API Specification for Malaysia DUITNOW

Version: 2.1

Description

This document introduces the **OpenAPI specification** which describes the REST APIs for HSBCs Collection of digital payments - Malaysia DuitNow QR Code.

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

Update Log

- [Jan 11, 2022] **v2.1** Revised several content sections
- [Jun 18, 2021] **v2.0** Added new Content, API Operation and Schema for Supporting Dynamic QR Payment
- [Jan 08, 2020] **v1.2** Added new API Payment Simulation API
- [Sep 20, 2019] **v1.1**
  - Updated `Disclaimer`
  - Enhanced Section `GETTING STARTED`
  - Added Content Section `REFERENCE`
- [Aug 16, 2019] **v1.0** Initial Version

How to Read this Document

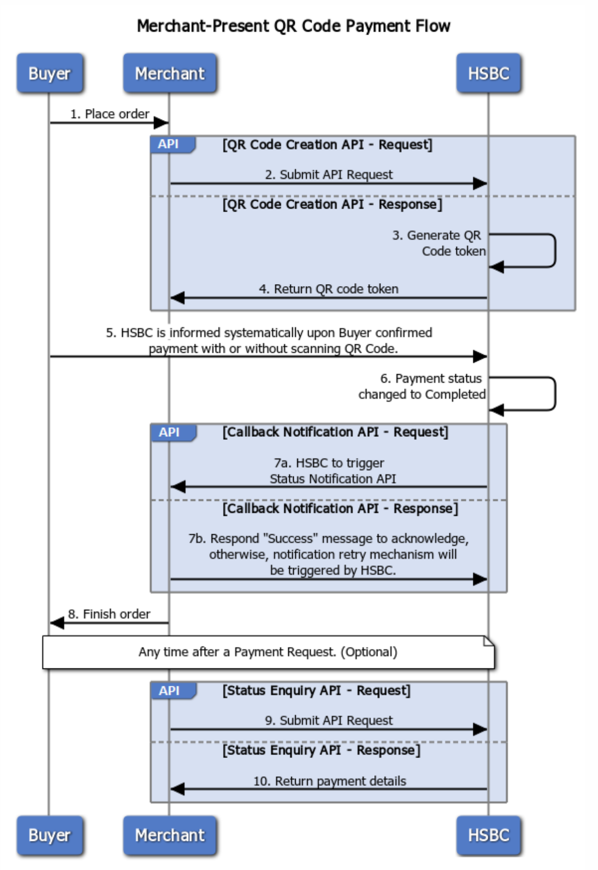
This document walks through the API listing the key functions by section: API Usage Flow, API Connectivity, and API Operation. There is also a FAQ and a list of Schema Definitions used by API operations.

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

Use Cases for this API

Make a Payment using a Dynamic QR Code

The standard API flow for a Merchant using a dynamic QR code on an Online Web Store or a POS terminal, is illustrated below:


Merchant-Present QR Code Payment Flow
Buyer
Merchant
HSBC
1. Place order
API
[QR Code Creation API - Request]
2. Submit API Request
[QR Code Creation API - Response]
3. Generate QR Code token
4. Return QR code token
5. HSBC is informed systematically upon Buyer confirmed payment with or without scanning QR Code.
6. Payment status changed to Completed
API
[Callback Notification API - Request]
7a. HSBC to trigger Status Notification API
[Callback Notification API - Response]
7b. Respond "Success" message to acknowledge, otherwise, notification retry mechanism will be triggered by HSBC.
8. Finish order
Any time after a Payment Request. (Optional)
API
[Status Enquiry API - Request]
9. Submit API Request
[Status Enquiry API - Response]
10. Return payment details
Buyer
Merchant
HSBC

1. The Buyer places an order.
2. The Merchant submits a Create QR code request.
3. The HSBC backend system generates a QR Code token or image.

