is updated from backend system.
9. Redirect back to merchant website once the payment process is completed in the Payment Gateway.

> ! **NOTICE:**
> Merchant can define this redirect back URL in request fields `redirectUrl` in Payment Page Redirect API.

10. Merchant is recommended to submit a Payment Status Enquiry API right after the Payment Page is redirected back to the Merchant's website.
11. HSBC will return the latest payment status while Merchant can utilize this information to construct their Order Confirmation Page.
12. HSBC will then trigger Callback Status Notification and send payment status back to Merchant ***asynchronously***.

> ! **NOTICE:**
> This server-to-server Notification will be sent out for a successful payment or refund case only. Merchant can define their URL endpoint in request field `notificationUrl` in Payment Page Redirect API.

13. Merchant responds the API to acknowledge. Fail to return a proper response will trigger Notification resend mechanism.

# Installment and Revolving Payments

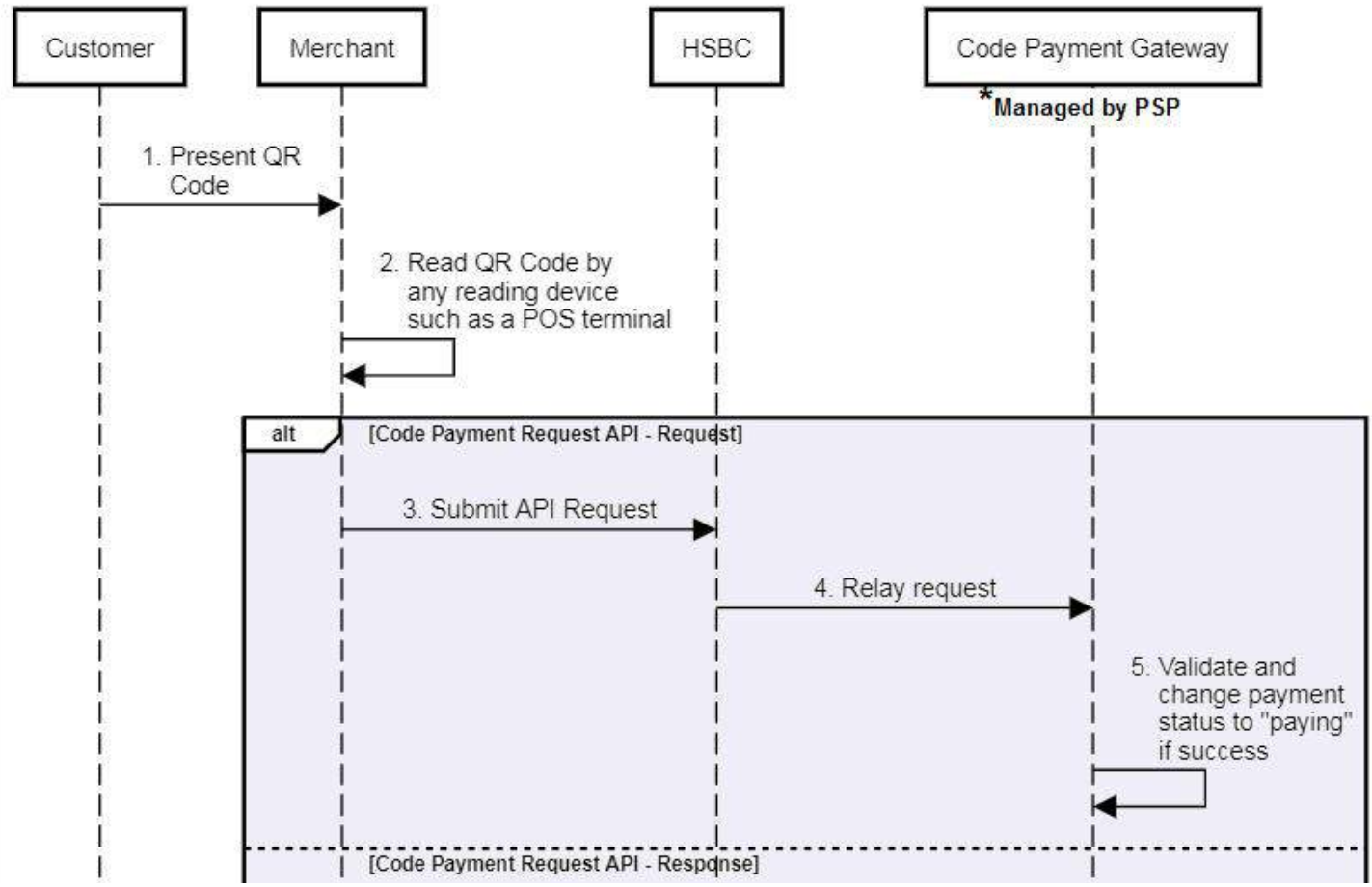
To allow customer to submit an installment or revolving payment request, merchant can either create a new Plan or reuse an existing plan and put the corresponding `plan_id` into Payment Page Redirect API and follow the same API flow as we mentioned in the previous section.

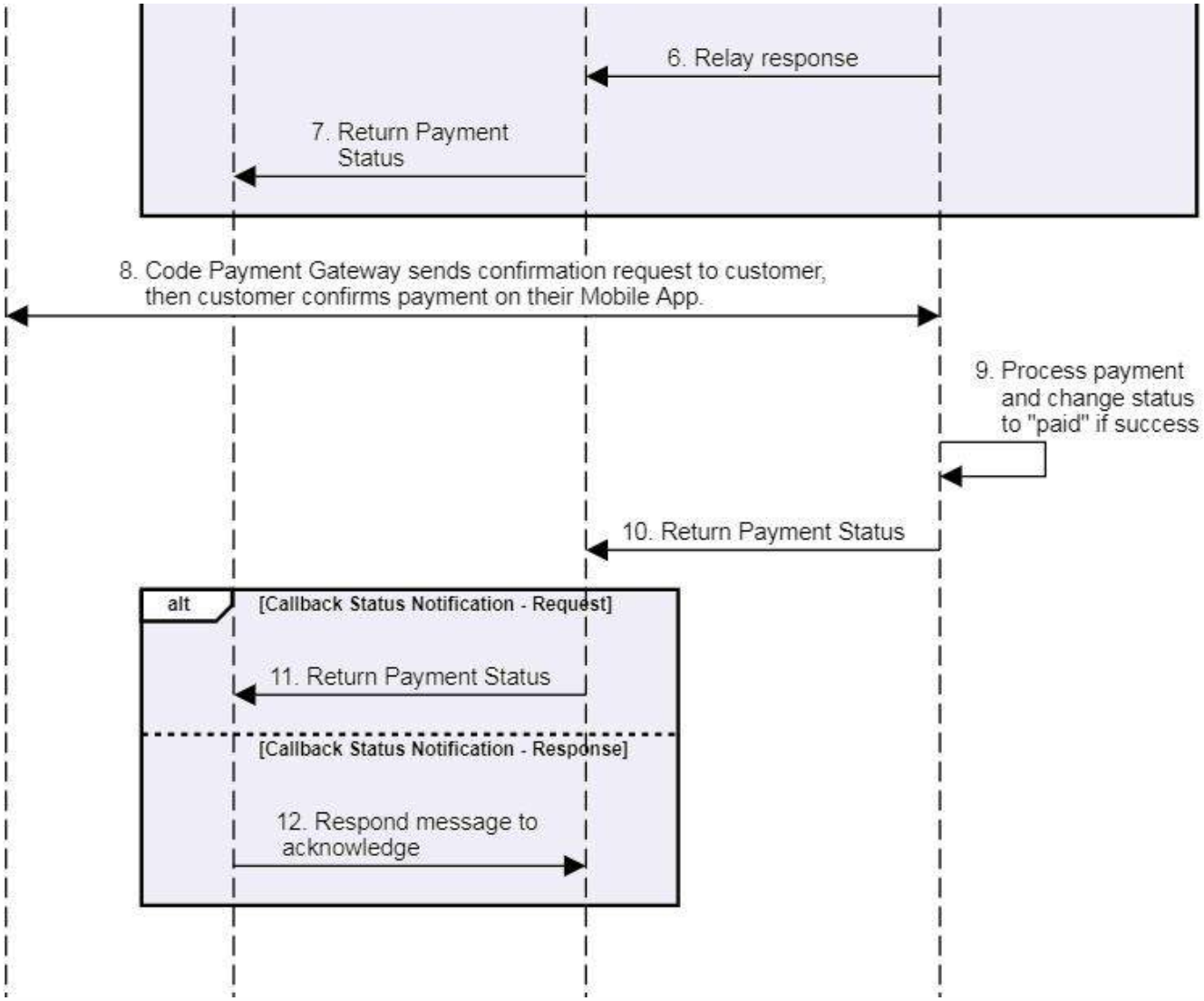
> **NOTICE:**
> Full refund is supported for both Installment and Revolving payment. Moreover, a refund request for a revolving also means to terminate the subscription.
>
> Callback Status Notification will be sent for the first revolving payment submission only.

# Code Payment

To see the list of Code Brands / Companies currently supported by HSBC Omni Collect for Japan, please refer to the API field `type` of API Schema code_Obj.

## API Use Case






1. Customer presents QR code to Merchant.
2. Merchant reads QR code with any reading device such as POS terminal.
3. Merchant decodes the QR code image into a string and submit Code Payment request to HSBC
4. HSBC relays request to Code Payment Gateway.
5. Code Payment Gateway processes validation and changes payment status to `paying` if success
6. HSBC relays response from Code Payment Gateway.
7. API returns corresponding payment status.
8. Code Payment Gateway sends confirmation request to customer, then customer confirms payment on their Mobile App.
9. Code Payment Gateway processes payment and changes status to `paid` if success
10. HSBC relays response from Code Payment Gateway.
11. HSBC pushes payment result to Merchant.
12. Merchant responds the API to acknowledge. Fail to return a proper response will trigger Notification resend mechanism.