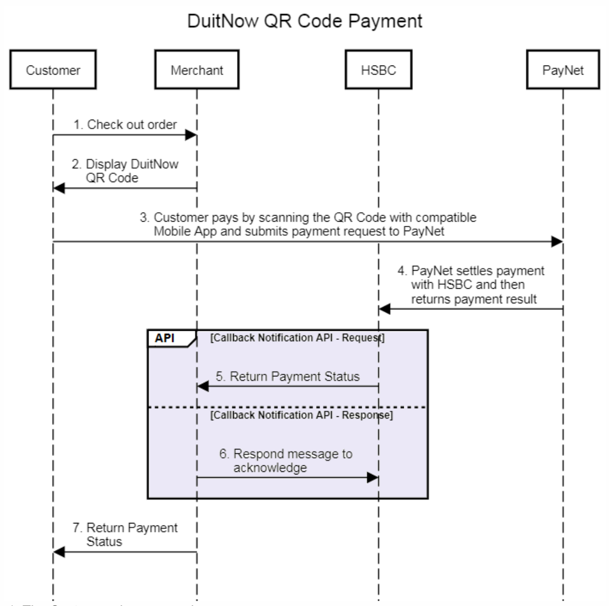
4. HSBC returns the QR Code token or image via the API response.
5. The Merchant displays the QR Code Image on their online store or POS terminal. HSBC receives an acknowledgement as soon a the Buyer confirms payment after scanning the QR code.

> **NOTE:**
> For testing purpose, the Merchant can simulate a payment at this stage.

6. Payment is completed.
7. The acknowledge message is sent back via a Callback Payment Notification. The payment status is changed to `Completed` at the HSBC backend system.
8. The Merchant notifies a completed order to the buyer.
9. To check the payment status, the Merchant submits an Order Status Enquiry any time after a payment request, or if no acknowledge message is returned after a certain period of time.
10. HSBC returns the payment status via the API response.

Make a Payment using a Static QR Code

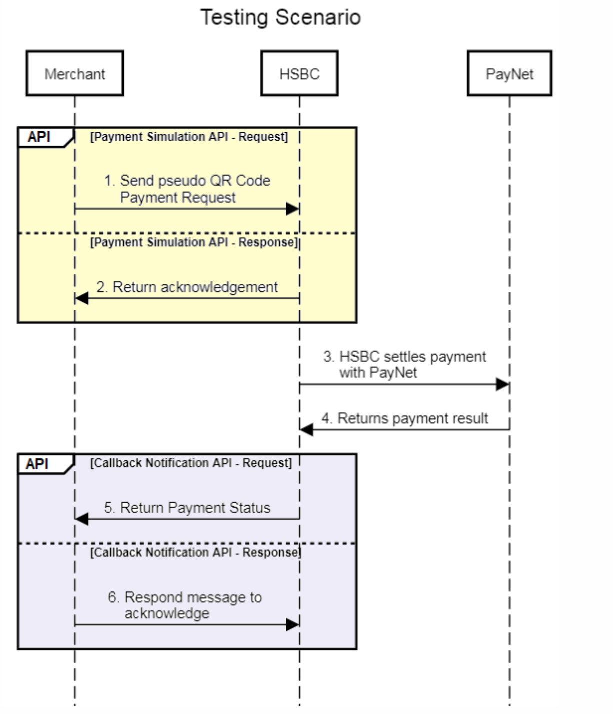
Payment status is returned to the Merchant via an asynchronous callback notification as soon as the payment request is settled. After the HSBC payment platform completes reconciliation with PayNet and receives payment results, HSBC will push the result back to the Merchant by calling this API.


DuitNow QR Code Payment
Customer
Merchant
HSBC
PayNet
1. Check out order
2. Display DuitNow QR Code
3. Customer pays by scanning the QR Code with compatible Mobile App and submits payment request to PayNet
4. PayNet settles payment with HSBC and then returns payment result
API
[Callback Notification API - Request]
5. Return Payment Status
[Callback Notification API - Response]
6. Respond message to acknowledge
7. Return Payment Status

1. The Customer places an order.
2. The Merchant displays a DuitNow QR Code on their website (e-Commerce model), or on a POS device (in-store model).
3. The customer pays by scanning the QR Code with any compatible Mobile App. Payment requests are submitted to PayNet for further settlement.
4. The Payment is settled and completed. PayNet returns the payment status to HSBC.
5. As soon as the payment status is received from PayNet, HSBC sends back the Payment Status via a QR Payment Status Notification API.
6. The Merchant responds to the message to acknowledge successful message delivery. Otherwise, notification will be resent.
7. The Merchant notifies the completed order to the customer.