## Check Status Feature

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

## Cancel & Refund

Merchant can request Refund API to refund a settled transaction (Card Company recorded). HSBC currently accepts Full Refund only.

## Order Confirmation

Regarding to the aforementioned API usage flow, the last step is to redirect the Payment Page back to the Merchant website. Merchant can build a dynamic Order Confirmation Page with payment status (e.g. successful or failed) where the details can be retrieved from the immediate Payment Status Enquiry API or the asynchronous Callback Status Notification.

# 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<br>• Password |
| | Locate API Gateway Policy of the corresponding user | • Client ID<br>• Client Secret |
| **User Identification** | A Merchant Profile | • Merchant ID<br>• Merchant Profile |

| | | |
|---|---|---|
| **Connection Security** | HTTPS Connection (TLS 1.2) and Network Whitelisting | • SSL Certificate<br>• Network Whitelist |
| **Message Security** | Digital Signing and Data Encryption | • A pair of Private Key & Public Key Certificate (PKI Model)<br>• JWS Key ID<br>• JWE Key ID |

---

# 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:<br>`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:<br>`x-hsbc-client-id: [Client ID]` | In HTTP header:<br>`x-hsbc-client-secret: [Client Secret]` |

---

# User Identification

| **Merchant Profile & Merchant ID** | |
|---|---|
| • Merchant Profile contains all necessary information from a | • Merchant ID is used for Merchant |

| | | |
|---|---|---|
| **Purpose** | Merchant in order to enable payment service. | 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* |
| **Implementation** | *nil* | *nil* | • Merchant's domain URL will be configured in HSBC's network whitelist by HSBC team |

# Message Security - Data Encryption and Signing

On top of the Transport Layer Security, HSBC adopts additional security on the message being passed through

the connection session. Data Encryption actually serves as a locked briefcase containing the data (the API message) within the HTTPS "tunnel". In other word, the communication has double protection.

> **DO YOU KNOW?**
> Javascript Object Signing and Encryption **(JOSE™)**, is a framework intended to provide methods to securely transfer information between parties. The JOSE framework provides a collection of specifications, including JSON Web Signature **(JWS™)** and JSON Web Encryption **(JWE™)**, to serve this purpose.

HSBC uses JWS to sign message payload and JWE to encrypt the signed message while these two objects are created by using a pair of Private Key & Public Key Certificate (PKI Model).

| **Private Key & Public Key Certificate (PKI Model)** | | |
|---|---|---|
| **Purpose** | • Digitally sign a API request message<br>• Decrypt a API response message | • Encrypt the signed API request message<br>• 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 | |

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

| **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, and;<br>• In HTTP header:<br>`x-hsbc-msg-encrypt-id: [Merchant ID]+[JWS ID]+[JWE ID]` | |

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

---

# How to Sign and Encrypt Outgoing Message

Every message sent to HSBC must be signed and encrypted. From the point of view of a Merchant, an **Outgoing Message** means:

- the Request Message of a Normal 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. Do not worry if you are not familiar with Java, the idea is to let you know the steps and all needed components:

> ! **NOTICE:** These Java codes are for demonstration only and 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").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 **JWS Header** using `RS256` signing algorithm and **JWS keyID**, in this case, `0001`

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

Next, you are going to **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 **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 **JWE Object** by combining JWE Header and JWE Payload
4. Retrieve **HSBC's Public Key** as the encrypter
5. Create **Encrypted JWE Object** by encrypted it with HSBC's Public Key

Yes, you are now ready to put the Encrypted JWE Object as 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 point of view of a Merchant, an **Incoming Message** means:

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

Let's look into the following example to see how you 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 **Encrypted JWE Object** by parsing the encrypted response message payload
2. Retrieve **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. Yet, before that, 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 **JWS Object** by parsing the `Signed Message`
2. Retrieve **HSBC's Public Key** as the verifier
3. Verify the signed JWS Object. Invoke error handling if invalid signature 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 API Request

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

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

However, data encryption is actually a required data security imposed by HSBC standard, Merchant has to invoke the encryption logic before moving to Production and fully tested during testing phase.

---

# Make Your API Request with Plain Messages

> ! **NOTICE:**
> Skipping message encryption is the flexibility provided in Sandbox Environment for testing purpose.

**Submit API request using cURL™ as an example**
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 in your system platform:

```
curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcjp-ea-merchantser
 -H "message_encrypt: false"
 -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
 -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
 -H "Content-Type: application/json"
 -d "{ \"txnRef\": \"PAY-QJZV956664\", \"merId\": \"42298549900001\"}"
```

1. Submit `POST` request to the API URL endpoint
2. Put 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 Client ID in HTTP header `x-HSBC-client-id`
5. Put Client Secret in HTTP header `x-HSBC-client-secret`
6. Put Merchant ID, JWS ID and JWE ID in HTTP header `x-HSBC-msg-encrypt-id` respectively
7. Set `Content-Type` to JSON format
8. Plain `json` message payload

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

---

# Making API Request with Message Encryption

**Step 1.** Run this cURL™ command in your system platform:

```
#1 curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcjp-ea-merchantser
#2  -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#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-WGPZt
```

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

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

**Step 2.** For a successful request (HTTP Status Code 200), an encrypted response message will be returned, otherwise, a plain `json` with failure message will be 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.<br><br>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"`.<br><br>**List of Critical Fields:**<br>`txnRef`<br>`merId`<br>`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.<br><br>Field & value must be present in the request with valid `JSON` format. |
| Optional | Annotated with optional tag in field definition section.<br><br>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.<br><br>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+10` (Australia local time) or `GMT+8` (Singapore 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`.<br><br>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 HSBC SSL server certificates?

Merchant developer is able to export SSL server certificates that has been installed in your browser. By doing this, visit the **domain** of the corresponding API endpoint in your browser. For example, to get the SSL certificate of sandbox environment, use domain name https://devclustercmb.api.p2g.netd2.hsbc.com.hk/

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

## Message Encryption Questions

### What certificates will I need to work for Message Encryption in HSBC's sandbox and production environments?

A self-sign certificate is acceptable. However, If Merchant decides to enhance security, a CA-Signed Certificate is always welcome.

## 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 the url does not belong to HSBC, use it on your own discretion. By clicking the url or website, it means you accept this 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 those urls or websites do not belong to HSBC, use it on your own discretion. By clicking those urls or websites, it means you accept this terms and conditions.*

# Payments

Contains resource collections for Credit card and Code payments, enquiry, notification, etc.

Payments

# Payment Page Redirect for Credit Card Payment

POST `/payment/pageRedirect`

DESCRIPTION

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

**How to do Redirection**

Merchant is required to use HTTP Form POST to submit the redirect link which is presented in a `HTML Form` format. Below is a sample, please be noticed any data modification inside the form is not allowed. Otherwise, the data integrity checking will block the connection from accessing the online payment page.

```
<script language="javascript">window.onload=function(){document.pay_form.submit();}</script>
<form id="pay_form" name="pay_form" action="https://www.e-scott.jp/euser/snp/SSNPxxxx.do" method="po
<input name="MerchantId" type="hidden" id="MerchantId" value="000xxxxx" />
<input name="EncryptValue" type="hidden" id="EncryptValue" value="dcbQvu8ged6udFbAzo1xIGat6GmWYOosls
</form>
```

## REQUEST PARAMETERS

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

## REQUEST BODY

| | |
|---|---|
| payLinkReqtModel | *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": "PAY-QJZV956664",
    "tenant_id": "0001",
```

```
    "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
  },
  "system": {
    "redirectUrl": "https://www.example.com/redirect",
    "notificationUrl": "https://www.example.com/notification"
  },
  "payment": {
    "country": "JP",
    "amount": 10000,
    "description": "Payment Order of #PAY-QJZV956664"
  },
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 9000,
      "unit": 1,
      "vat": 1000,
      "subAmt": 10000
    }
  ],
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoker"
    }
  ]
}
```


## RESPONSES

| | |
|---|---|
| **200 OK**<br>payLinkRespModel | Successful operation.<br><br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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": "PAY-QJZV956664"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "sysDatetime": "2020-01-01T13:00:00+09:00",
      "redirectLink": "<Encoded_Redirect_Submit_Form>"
    }
  }
}
```


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

# Code Payment

POST `/payment/code`

## DESCRIPTION

Unlike making credit card payment via an Online Payment Page, this endpoint makes a direct Code payment request.

## REQUEST PARAMETERS

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

REQUEST BODY

| | |
|---|---|
| codeReqtModel | *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": "PAY-QJZV956664",
    "tenant_id": "0001"
  },
  "system": {
    "notificationUrl": "https://www.example.com/notification",
    "qr_str": "<QR_Code_String>"
  },
  "payment": {
    "country": "JP",
    "amount": 10000,
    "description": "Payment Order of #PAY-QJZV956664"
  },
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 9000,
      "unit": 1,
      "vat": 1000,
      "subAmt": 10000
    }
  ],
```

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


## RESPONSES

| | |
|---|---|
| **200 OK**<br>codeRespModel | Successful operation.<br><br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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": "PAY-QJZV956664",
      "tenant_id": "0001",
      "process_id": "ee5b902a153f104281f4b81c5ce8216b",
```

```
      "process_pass": "f1973eef815a6e1541b356ab06e2478c",
      "error_code": "BARCODE_ERROR",
      "error_msg": "正しいバーコードをスキャンしてください。"
    },
    "payment": {
      "id": "000014640567",
      "resp_code": "OK",
      "amount": 650000,
      "description": "Payment Order of #PAY-QJZV956664",
      "datetime": "2020-01-01T13:02:00+09:00"
    },
    "code": {
      "id": "000000002563",
      "type": "3",
      "status": "1",
      "currency": "JPY",
      "amount": 650000
    }
  }
}
```


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 Enquiry

GET `/payment/transaction/{txnRef}`

## DESCRIPTION

HSBC Omni Collect will return the latest transaction status according to the transaction reference number Merchant provides.

## REQUEST PARAMETERS

| | | |
|---|---|---|
| **Authorization** | BASIC [Base64-encoded Credential] | |
| required | | |
| in header | | |

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

## RESPONSES

| | |
|---|---|
| **200 OK**<br>enquiryRespModel | Successful operation.<br><br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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-da00-4883",
    "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": "PAY-QJZV956664",
      "tenant_id": "0001",
      "process_id": "ee5b902a153f104281f4b81c5ce8216b",
      "process_pass": "f1973eef815a6e1541b356ab06e2478c",
      "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
    },
    "payments": [
      {
        "id": "000014640567",
        "resp_code": "OK",
        "approvalNo": "0003000",
        "amount": 100000,
        "description": "Payment Order of #PAY-QJZV956664"
      }
    ],
    "refunds": [
      {
        "id": "RFD-DFCV112233",
        "resp_code": "OK",
        "approvalNo": "0003000",
        "amount": 100000,
        "create_datetime": "2020-01-01T13:02:00+09:00"
      }
    ],
    "code": {
      "id": "000000002563",
      "type": "3",
      "status": "1",
      "currency": "JPY",
      "amount": 650000
    },
    "links": [
      {
        "href": "/plan/@id",
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "rel": "plan",
        "method": "GET"
      }
    ]
  }
}
```


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

# Refund

POST `/payment/refund`

## DESCRIPTION

This API is used to send a refund request for a previously settled transaction. It supports both credit card and code payment.

## 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.* |

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "PAY-QJZV956664",
  "refund_id": "RFD-DFCV112233"
}
```


## RESPONSES

| | |
|---|---|
| **200 OK**<br>refundRespModel | Successful operation.<br><br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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": "PAY-QJZV956664"
    },
    "refund": {
      "id": "RFD-DFCV112233",
      "resp_code": "OK",
      "approvalNo": "0003000",
      "amount": 100000,
      "create_datetime": "2020-01-01T13:02:00+09:00"
    },
```

```
    "code": {
      "id": "000000002563",
      "type": "3",
      "status": "1",
      "currency": "JPY",
      "amount": 650000
    }
  }
}
```


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

# Callback Status Notification

POST `/<Callback_URL_1>`

## DESCRIPTION

Once Omni Collect receives a payment or refund request, subsequent payment status change or update will be returned to Merchant by asynchronous callback until the status is reached to its final state.

| Operation | Intermediate State | Final State |
|---|---|---|
| Credit Card Payment | n/a | `"payment": {"resp_code": "OK"}` |
| Credit Card Refund | n/a | `"refund": {"resp_code": "OK"}` |
| Code Payment | `"code": {"status": "1"}` = Paying | `"code": {"status": "2"}` = Paid |
| Code Refund | `"code": {"status": "3"}` = Refunding | `"code": {"status": "4"}` = Refunded |

**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**<br>required<br>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": "PAY-QJZV956664",
    "tenant_id": "0001",
    "process_id": "ee5b902a153f104281f4b81c5ce8216b",
    "process_pass": "f1973eef815a6e1541b356ab06e2478c",
    "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
  },
  "merchant": {
    "merId": "42298549900001"
  },
  "payment": {
    "id": "000014640567",
    "resp_code": "OK",
    "approvalNo": "0003000",
```

```
    "amount": 100000,
    "description": "Payment Order of #PAY-QJZV956664"
  },
  "refund": {
    "id": "RFD-DFCV112233",
    "resp_code": "OK",
    "approvalNo": "0003000",
    "amount": 100000,
    "create_datetime": "2020-01-01T13:02:00+09:00"
  },
  "code": {
    "id": "000000002563",
    "type": "3",
    "status": "1",
    "currency": "JPY",
    "amount": 650000
  },
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoker"
    }
  ]
}
```


RESPONSES

| | |
|---|---|
| **200 OK**<br>statusRtnRespModel | Successful operation. |

Response Content-Types: application/json

Response Example (200 OK)


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


# Plans

Create an Instalment or Recurring Payment through a Plan. It acts as a reusable template and contains details of

the billing cycle. Depending upon your business, you can create multiple plans with different billing cycles.

Once a plan is created, submit the `Plan ID` in Payment Page Redirect API.

Plans

# Create Plan

POST `/plan`

## DESCRIPTION

Create an instalment or recurring payment plan.

## 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

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

Request Content-Types: application/json

Request Example

```
{
  "type": "I",
  "description": "Monthly Installment Plan #1",
  "total_count": 12
}
```


## RESPONSES

| | |
|---|---|
| **200 OK**<br>createPlanRespModel | Successful operation.<br><br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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"
    },
    "plan": {
      "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
      "type": "I",
      "description": "Monthly Installment Plan #1",
      "total_count": 12,
      "create_date": "2020-01-01T13:02:00+09:00"
    },
    "links": [
      {
```

```
        "href": "/plan/@id",
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "rel": "self",
        "method": "GET"
      }
    ]
  }
}
```


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"
}
```


Plans

# Retrieve All Plans

GET `/plan`

## DESCRIPTION

Use this endpoint to fetch all plans.

## 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** | [Merchant ID]+[JWS ID]+[JWE ID] |

required
in header

**Content-Type**
required
in header

application/json

## RESPONSES

**200 OK**
getPlanRespModel

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",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "plans": [
      {
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "type": "I",
        "description": "Monthly Installment Plan #1",
        "total_count": 12,
        "create_date": "2020-01-01T13:02:00+09:00"
      }
```

```
    ]
  }
}
```

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"
}
```

Plans

# Retrieve Plan by Plan ID

GET `/plan/{plan_id}`

## DESCRIPTION

Use this endpoint to fetch details of a plan by its ID.

## 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**<br>required<br>in header | application/json |
| **plan_id: string**<br>required<br>in path | *Data Encryption is enforced.* |

## RESPONSES

| | |
|---|---|
| **200 OK**<br>getPlanRespModel | Successful operation.<br><br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "plans": [
      {
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "type": "I",
        "description": "Monthly Installment Plan #1",
        "total_count": 12,
        "create_date": "2020-01-01T13:02:00+09:00"
```

```
        }
      ]
    }
  }
```

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"
}
```

# 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.<br><br>**Important Notices:**<br>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 |
|---|---|---|
| 200 | Successful operation | A successful API operation in terms of Authorization, Connectivity and valid JSON Message Structure.<br><br>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 downstream systems which return 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

Example


```
{
  "messageId": "89817674-daOO-4883",
  "returnCode": "200",
  "returnReason": "Successful operation",
```

```
  "sentTime": "2016-11-15T10:00:00.000Z",
  "responseTime": "2016-11-15T10:00:00.000Z"
}
```

## itemsObj: object

PROPERTIES

**product_name:** string range: (up to 200 chars) required

Product Item Name / Description

**product_id:** string range: (up to 50 chars) required

Product Number / ID

**unitAmt:** integer range: 100 ≤ x ≤ 999999999 required

Unit Amount of each item

> ! NOTICE: Do not use comma or dot. For example, value `1250000` means `12,500.00`

**unit:** integer range: 1 ≤ x ≤ 9999 required

No. of Unit

**vat:** integer range: 0 ≤ x ≤ 999999999 required

Total VAT Tax Amount for all units

> ! NOTICE: Do not use comma or dot. For example, value `1250000` means `12,500.00`

**subAmt:** integer range: 100 ≤ x ≤ 999999999 required

The Sum of one particular item with mulitple orders plus VAT. For example: `unitAmt x unit + vat = subAmt`

> ! NOTICE: Do not use comma or dot. For example, value `1250000` means `12,500.00`

Example

```
{
  "product_name": "Product Item 1",
  "product_id": "A",
  "unitAmt": 9000,
  "unit": 1,
```

```
  "vat": 1000,
  "subAmt": 10000
}
```

## 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

> ! **NOTICE:** The sequence of this field inside the `udfs` array object you define in the request message of one particular transaction will be maintained the same as it is returned in the response message of other APIs.