Simulate a Payment in the Sandbox Environment

HSBC offers a Payment Simulation API to simulate the process in which the buyer confirms the payment by scanning a QR Code. This API is only available in the Sandbox Environment.



Testing Scenario
Merchant
HSBC
PayNet
API
[Payment Simulation API - Request]
1. Send pseudo QR Code Payment Request
[Payment Simulation API - Response]
2. Return acknowledgement
3. HSBC settles payment with PayNet
4. Returns payment result
API
[Callback Notification API - Request]
5. Return Payment Status
[Callback Notification API - Response]
6. Respond message to acknowledge

How to Connect

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

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

API Gateway URL

API Gateway URL must be included before each API endpoint to make API calls.

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

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

Sandbox (Simulation API Only)
https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcasp-paysim-ea-merchantservices-cert-proxy/v1

API Authentication

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

Client ID & Client Secret	
Purpose	API Gateway locates the corresponding policy of the specific API consumer

Client ID & Client Secret		
Components	• Client ID	• Client Secret
Where to get it?	Delivered by HSBC via secure email during onboarding procedure	
Implementation	In HTTP header: `x-hsbc-client-id: [Client ID]`	In HTTP header: `x-hsbc-client-secret: [Client Secret]`

User Identification

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

Connection Security

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

Message Security - Data Encryption and Signing

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

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

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

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

> ! **NOTE:**
> Technically, an X.509 certificate can serve as a SSL Certificate as well as a Public Key Certificate for Data Encryption. However, for segregation of certificate usage, HSBC recommends that the Merchant uses a different X.509 Certificate for Data Encryption.

Moreover, the Public Key Certificate does not have to be CA-signed. However, if the Merchant decides to enhance security, a CA-Signed Certificate is acceptable.

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

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

How to Sign and Encrypt Outgoing Message

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

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

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

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

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

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

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

    return jwsObject;
}
```

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

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

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

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

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

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

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

    return jweObject;
}
```

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

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

How to Decrypt Message and Verify Signature of an Incoming Message

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

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

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

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

#2  PrivateKey privateKey = (PrivateKey) keyStore.getPrivateKey("merchant_priva

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

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

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

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

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

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

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

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

Summary

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

How to Make an API Request

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

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

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

Make Your API Request with Plain Messages

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

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

Step 1. Run this command on your platform:

POST GET

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

1. Submit the `POST` request to the API URL endpoint.
2. Set the secret header `message_encrypt: false` to indicate this API request is without message encryption. This header is only applicable in Sandbox environment.
3. Put the Basic Authorization in HTTP header `Authorization`.

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

```
curl -X GET "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobile
 -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"
```

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

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

Making API Request with Message Encryption

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

POST GET

```
curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobil
 -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 "eyJraWQiOiIwMDAxIiwiZW5jIjoiQTEyOEdDTSIsImFsZyI6IlJTQS1PQUVQLTI1NiJ
```

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

```
curl -X GET "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobile
 -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"
```

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

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

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

Data Type Overview

Data Type Control:

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

Data Type	Allowed Characters	Definition & Important Notice
String *(For critical field)*	`0-9` `a-z` `A-Z` `-` `_` `.`	Critical field is used to be either a key or search criteria in HSBC backend system and hence tight restriction is applied to the allowed characters. Moreover, the starting and ending space of the string value will be trimmed before stored in HSBC system. For example, string `" example 12 34 "` will be trimmed to `"example 12 34"`. **List of Critical Fields:** All `id` (s)
Integer	`0-9`	Instead of having Max Length check for String, integer range will be checked, e.g. `0 ≤ x ≤ 9999`

Field Mandatory Control:

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

Time Zone Control:

Aspect	Format	Definition & Important Notice
In Request Message	`yyyy-MM-dd'T'HH:mm:ssZ`	Time zone is expected to be `GMT+8` (Malaysia standard time). Merchant is required to perform any necessary time zone conversion before submit request if needed.
In Response Message	`yyyy-MM-dd'T'HH:mm:ss±hh:mm`	Timezone returned in `api_gw` object is generated from HSBC API Gateway which located in Cloud and hence is calculated in `GMT+0`. On the other hand, time field in `response` object will be returned together with timezone information. For more details, please read each field definition carefully.