Example

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

## payLinkReqtModel: object

PROPERTIES

**transaction:** pay_rqt_txn_Obj required

**system:** pay_rqt_system_Obj required

**payment:** pay_rqt_payment_Obj required

**items:** Array< itemsObj > range: (up to 100 objects) required

Array of Product Descriptions in the basket

**udfs:** Array< udfsObj > range: (up to 50 objects) optional

Array of User Defined Fields

Example


```
{
  "transaction": {
    "txnRef": "PAY-QJZV956664",
    "tenant_id": "0001",
    "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
  },
  "system": {
    "redirectUrl": "https://www.example.com/redirect",
    "notificationUrl": "https://www.example.com/notification"
  },
  "payment": {
    "country": "JP",
    "amount": 10000,
    "description": "Payment Order of #PAY-QJZV956664"
  },
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 9000,
      "unit": 1,
      "vat": 1000,
      "subAmt": 10000
    }
  ],
  "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 range: (up to 100 chars) required

Unique ID referred to a specific transaction

- Merchant is required to generate a unique ID for each transaction in alphanumeric format, duplicated ID will be rejected.

**tenant_id:** object required
Tenant ID. Given by HSBC during Merchant Profile creation.

**plan_id:** string range: (up to 100 chars) optional
Input Corresponding Plan ID to associate a installment/recurring payment

Example


```
{
  "txnRef": "PAY-QJZV956664",
  "tenant_id": "0001",
  "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
}
```


## pay_rqt_system_Obj: object

PROPERTIES

**redirectUrl:** string range: (up to 500 chars) required
Define URL endpoint for redirecting back to merchant's site after payment

**notificationUrl:** string range: (up to 500 chars) required
Define URL endpoint for receiving status update notification (server-to-server) from HSBC after payment/refund request is completed.

Example


```
{
  "redirectUrl": "https://www.example.com/redirect",
  "notificationUrl": "https://www.example.com/notification"
}
```

# pay_rqt_payment_Obj: object

PROPERTIES

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

| Possible Value | Definition |
| --- | --- |
| JP | Japan |

**amount:** integer range: 100 ≤ x ≤ 999999999 required
Payment Amount

> ! NOTICE: Amount value must include 2 decimal places due to the system default setting for all currencies. Furthermore, do not use any comma or dot. For instance, value `150000` means `1,500.00` yen.

**description:** string range: (up to 200 chars) optional
Payment Description

Example

```
{
  "country": "JP",
  "amount": 10000,
  "description": "Payment Order of #PAY-QJZV956664"
}
```

# payLinkRespModel: 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": "PAY-QJZV956664"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "sysDatetime": "2020-01-01T13:00:00+09:00",
      "redirectLink": "<Encoded_Redirect_Submit_Form>"
    }
  }
}
```


# pay_rpn_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 100 chars) required

Returning back Transaction Reference

Example


```
{
  "txnRef": "PAY-QJZV956664"
}
```

# pay_rpn_system_Obj: object

PROPERTIES

**sysCode:** string range: (up to 6 chars) required

System Return Code

| Possible Value | Definition |
|---|---|
| 000000 | Request Successful |
| 800110 | Invalid Calculation Found in Product Sub-Amount |
| 800120 | Invalid Calculation Found in Order Total Amount |
| 900030 | Duplicate Transaction Reference |
| 999999 | System Error |

**sysMsg:** string range: (up to 128 chars) required

Corresponding Text Message of System Return Code

**sysDatetime:** string range: (up to 25 chars) required

Time of sending out this request / response

- Server system time. A `GMT+9` timezone information is appended to the end of the timestamp to indicate this time is a Japan local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

**redirectLink:** string range: (up to 5120 chars) optional

Encoded Redirect Link with all form submit parameters. Return only for successful request.

Example


```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful",
  "sysDatetime": "2020-01-01T13:00:00+09:00",
  "redirectLink": "<Encoded_Redirect_Submit_Form>"
}
```

# codeReqtModel: object

PROPERTIES

**transaction:** code_rqt_txn_Obj required

**system:** code_rqt_system_Obj required

**payment:** code_rqt_payment_Obj required

**items:** Array< itemsObj > required

Array of Product Descriptions in the basket

**udfs:** Array< udfsObj > optional

Array of User Defined Fields

Example


```
{
  "transaction": {
    "txnRef": "PAY-QJZV956664",
    "tenant_id": "0001"
  },
  "system": {
    "notificationUrl": "https://www.example.com/notification",
    "qr_str": "<QR_Code_String>"
  },
  "payment": {
    "country": "JP",
    "amount": 10000,
    "description": "Payment Order of #PAY-QJZV956664"
  },
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 9000,
      "unit": 1,
      "vat": 1000,
      "subAmt": 10000
    }
  ],
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoker"
    }
  ]
}
```

## code_rqt_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 100 chars) required
Unique ID referred to a specific transaction

- Merchant is required to generate a unique ID for each transaction in alphanumeric format, duplicated ID will be rejected.

**tenant_id:** string range: (up to 4 chars) required
Tenant ID. Given by HSBC during Merchant Profile creation.

Example

```
{
  "txnRef": "PAY-QJZV956664",
  "tenant_id": "0001"
}
```

## code_rqt_system_Obj: object

PROPERTIES

**notificationUrl:** string range: (up to 500 chars) required
Define URL endpoint for receiving status update notification (server-to-server) from HSBC after payment/refund request is completed.

**qr_str:** string range: (up to 128 chars) required
Decode the QR Code image into a string

Example

```
{
  "notificationUrl": "https://www.example.com/notification",
  "qr_str": "<QR_Code_String>"
}
```

# code_rqt_payment_Obj: object

PROPERTIES

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

| Possible Value | Definition |
| --- | --- |
| JP | Japan |

**amount:** integer range: 100 ≤ x ≤ 999999999 required
Payment Amount

> ! NOTICE: Amount value must include 2 decimal places due to the system default setting for all currencies. Furthermore, do not use any comma or dot. For instance, value `150000` means `1,500.00` yen.

**description:** string range: (up to 200 chars) optional
Payment Description

Example

```
{
  "country": "JP",
  "amount": 10000,
  "description": "Payment Order of #PAY-QJZV956664"
}
```

# codeRespModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**system:** code_rpn_system_Obj required

**transaction:** code_rpn_txn_Obj required

**payment:** code_rpn_pay_Obj optional

Related information of Payment Request (with Payment Gateway). Return only for successful payment request.

**code:** code_Obj optional

Related information of Code Transaction (with Code Companies). Return only for successful payment request

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": "PAY-QJZV956664",
      "tenant_id": "0001",
      "process_id": "ee5b902a153f104281f4b81c5ce8216b",
      "process_pass": "f1973eef815a6e1541b356ab06e2478c",
      "error_code": "BARCODE_ERROR",
      "error_msg": "正しいバーコードをスキャンしてください。"
    },
    "payment": {
      "id": "000014640567",
      "resp_code": "OK",
      "amount": 650000,
      "description": "Payment Order of #PAY-QJZV956664",
      "datetime": "2020-01-01T13:02:00+09:00"
    },
    "code": {
      "id": "000000002563",
      "type": "3",
      "status": "1",
      "currency": "JPY",
      "amount": 650000
    }
  }
}
```

# code_rpn_system_Obj: object

PROPERTIES

**sysCode:** string range: (up to 6 chars) required

System Return Code

| Possible Value | Definition |
|---|---|
| 000000 | Request Successful |
| 800110 | Invalid Calculation Found in Product Sub-Amount |
| 800120 | Invalid Calculation Found in Order Total Amount |
| 900030 | Duplicate Transaction Reference |
| 900000 | Transaction is Failed |
| 999999 | System Error |

**sysMsg:** string range: (up to 128 chars) required

Corresponding Text Message of System Return Code

Example


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


# code_rpn_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 100 chars) required

Returning Transaction Reference

**tenant_id:** string range: (up to 4 chars) required

Returning Tenant ID

**process_id:** string range: (up to 32 chars) optional

Returning Process ID for a successful request. For checking transactions in Merchant Portal.

**process_pass:** string range: (up to 32 chars) optional

Returning Process Password for a successful request. For checking transactions in Merchant Portal.

**error_code:** string range: (up to 32 chars) optional

Error Code. Return only if any issue happens

**error_msg:** string range: (up to 128 chars) optional

Error Message. Return only if any issue happens

Example


```
{
  "txnRef": "PAY-QJZV956664",
  "tenant_id": "0001",
  "process_id": "ee5b902a153f104281f4b81c5ce8216b",
  "process_pass": "f1973eef815a6e1541b356ab06e2478c",
  "error_code": "BARCODE_ERROR",
  "error_msg": "正しいバーコードをスキャンしてください。"
}
```


# code_rpn_pay_Obj: object

PROPERTIES

**id:** string range: (up to 12 chars) required

The identifier of the corresponding Code Payment Request made via the payment gateway.

**resp_code:** string range: (up to 5 chars) required

Respond Code of the corresponding Code Payment Request.

> ! **NOTICE:**
> Respond Code is an operational status of the request returned by the Payment Gateway. `OK` means the request is accepted and will be processed by Payment Gateway, other then `OK` means fail and please contact HSBC support.

**amount:** integer range: 100 ≤ x ≤ 999999999 required

Payment Amount

> ! NOTICE: Amount value must include 2 decimal places due to the system default setting for all currencies. Furthermore, do not use any comma or dot. For instance, value `150000` means `1,500.00` yen.

**description:** string range: (up to 200 chars) optional

Payment Description. Return if it has been defined in the request.

**datetime:** string range: (up to 25 chars) required

Returning Transaction time for the successful Code Payment Request

- Bank system local time. A `GMT+9` timezone information is appended to the end of the timestamp to indicate this time is a Japan local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

Example


```
{
  "id": "000014640567",
  "resp_code": "OK",
  "amount": 650000,
  "description": "Payment Order of #PAY-QJZV956664",
  "datetime": "2020-01-01T13:02:00+09:00"
}
```


# code_Obj: object

PROPERTIES

**id:** string range: (up to 32 chars) required

The identifier of the corresponding Code Transaction made via the Code Payment Gateway.

**type:** string range: (up to 2 chars) required

Types of Code

| Possible Value | Definition *(Code Brands / Companies)* | Refund Deadline *(counting from the day after the payment completion date)* |
|---|---|---|

| | | |
|---|---|---|
| 0 | WeChat Pay | 89 days |
| 1 | Alipay | 89 days |
| 3 | 楽天ペイ | 9 days |
| 5 | PayPay | 14 days |
| 6 | メルペイ | 365 days |
| 7 | d 払い | 90 days |
| 8 | LINE Pay | 30 days |
| 9 | au PAY | 90 days |
| E | J-Coin Pay | 365 days |
| Z | 銀聯 | 30 days |

**status:** string range: (up to 2 chars) required
Code Transaction Status

| Possible Value | Definition |
|---|---|
| 1 | Paying |
| 2 | Paid |
| 3 | Refunding |
| 4 | Refunded |
| 6 | Cancelled |
| 99 | System Error |

**currency:** string enum: [ JPY ] range: (up to 10 chars) required
Code Payment Currency

**amount:** integer range: 100 ≤ x ≤ 999999999 required
Code Payment Amount

> ! NOTICE: Amount value must include 2 decimal places due to the system default setting for all currencies. Furthermore, do not use any comma or dot. For instance, value `150000` means `1,500.00` yen.

Example


```
{
  "id": "000000002563",
  "type": "3",
  "status": "1",
  "currency": "JPY",
  "amount": 650000
}
```


## enquiryRespModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**system:** enq_rpn_sys_Obj required

**transaction:** enq_rpn_txn_Obj required

**payments:** Array< payment_rpn_Obj > optional

Return if the request is successful

**refunds:** Array< refund_rpn_Obj > optional

Return if refund has been requested to the corresponding payment

**code:** code_Obj optional

Return if it is a code payment

**links:** Array< halLinkObj > optional

Collection of related resources

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": "PAY-QJZV956664",
      "tenant_id": "0001",
      "process_id": "ee5b902a153f104281f4b81c5ce8216b",
      "process_pass": "f1973eef815a6e1541b356ab06e2478c",
      "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
    },
    "payments": [
      {
        "id": "000014640567",
        "resp_code": "OK",
        "approvalNo": "0003000",
        "amount": 100000,
        "description": "Payment Order of #PAY-QJZV956664"
      }
    ],
    "refunds": [
      {
        "id": "RFD-DFCV112233",
        "resp_code": "OK",
        "approvalNo": "0003000",
        "amount": 100000,
        "create_datetime": "2020-01-01T13:02:00+09:00"
      }
    ],
    "code": {
      "id": "000000002563",
      "type": "3",
      "status": "1",
      "currency": "JPY",
      "amount": 650000
    },
    "links": [
      {
        "href": "/plan/@id",
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "rel": "plan",
        "method": "GET"
      }
    ]
  }
}
```


# 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 |
| 900000 | Transaction is Failed |
| 999999 | System Error |

**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 100 chars) required
Returning Transaction Reference

**tenant_id:** string range: (up to 4 chars) optional
Returning Tenant ID for a successful request

**process_id:** string range: (up to 32 chars) optional
Returning Process ID for a successful request. For checking transactions in Merchant Portal.

**process_pass:** string range: (up to 32 chars) optional
Returning Process Password for a successful request. For checking transactions in Merchant Portal.

**plan_id:** string range: (up to 100 chars) optional
Returning Plan ID if the request associates a plan

Example

```
{
  "txnRef": "PAY-QJZV956664",
  "tenant_id": "0001",
  "process_id": "ee5b902a153f104281f4b81c5ce8216b",
  "process_pass": "f1973eef815a6e1541b356ab06e2478c",
  "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
}
```

# payment_rpn_Obj: object

PROPERTIES

**id:** string range: (up to 20 chars) required

The identifier of the corresponding Payment Request made via the payment gateway.

**resp_code:** string range: (up to 5 chars) required

Respond Code of the corresponding Payment Request.

> ! **NOTICE:**
> Respond Code is an operational status of the request returned by the Payment Gateway. `OK` means the request is accepted and will be processed by Payment Gateway, other then `OK` means fail and please contact HSBC support.

**approvalNo:** string range: (up to 7 chars) optional

Returning Transaction Approval Number, only for Credit Card Payment

**amount:** integer range: 100 ≤ x ≤ 999999999 required

Payment Amount

> ! NOTICE: Amount value must include 2 decimal places due to the system default setting for all currencies. Furthermore, do not use any comma or dot. For instance, value `150000` means `1,500.00` yen.

**description:** string range: (up to 200 chars) optional

Payment Description. Return if it has been defined in the request.

Example


```
{
  "id": "000014640567",
  "resp_code": "OK",
  "approvalNo": "0003000",
  "amount": 100000,
  "description": "Payment Order of #PAY-QJZV956664"
}
```


# refund_rpn_Obj: object

PROPERTIES

**id:** string range: (up to 100 chars) required
The identifier of the corresponding Refund Request made via the payment gateway.

**resp_code:** string range: (up to 5 chars) required
Respond Code of the corresponding Refund Request.

> ! **NOTICE:**
> Respond Code is an operational status of the request returned by the Payment Gateway. `OK` means the request is accepted and will be processed by Payment Gateway, other then `OK` means fail and please contact HSBC support.

**approvalNo:** string range: (up to 7 chars) optional
Returning Refund Approval Number, only for Credit Card Payment

**amount:** integer range: 100 ≤ x ≤ 999999999 required
Refund Amount

> ! NOTICE: Amount value must include 2 decimal places due to the system default setting for all currencies. Furthermore, do not use any comma or dot. For instance, value `150000` means `1,500.00` yen.

**create_datetime:** string range: (up to 25 chars) optional
Returning Transaction time for the successful Refund Request

- Bank system local time. A `GMT+9` timezone information is appended to the end of the timestamp to indicate this time is a Japan local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

Example


```
{
  "id": "RFD-DFCV112233",
  "resp_code": "OK",
  "approvalNo": "0003000",
  "amount": 100000,
  "create_datetime": "2020-01-01T13:02:00+09:00"
}
```


## refundReqtModel: object

PROPERTIES

**txnRef:** string range: (up to 100 chars) required

Merchant to pass the original Transaction Reference

**refund_id:** string range: (up to 100 chars) optional

Merchant can optionally assign an unique Refund Reference Number for every refund transaction. The number will then be returned in response message `"refund": {"id": ""}`, otherwise the `id` will be assigned by payment gateway.

Example


```
{
  "txnRef": "PAY-QJZV956664",
  "refund_id": "RFD-DFCV112233"
}
```


## 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_Obj required

**code:** code_Obj optional

Return if it is a code payment

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": "PAY-QJZV956664"
    },
    "refund": {
      "id": "RFD-DFCV112233",
      "resp_code": "OK",
      "approvalNo": "0003000",
      "amount": 100000,
      "create_datetime": "2020-01-01T13:02:00+09:00"
    },
    "code": {
      "id": "000000002563",
      "type": "3",
      "status": "1",
      "currency": "JPY",
      "amount": 650000
    }
  }
}
```


## refund_rpn_sys_Obj: object

PROPERTIES

**sysCode:** string range: (up to 6 chars) required

System Return Code

| Possible Value | Definition |
| --- | --- |
| 000000 | Request Successful |
| 900000 | Transaction is Failed |
| 900010 | Transaction Record Not Found |
| 900030 | 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 100 chars) required

Return Transaction Reference

Example


```
{
  "txnRef": "PAY-QJZV956664"
}
```

# statusRtnReqtModel: object

PROPERTIES

**transaction:** enq_rpn_txn_Obj required

**merchant:** merchant_Obj required

**payment:** payment_rpn_Obj required

**refund:** refund_rpn_Obj optional

Return if it is a refund request

**code:** code_Obj optional

Return if it is a code payment

**udfs:** Array< udfsObj > range: (up to 50 objects) optional

Array of User Defined Fields

Example


```
{
  "transaction": {
    "txnRef": "PAY-QJZV956664",
    "tenant_id": "0001",
    "process_id": "ee5b902a153f104281f4b81c5ce8216b",
    "process_pass": "f1973eef815a6e1541b356ab06e2478c",
    "plan_id": "PLN-123e4567-e89b-12d3-a456-426614174000"
  },
  "merchant": {
    "merId": "42298549900001"
  },
  "payment": {
    "id": "000014640567",
    "resp_code": "OK",
    "approvalNo": "0003000",
    "amount": 100000,
    "description": "Payment Order of #PAY-QJZV956664"
  },
  "refund": {
    "id": "RFD-DFCV112233",
    "resp_code": "OK",
    "approvalNo": "0003000",
    "amount": 100000,
    "create_datetime": "2020-01-01T13:02:00+09:00"
  },
  "code": {
    "id": "000000002563",
    "type": "3",
```

```
    "status": "1",
    "currency": "JPY",
    "amount": 650000
  },
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoker"
    }
  ]
}
```


## merchant_Obj: object

PROPERTIES

**merId:** string range: (up to 10 chars) required

Returning Merchant ID

Example


```
{
  "merId": "42298549900001"
}
```


## statusRtnRespModel: object

PROPERTIES

**status:** string range: (up to 30 chars) required

Return Message

Example


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


## createPlanReqtModel: object

PROPERTIES

**type:** string enum: [ R, I ] range: (up to 1 chars) required

Plan type

| Possible Value | Definition | Remark |
|---|---|---|
| R | Revolving | Once a Revolving Payment is initiated, the payment will keep rolling until a Cancel/Refund Operation is submitted. |
| I | Installment | The total number of installment must be defined. |

**description:** string range: (up to 100 chars) required

Description

**total_count:** integer range: 2 ≤ x ≤ 84 conditional

Installment Total Count. Required if `{"type": "I"}`

Example


```
{
  "type": "I",
  "description": "Monthly Installment Plan #1",
  "total_count": 12
}
```

# createPlanRespModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**system:** systemPostObj required

**plan:** planObj optional

Return if the request is successful

**links:** Array< halLinkObj > optional

Collection of related resources

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"
    },
    "plan": {
      "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
      "type": "I",
      "description": "Monthly Installment Plan #1",
      "total_count": 12,
      "create_date": "2020-01-01T13:02:00+09:00"
    },
    "links": [
      {
        "href": "/plan/@id",
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "rel": "self",
        "method": "GET"
      }
    ]
  }
}
```

# systemPostObj: object

PROPERTIES

**sysCode:** string range: (up to 6 chars) required

System Return Code

| Possible Value | Definition |
| --- | --- |
| 000000 | Request Successful |
| 900000 | Request Failed |
| 999999 | System Error |

**sysMsg:** string range: (up to 128 chars) required

Corresponding Text Message of System Return Code

Example


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


# systemGetObj: object

PROPERTIES

**sysCode:** string range: (up to 6 chars) required

System Return Code

| Possible Value | Definition |
| --- | --- |
| 000000 | Request Successful |
| 900000 | Request Failed |

| | |
|---|---|
| 900010 | Record Not Found |
| 999999 | System Error |

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

**no_of_record:** integer range: 1 ≤ x ≤ 999 required
Total No. of Record(s)

**no_of_page:** integer range: 1 ≤ x ≤ 999 required
Total No. of Page(s)

Example


```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful",
  "no_of_record": 99,
  "no_of_page": 1
}
```


## halLinkObj: object

PROPERTIES

**href:** string range: (up to 100 chars) required
Hypertext Application Language (HAL) - URL Endpoint of the related resource

**id:** string range: (up to 100 chars) required
Hypertext Application Language (HAL) - Entity ID of the related resource where it replaces the `@id` in the URI.