FAQ

SSL Connection Questions

Where can I find the HSBC SSL server certificates?

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

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

Message Encryption Questions

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

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

Javascript Object Signing and Encryption (JOSE) Framework Questions

Where can I get more information about JOSE Framework?

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

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

Where can I download JOSE libraries for development?

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

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

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

Orders

Create an Order and link them to payments

Orders

Create an Order by creating QR code

POST `/orders/{id}/createQR`

DESCRIPTION

This endpoint creates a QR Code *token or an image*. Once this API request is submitted by a merchant, HSBC will return QR Code token/image to the Merchant based on DuitNow specification. The Merchant then displays the QR Code to Buyer, for payment initiation process.

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
id: string range: (up to 25 chars) required in path	*A unique reference number defined by Merchant* *Duplicate ID will be rejected.*

REQUEST BODY

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

RESPONSES

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

Request Content-Types: application/json

Request Example

```
{
  "txnChannel": "01",
  "payMethod": [
    "DUITNOW"
  ],
  "qrOption": "02",
  "notificationUrl": "https://merchant.com/returnStatus",
  "expiryTime": "2021-06-11T14:10:25+08:00",
  "city": "MY",
  "merchantDescription": "Merchant Description",
  "fee": {
    "option": "02",
    "amount": 75000,
    "percentage": 1050
  }
  "posMachineId": "00112233-4455-6677-8899-aabbccddeeff",
  "employeeId": "00112233-4455-6677-8899-xxyyzzxxyyzz",
  "order": {
    "amount": 500000,
    "currency": "MYR",
    "goodsDescription": "Goods Description",
    "storeLabel": "987654321",
    "billNumber": "987654321",
    "rrn": "987654321",
    "rrn2": "987654321"
  }
}
```

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"
    },
    "order": {
      "id": "1234567890123456789012345",
      "txnRef": "1234567890123456789012345",
      "amount": 500000,
      "currency": "MYR",
      "qrCode": "QR_CODE_DATA"
    }
  }
}
```

Response Example (400 Bad Request)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "Return Reason Message here",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  }
}
```

Orders

Retrieve a particular Order by ID

GET `/orders/{id}`

DESCRIPTION

This endpoint returns the latest Order status based on the Order ID provided by the Merchant.

One Order can contain multiple Payments. `payments` object will be returned only if there is any prior payment transaction occurred to that particular Order.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-client-id required	[Client ID]

in header	
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
id: string range: (up to 25 chars) required in path	*A unique reference number defined by Merchant*

RESPONSES

200 OK getOrderRespModel	Successful operation. *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*
400 Bad Request exceptionModel	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",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "order": {
      "id": "1234567890123456789012345",
      "txnRef": "1234567890123456789012345",
      "amount": 500000,
      "currency": "MYR",
      "goodsDescription": "Goods Description",
      "referenceLabel": "987654321987654321",
      "storeLabel": "987654321",
      "terminalLabel": "987654321",
      "consumerLabel": "987654321",
      "loyaltyNumber": "987654321",
      "billNumber": "987654321",
      "rrn": "987654321",
      "rrn2": "987654321",
      "mobileNumber": "987654321",
      "purposeOfTransaction": "987654321",
      "additionalConsumerDataRequest": "XXX",
      "geoCoordinates": "13.2904027,108.4265113",
      "suppInfo": "Supplementary Info",
      "payments": [
        {
          "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
          "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
          "bankTxnTime": "2018-06-11T14:10:25+07:00",
          "notifyTime": "2018-06-11T14:10:25+07:00",
          "amount": 500000,
          "currency": "MYR"
        }
      ]
    }
  }
}
```

Response Example (400 Bad Request)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "Return Reason Message here",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  }
}
```

Payments

One Order can have multiple Payment transactions

Payments

Retrieve a particular Payment Transaction by ID

GET `/payments/{id}`

DESCRIPTION

This endpoint returns the latest Payment status based on the Payment ID provided by the Merchant.