**rel:** string range: (up to 100 chars) required
Hypertext Application Language (HAL) - Related entity name

**method:** string range: (up to 100 chars) required
Hypertext Application Language (HAL) - HTTP Method of the related resource

Example

```
{
  "href": "XXXX",
  "id": "XXXX",
  "rel": "XXXX",
  "method": "XXXX"
}
```


# planObj: object

PROPERTIES

**id:** string range: (up to 100 chars) required

Plan ID

**type:** string enum: [ R, I ] range: (up to 1 chars) required

Plan type

| Possible Value | Definition | Remark |
|---|---|---|
| R | Revolving | Once a Revolving Payment is initiated, the payment will keep rolling until a Cancel/Refund Operation is submitted. |
| I | Installment | The total number of installment must be defined. |

**description:** string range: (up to 100 chars) required

Description

**total_count:** integer range: 2 ≤ x ≤ 84 conditional

Installment Total Count. Required if `{"type": "I"}`

**create_date:** string range: (up to 25 chars) required

Creation date of this Plan

Example


```
{
  "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
  "type": "I",
  "description": "Monthly Installment Plan #1",
  "total_count": 12,
  "create_date": "2020-01-01T13:02:00+09:00"
```

```
}
```


# getPlanRespModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**system:** systemGetObj required

**plans:** Array< planObj > optional

Array of all Plan(s) previously created

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",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "plans": [
      {
        "id": "PLN-123e4567-e89b-12d3-a456-426614174000",
        "type": "I",
        "description": "Monthly Installment Plan #1",
        "total_count": 12,
        "create_date": "2020-01-01T13:02:00+09:00"
      }
    ]
  }
}
```

# Lifecycle of Cryptographic Keys

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

1. Generate keys pair (Private Key and Public Key Certificate)
2. ***Optional:*** *Export CSR (Certificate Signing Request) and get signed with CA (Certificate Authority)*

> ! **DO 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. Key Maintenance
5. 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. Yet, 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.

> ! **NOTICE:**
> Certificate Specification:
> - Certificate type: `X.509 version 3`
> - Signing algorithm: `2048-bit RSA`
> - Hashing algorithm: `SHA-256`

```
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.