Payment always belong to one Order. The corresponding Order details can be found in `links` object.

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
id: string range: (up to 35 chars) required in path	*Payment ID, a unique reference number returned by system*

RESPONSES

200 OK getPaymentRespModel	Successful operation. *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*
400 Bad Request exceptionModel	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",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "payment": {
      "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
      "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
      "bankTxnTime": "2018-06-11T14:10:25+07:00",
      "notifyTime": "2018-06-11T14:10:25+07:00",
      "amount": 500000,
      "currency": "MYR"
    },
    "links": [
      {
        "href": "/orders/@id",
        "id": "1234567890123456789012345",
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

Response Example (400 Bad Request)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "Return Reason Message here",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  }
}
```

Transactions

The resource Transaction is the polymorphic form of different transaction type such as Payment (Transaction), Refund (Transaction) or Subscripted Recurring (Transaction), etc.

> ! **NOTICE:**
> Only Payment Transaction is being supported for the time being.

Transactions

Retrieve a particular Transaction by ID

GET `/transactions/{id}`

DESCRIPTION

This endpoint returns the latest status of a Transaction based on the type of the ID provided by the Merchant.

Related resource(s) will be presented in `links` objects.

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
id: string range: (up to 35 chars) required in path	*A Payment ID*

RESPONSES

Response Content-Types: application/json

200 OK getPaymentRespModel for Payment Transaction	Successful operation. *Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*
400 Bad Request exceptionModel	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 Example (200 OK) for Payment Transaction

```
{
  "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",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "payment": {
      "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
      "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
      "bankTxnTime": "2018-06-11T14:10:25+07:00",
      "notifyTime": "2018-06-11T14:10:25+07:00",
      "amount": 500000,
      "currency": "MYR"
    "links": [
      {
        "href": "/orders/@id",
        "id": "12345678901234567890123456",
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

Response Example (400 Bad Request)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "Return Reason Message here",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  }
}
```

Transactions

Callback Payment Notification

POST `/<Callback URL predefined by Merchant>`

DESCRIPTION

Payment status will be returned to Merchant by asynchronous callback notification once a payment request is settled. After HSBC payment platform completes reconciliation with PayNet and receives payment result, HSBC will push the result back to Merchant by calling this API.

> ! **Implementation** Mobile Collection 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 3 retries will be triggered in every 3 - 5 minutes. Maximum 4 calls including the 1st attempt.

> ! **Endpoint Definition** URL endpoint is defined in field `notificationUrl` during QR code creation.

> ! **Exception Handling** Only successful case will be returned. Merchant can check Order Status if found no acknowledge message returned after a certain period of time.

REQUEST PARAMETERS

Content-Type: string required in header	text/plain

REQUEST BODY

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

Request Content-Types: text/plain

Request Example

```
{
  "system": {
    "proCode": "000000",
    "proMsg": "Transaction Successful"
  },
  "transaction": {
    "txnRef": "12345678901234567890123456",
    "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
    "bankTxnTime": "2018-06-11T14:10:25+07:00",
    "notifyTime": "2018-06-11T14:10:25+07:00",
    "referenceLabel": "987654321987654321"
  },
  "merchant": {
    "merId": "12345QWERT",
    "storeLabel": "987654321",
    "terminalLabel": "987654321"
  },
  "customer": {
    "consumerLabel": "987654321",
    "loyaltyNumber": "987654321"
  },
  "order": {
    "amount": 500000,
    "currency": "MYR"
  },
  "bill": {
    "billNumber": "987654321",
    "rrn": "987654321",
    "rrn2": "987654321"
  },
  "supplementary": {
    "mobileNumber": "987654321",
    "purposeOfTransaction": "987654321",
    "additionalConsumerDataRequest": "XXX",
    "geoCoordinates": "13.2904027,108.4265113",
    "suppInfo": "Supplementary Info"
  }
}
```

RESPONSES

200 OK	Successful operation.

Response Content-Types: application/json

Response Example (200 OK)

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

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

Simulation

Contains resource collections for Simulating a Payment.

Simulation

Payment Simulation API

POST `/my/payment/simulation`

**Please be reminded a different BASE URL is used by Simulation APIs.*

DESCRIPTION

This endpoint is used to simulate the process the buyer confirms payment with scanning the QR Code. It is only available in Sandbox Environment for testing purpose. Encryption of both request and response message is bypassed.

REQUEST PARAMETERS

Content-Type required in header string	application/json
message_encrypt required in header string	false

REQUEST BODY

paySimRequestModel	Message Encryption is not required for this message payload

Request Content-Types: application/json

Request Example

```
{
  "system": {
    "is_notification_encrypted": "Y"
  },
  "transaction": {
    "txnRef": "12345678901234567890l2345",
    "referenceLabel": "987654321987654321",
    "txnChannel": "01"
  },
  "merchant": {
    "merId": "12345QWERT",
    "storeLabel": "987654321",
    "terminalLabel": "987654321"
  },
  "customer": {
    "consumerLabel": "987654321",
    "loyaltyNumber": "987654321"
  },
  "order": {
    "amount": 500000,
    "currency": "MYR"
  },
  "bill": {
    "billNumber": "987654321",
    "rrn": "987654321",
    "rrn2": "987654321"
  },
  "supplementary": {
    "mobileNumber": "987654321",
    "purposeOfTransaction": "987654321",
    "additionalConsumerDataRequest": "XXX",
    "geoCoordinates": "13.2904027,108.4265113",
    "suppInfo": "Supplementary Info"
  }
}
```

RESPONSES

200 OK paySimResponseModel	Successful operation.
400 Bad Request exceptionModel	Bad Request.
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": {
    "txnRef": "12345678901234567890l2345",
    "proCode": "000000",
    "proMsg": "Payment Success"
  }
}
```

Response Example (400 Bad Request)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "Return Reason Message here",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  }
}
```

Schema Definitions

exceptionModel: object

PROPERTIES

api_gw: commonRespObj required

Example

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "Return Reason Message here",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  }
}
```

createQRReqtModel: object

PROPERTIES

txnChannel: string enum: [01, 02] range: (up to 2 chars) required
Transaction Channel

Possible Value	Definition
01	e-Commerce / m-Commerce
02	POS

payMethod: string[] required
Payment Method

Possible Value	Definition
DUITNOW	Malaysia DuitNow QR Code

qrOption: string enum: [01, 02] range: (up to 2 chars) required
QR Code Option. Data type returned in field `qrCode` from response message.

Possible Value	Definition
01	Return Raw QR Data
02	Return Base64 encoded QR image in PNG format

notificationUrl: string range: (up to 128 chars) required
URL provided by Merchant for returning status used by Payment Status Notification API

expiryTime: string range: (up to 25 chars) optional
QR code Expiry Time

- Client system time. The timezone is expected to be `GMT+8` (Malaysia local time). Merchant is required to perform timezone conversion if needed.

city: string enum: [MY] range: (up to 2 chars) optional
Country Code (Format: ISO 3166-1 alpha-2)

Possible Value	Definition
MY	Malaysia

merchantDescription: string range: (up to 20 chars) optional
Merchant Description

fee: object optional

PROPERTIES

option: string enum: [01, 02, 03] range: (up to 2 chars) required
Fee Option

Possible Value	Definition
01	Indicates that consumer should be prompted to enter Tip
02	Indicates that the merchant would mandatorily charge a flat convenience fee
03	Indicates that merchant would charge a percentage convenience fee

amount: integer range: 1 ≤ x ≤ 999999999999 conditional
Fee Amount. Conditional field, required when `option = "02"`

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

percentage: integer range: 1 ≤ x ≤ 9999 conditional
Fee Percentage. Conditional field, required when `option = "03"`

- Format: Eliminate punctuation and sign, support 2 decimal places e.g. `2134 = 21.34%`
- Example: `6 = 0.06%` indicate that convenience fee percentage is 0.06% and must be calculate as transaction amount * 0.06%

posMachineId: string range: (up to 36 chars) conditional
Unique ID of a POS device.