> ! **DO YOU KNOW?**
> Keystore is a password-protected repository of keys and certificates. 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 `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:
```

> ! **NOTICE:** Private Key password and Keystore password can be the same or Merchant can set them differently to be more secure.

2. **Optional:** Export CSR and get signed with CA. This step can be skipped if 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 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.

> ! **NOTICE:** `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 Certificate and send to HSBC for key exchange.

> ! **DO 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 maintain integrity of the public key. This 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.

> ! **NOTICE:** If Merchant associates the original keys pair `merchant_key_pair`, the exported Certificate is

> without CA-signed, and hence, Self-Signed. However, if 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.

> ! **NOTICE:** 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 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<br>• 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:<br>• **key password** - Do not save the password in plain text or hard-coded in application. Recommend to encrypt it by any Password Encryption Tools<br>• **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 | For a self-signed Certificate, the same condition has been mentioned as above.<br><br>However, the validity period of a CA-signed Certificate is depended on the purchase plan of the issuing CA. |

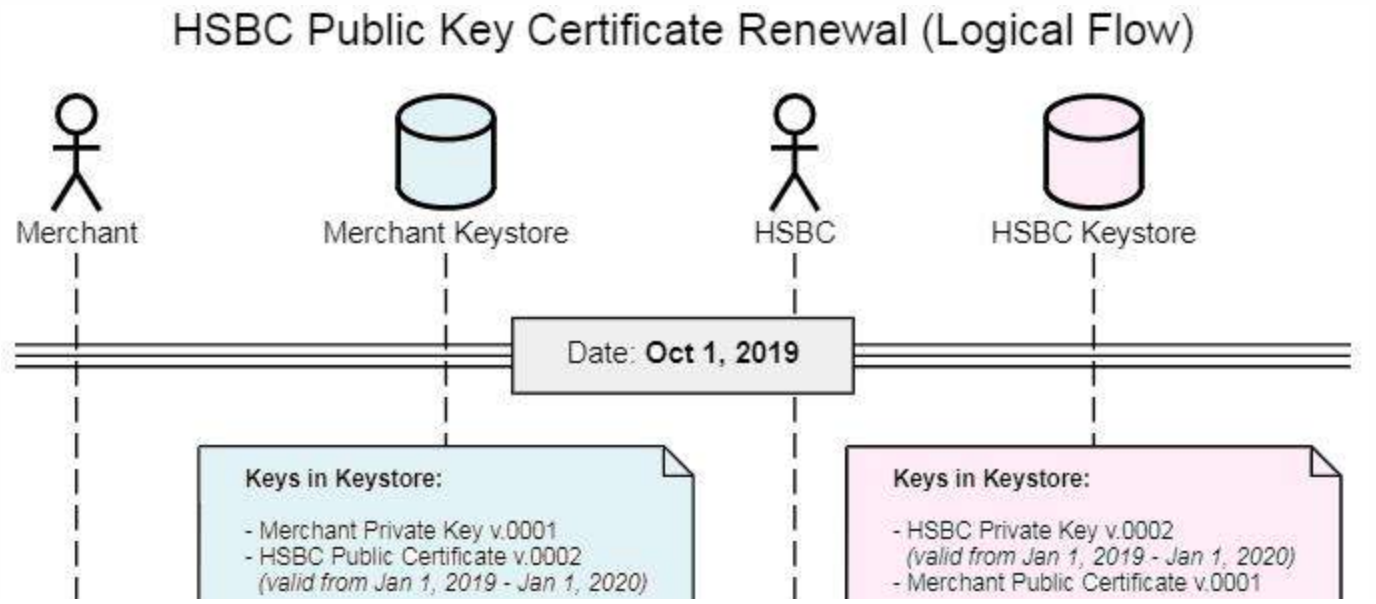
| | | |
|---|---|---|
| | one single key repository for a centralised key management. | The most common standard is 1 to 2 years. |
| HSBC's Public Key Certificate | Same as the above | 1 Year<br><br>**NOTICE:** Technically, the validity period is usually 1 Year plus 1 to 2 months more. The spare period 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. |

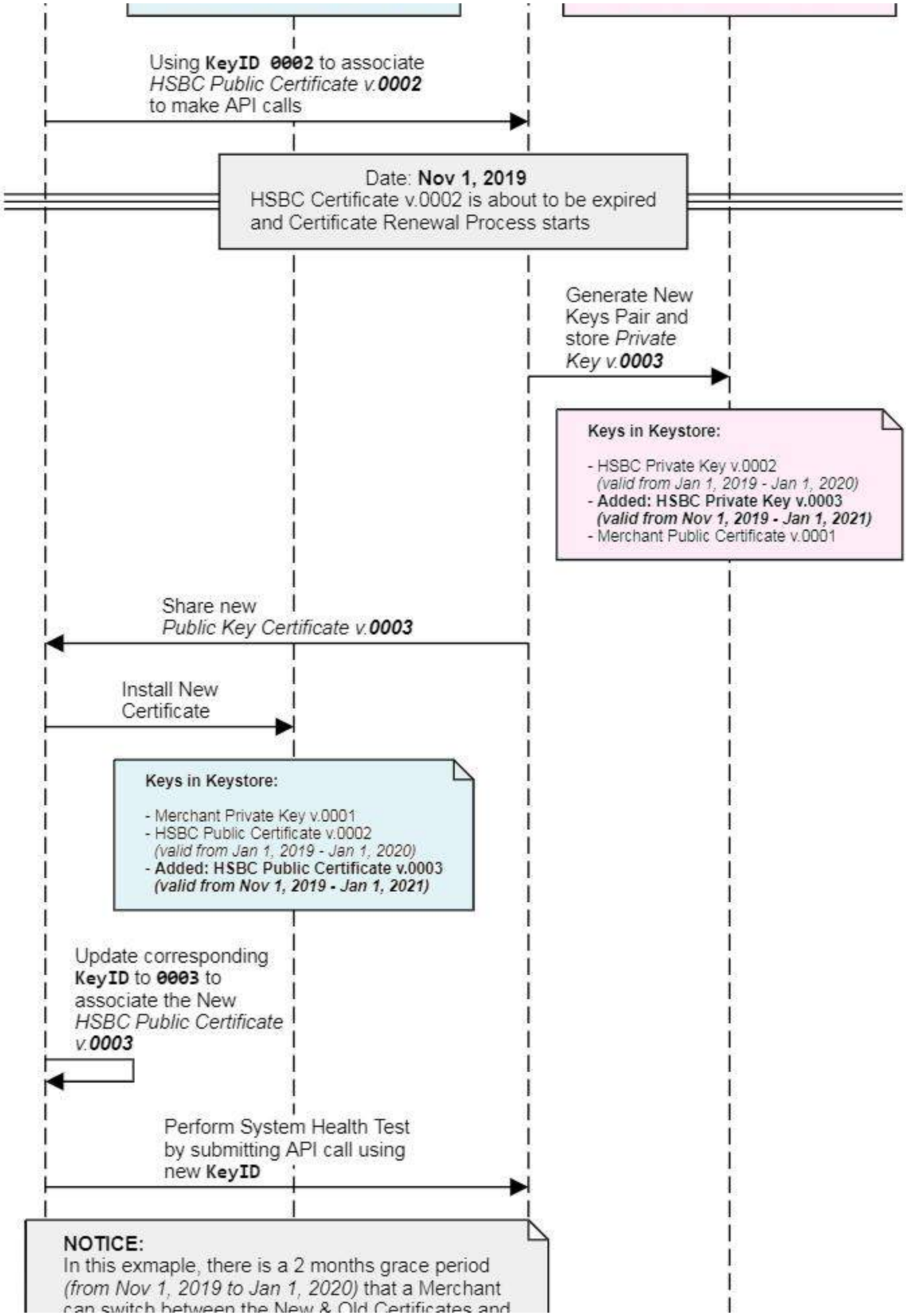
---

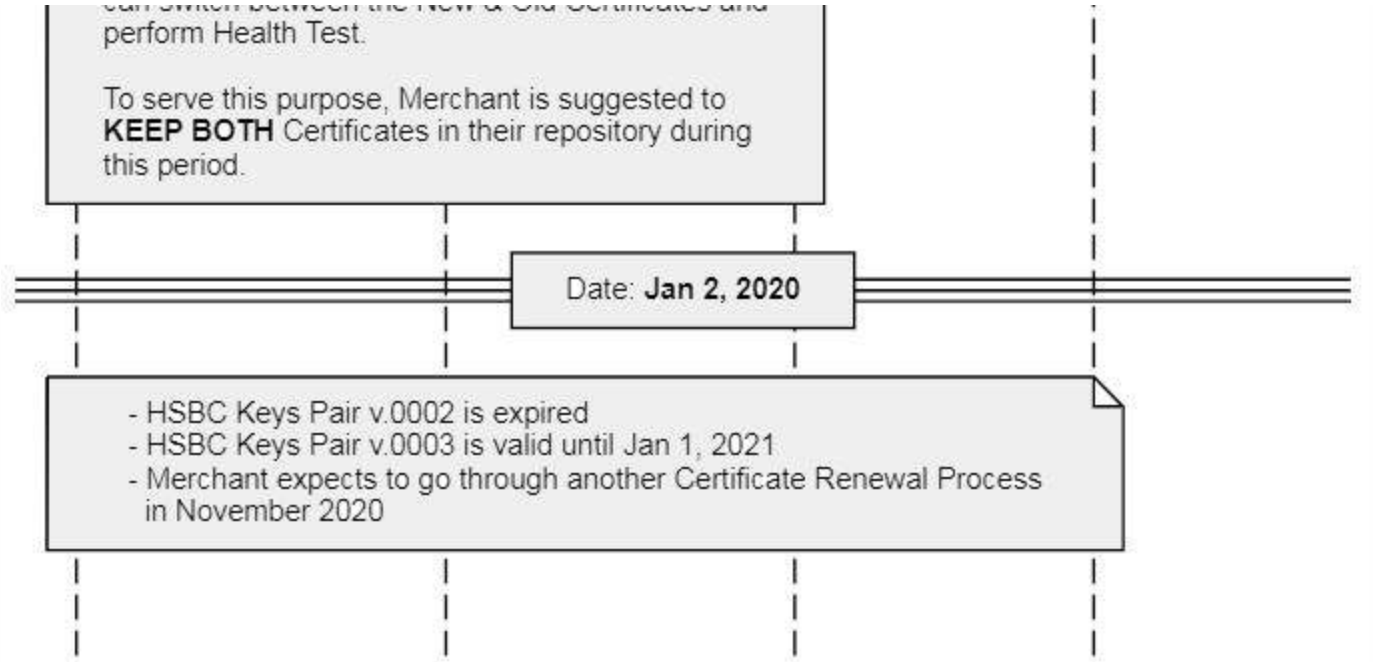
# Certificates and Keys Renewal

Every Public Key Certificate has an expiration date and when either Merchant's or HSBC's Certificate is about to expire, a key renewal process will be taken place. Please see the below Key Renewal Process Flow for your reference:

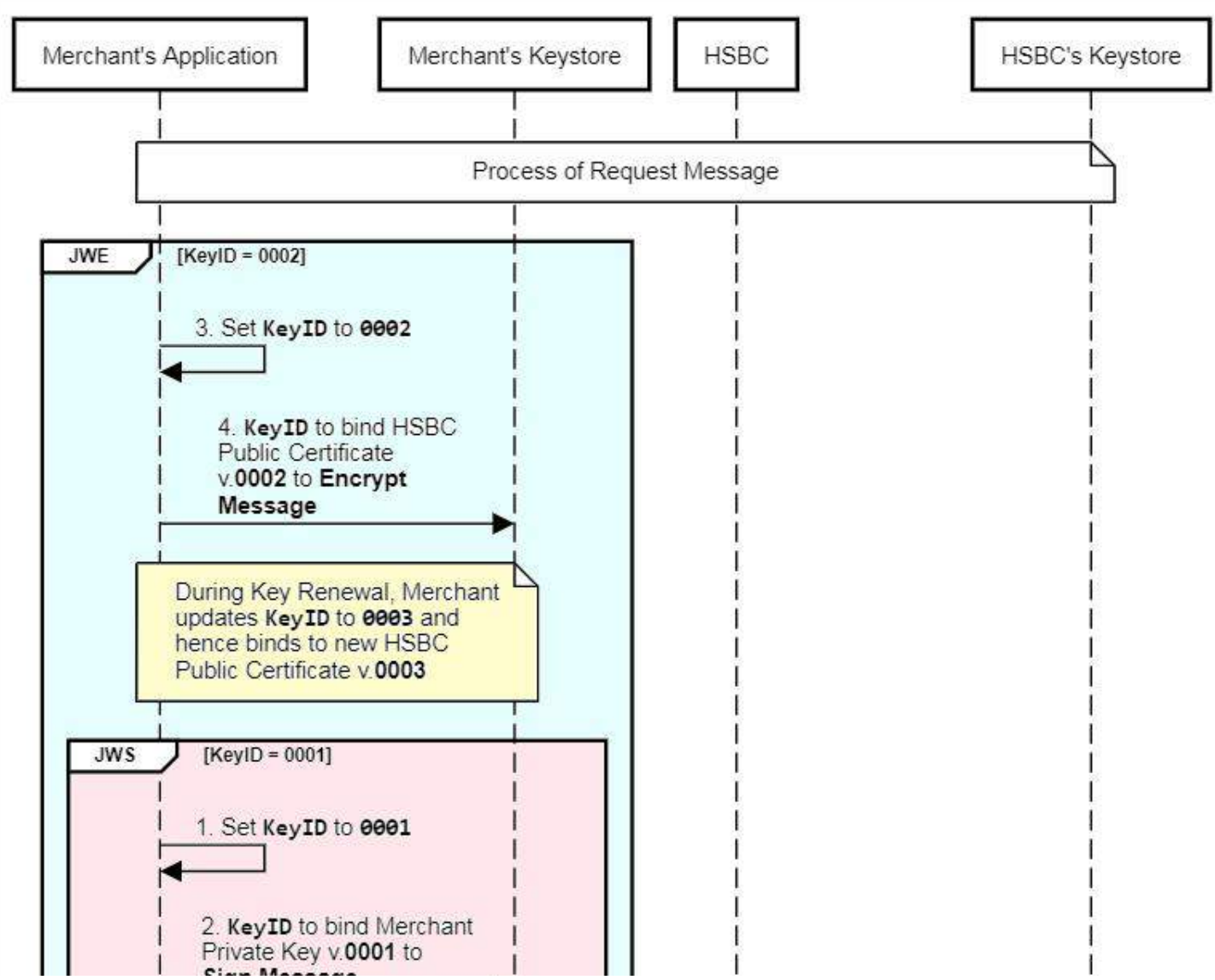
> ! **SOME RULES YOU SHOULD KNOW:**
>
> - **Keys Repository:** This is a make-up for demonstration purpose only.
> - **Keys Name:** Using a `Key Name` `KeyID` naming convention is for a simpler demonstration. The suggested identifier of one key should be the alias name inside a key repository.
> - **KeyID Value:** HSBC uses naming convention `0001`, `0002`, `0003` ... `n + 1`, when every time HSBC certificate is renewed, the `KeyID` value will be `n + 1`.
> - **KeyID Binding:** The binding between `KeyID` and corresponding `Keys Pair` in merchant's system can make use of any key/value logic, such as 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 make-up for demonstration purpose only.

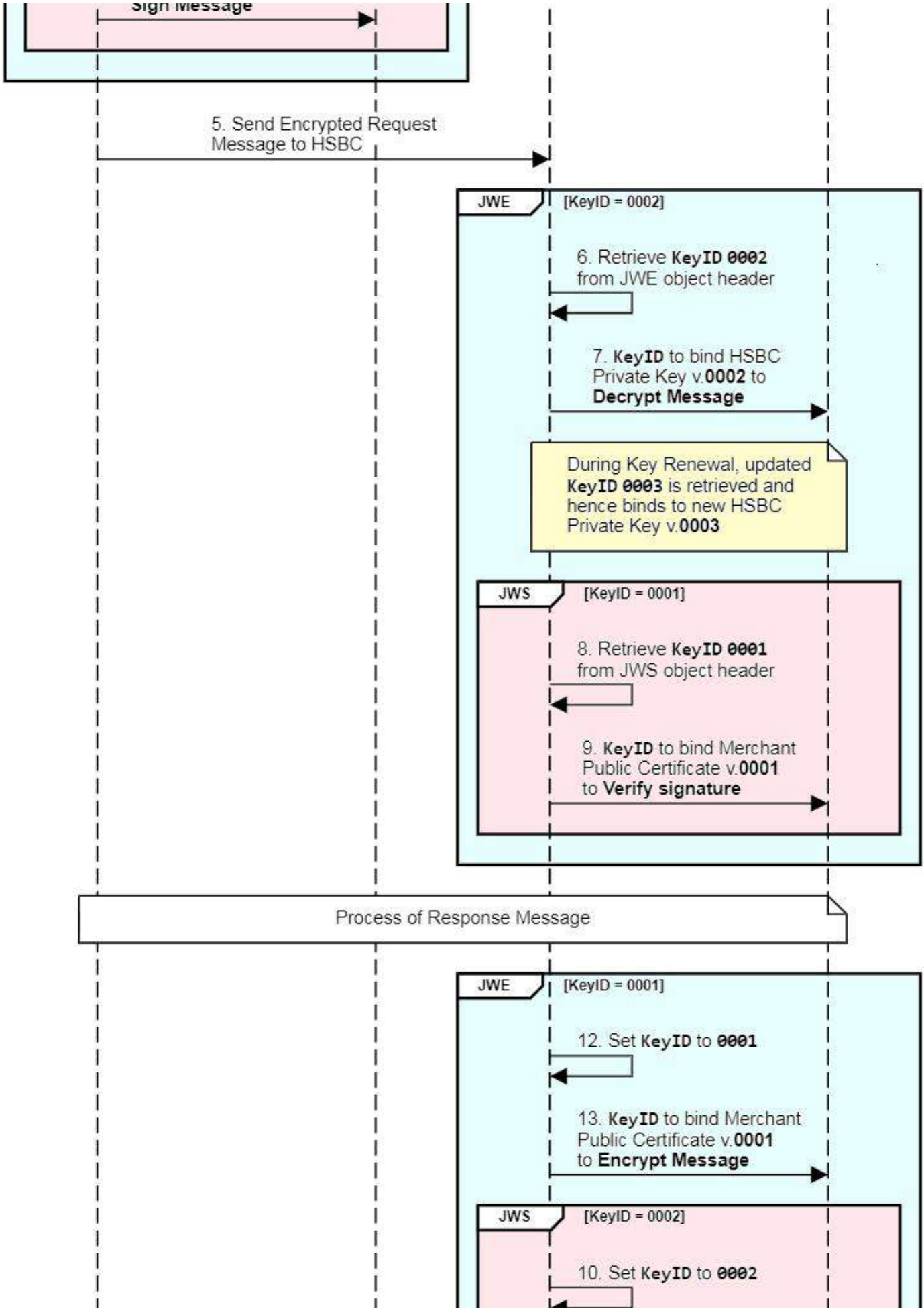


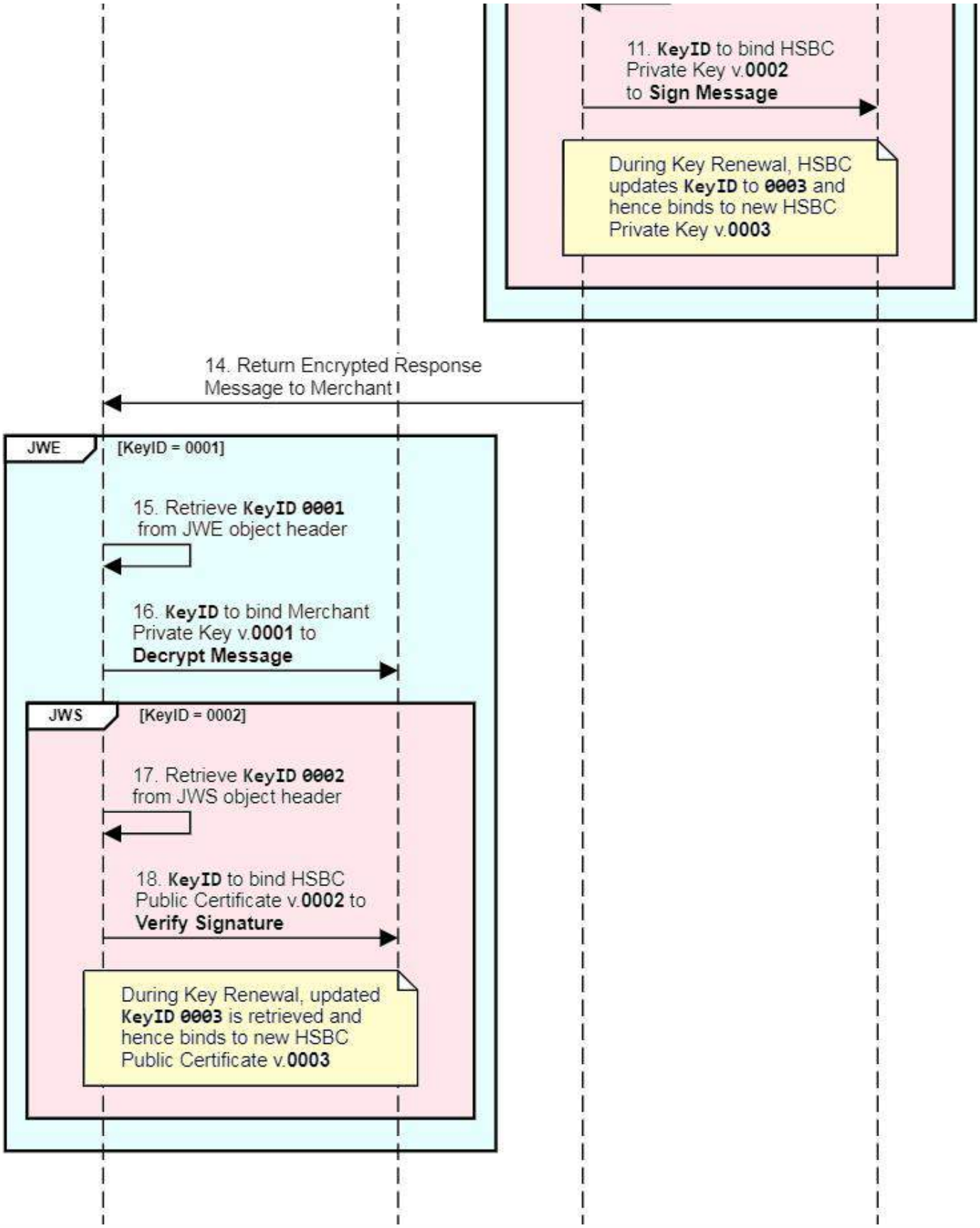

Using KeyID 0002 to associate HSBC Public Certificate v.0002 to make API calls
Date: Nov 1, 2019
HSBC Certificate v.0002 is about to be expired and Certificate Renewal Process starts
Generate New Keys Pair and store Private Key v.0003
Keys in Keystore:
- HSBC Private Key v.0002 (valid from Jan 1, 2019 - Jan 1, 2020)
- Added: HSBC Private Key v.0003 (valid from Nov 1, 2019 - Jan 1, 2021)
- Merchant Public Certificate v.0001
Share new Public Key Certificate v.0003
Install New Certificate
Keys in Keystore:
- Merchant Private Key v.0001
- HSBC Public Certificate v.0002 (valid from Jan 1, 2019 - Jan 1, 2020)
- Added: HSBC Public Certificate v.0003 (valid from Nov 1, 2019 - Jan 1, 2021)
Update corresponding KeyID to 0003 to associate the New HSBC Public Certificate v.0003
Perform System Health Test by submitting API call using new KeyID
NOTICE:
In this exmaple, there is a 2 months grace period (from Nov 1, 2019 to Jan 1, 2020) that a Merchant




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




Sign Message
5. Send Encrypted Request Message to HSBC
JWE
[KeyID = 0002]
6. Retrieve KeyID 0002 from JWE object header
7. KeyID to bind HSBC Private Key v.0002 to Decrypt Message
During Key Renewal, updated KeyID 0003 is retrieved and hence binds to new HSBC Private Key v.0003
JWS
[KeyID = 0001]
8. Retrieve KeyID 0001 from JWS object header
9. KeyID to bind Merchant Public Certificate v.0001 to Verify signature
Process of Response Message
JWE
[KeyID = 0001]
12. Set KeyID to 0001
13. KeyID to bind Merchant Public Certificate v.0001 to Encrypt Message
JWS
[KeyID = 0002]
10. Set KeyID to 0002




! **NOTICE:** All examples above are about the Certificate Renewal of HSBC, whenever Merchant wants to renew their Certificate, please switch your role and steps into HSBC's.

# Download Swagger

Click here to download Swagger 2.0 file in YAML format.

# Disclaimer

***IMPORTANT NOTICE***

This document is issued by The Hongkong and Shanghai Banking Corporation Limited, Hong Kong ("HSBC"). HSBC does not warrant that the contents of this document are accurate, sufficient or relevant for the recipient's purposes and HSBC gives no undertaking and is under no obligation to provide the recipient with access to any additional information or to update all or any part of the contents of this document or to correct any inaccuracies in it which may become apparent. Receipt of this document in whole or in part shall not constitute an offer, invitation or inducement to contract. The recipient is solely responsible for making its own independent appraisal of the products, services and other content referred to in this document. This document should be read in its entirety and should not be photocopied, reproduced, distributed or disclosed in whole or in part to any other person without the prior written consent of the relevant HSBC group member. 