- Conditional field. Required when `txnChannel = "02"`

employeeId: string range: (up to 36 chars) conditional
ID of a staff member who handles a specific POS transaction.

- Conditional field. Required when `txnChannel = "02"`

order: object required

PROPERTIES

amount: integer range: 1 ≤ x ≤ 9999999999999 required
Order Amount

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

currency: string enum: [MYR] range: (up to 3 chars) required
Order Currency

Possible Value	Definition
MYR	Malaysian Ringgit

Example

```
{
  "txnChannel": "01",
  "payMethod": [
    "DUITNOW"
  ],
  "qrOption": "02",
  "notificationUrl": "https://merchant.com/returnStatus",
  "expiryTime": "2021-06-11T14:10:25+08:00",
  "city": "MY",
  "merchantDescription": "Merchant Description",
  "fee": {
    "option": "02",
    "amount": 75000,
    "percentage": 1050
  }
  "posMachineId": "00112233-4455-6677-8899-aabbccddeeff",
  "employeeId": "00112233-4455-6677-8899-xxyyzzxxyyzz",
  "order": {
    "amount": 500000,
    "currency": "MYR",
    "goodsDescription": "Goods Description",
    "storeLabel": "987654321",
    "billNumber": "987654321",
    "rrn": "987654321",
    "rrn2": "987654321"
  }
}
```

goodsDescription: string range: (up to 1280 chars) optional
Goods Description

storeLabel: string range: (up to 25 chars) optional
Store Label

billNumber: string range: (up to 25 chars) optional
Bill Number

rrn: string range: (up to 20 chars) optional
Recipient Reference Number (Ref-1)

rrn2: string range: (up to 30 chars) optional
Recipient Reference Number (Ref-2)

createQRRespModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemPostObj required

order: object optional

Return if it is a successful request

PROPERTIES

id: string range: (up to 25 chars) required
Unique Entity ID of Order, identical to `txnRef`

txnRef: string range: (up to 25 chars) required
Transaction Reference defined by Merchant

amount: integer range: 1 ≤ x ≤ 9999999999999 required
Order Amount

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

currency: string enum: [MYR] range: (up to 3 chars) required
Order Currency

Possible Value	Definition
MYR	Malaysian Ringgit

qrCode: string range: (up to 20000 chars) required
QR Code Data

- QR Code Data will only be returned if it is a successful transaction.
- Merchant can choose to have QR Code image (Base64 encoded) returned for PayNow. The image size is around 10-15k bytes. The no. of encoded output characters versus input bytes is approximately 4 / 3 (33% overhead)

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "order": {
      "id": "1234567890123456789012345",
      "txnRef": "1234567890123456789012345",
      "amount": 500000,
      "currency": "MYR",
      "qrCode": "QR_CODE_DATA"
    }
  }
}
```

getOrderRespModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemGetObj required

order: orderObj optional

Return if it is a successful request

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "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
    },
    "order": {
      "id": "1234567890123456789012345",
      "txnRef": "1234567890123456789012345",
      "amount": 500000,
      "currency": "MYR",
      "goodsDescription": "Goods Description",
      "referenceLabel": "987654321987654321",
      "storeLabel": "987654321",
      "terminalLabel": "987654321",
      "consumerLabel": "987654321",
      "loyaltyNumber": "987654321",
      "billNumber": "987654321",
      "rrn": "987654321",
      "rrn2": "987654321",
      "mobileNumber": "987654321",
      "purposeOfTransaction": "987654321",
      "additionalConsumerDataRequest": "XXX",
      "geoCoordinates": "13.2904027,108.4265113",
      "suppInfo": "Supplementary Info",
      "payments": [
        {
          "id": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
          "bankTxnId": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
          "bankTxnTime": "2018-06-11T14:10:25+07:00",
          "notifyTime": "2018-06-11T14:0:25+07:00",
          "amount": 500000,
          "currency": "MYR"
        }
      ]
    }
  }
}
```

getPaymentRespModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemGetObj required

payment: paymentObj optional

Return if it is a successful request

links: Array< halLinkObj > optional

Collection of related resources

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "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
    },
    "payment": {
      "id": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
      "bankTxnId": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
      "bankTxnTime": "2018-06-11T14:10:25+07:00",
      "notifyTime": "2018-06-11T14:10:25+07:00",
      "amount": 500000,
      "currency": "MYR"
    "links": [
      {
        "href": "/orders/@id",
        "id": "1234567890123456789012345",
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

getRefundRespModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemGetObj required

refund: refundObj optional

Return if it is a successful request

links: Array< halLinkObj > optional

Collection of related resources

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "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
    },
    "refund": {
      "id": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
      "bankTxnId": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
      "bankTxnTime": "2018-06-11T14:10:25+07:00",
      "amount": 500000,
      "currency": "MYR",
      "status": "SUCCESS"
    },
    "links": [
      {
        "href": "/payments/@id",
        "id": "1234567890123456789012345",
        "rel": "payment",
        "method": "GET"
      },
      {
        "href": "/orders/@id",
        "id": "1234567890123456789012345",
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

createRefundReqtModel:

PROPERTIES

amount: integer range: 1 ≤ x ≤ 9999999999999 required

Refund Amount

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

notificationUrl: string range: (up to 128 chars) required

URL provided by Merchant for returning status used by Refund Status Notification API

Example

```
{
  "amount": 500000,
  "notificationUrl": "https://merchant.com/returnStatus"
}
```

createRefundRespModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemPostObj required

refund: refundObj optional

Return if it is a successful request

links: Array< halLinkObj > optional

Collection of related resources

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "refund": {
      "id": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
      "bankTxnId": "YYYYMMDDBBBBBBBB030OCCSSSSSSSS",
      "bankTxnTime": "2018-06-11T14:10:25+07:00",
      "amount": 500000,
      "currency": "MYR",
      "status": "SUCCESS"
    },
    "links": [
      {
        "href": "/payments/@id",
        "id": "12345678901234567890l2345",
        "rel": "payment",
        "method": "GET"
      },
      {
        "href": "/orders/@id",
        "id": "12345678901234567890l2345",
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

callbackRfdReqtModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemPostObj required

merchant: object required

PROPERTIES

merchantId: string range: (up to 30 chars) required
Merchant ID

refund: refundObj required

links: Array< halLinkObj > required
Collection of related resources

Example

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "merchant": {
      "merchantId": "12345QWERT"
    },
    "refund": {
      "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
      "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
      "bankTxnTime": "2018-06-11T14:10:25+07:00",
      "amount": 500000,
      "currency": "MYR",
      "status": "SUCCESS"
    },
    "links": [
      {
        "href": "/payments/@id",
        "id": "12345678901234567890l2345",
        "rel": "payment",
        "method": "GET"
      },
      {
        "href": "/orders/@id",
        "id": "12345678901234567890l2345",
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

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

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

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

Corr. Return Code	Return Message Sample	Definition
200	Successful operation	A successful API operation in terms of Authorization, Connectivity and valid JSON Message Structure. Any checking failure on Business Logic level will be still considered a successful API operation yet the Business Logic checking result will be returned in `response` object.
400	Client ID - Merchant ID mapping is not correct/updated!	The binding of Client ID, Merchant ID and Merchant Public Certificate is incorrect or not up-to-date.
400	object has missing required properties `field name`	Fail to pass JSON Field Mandatory Check.
400	instance type `data type` does not match any allowed primitive type	Fail to pass JSON Field Type Check.

Example

```
{
  "messageId": "89817674-da00-4883",
  "returnCode": "200",
  "returnReason": "Successful operation",
  "sentTime": "2016-11-15T10:00:00.000Z",
  "responseTime": "2016-11-15T10:00:00.000Z"
}
```

Corr. Return Code	Return Message Sample	Definition
400	string `field value` is too long	Fail to pass JSON Field Max Length Check
400	instance failed to match at least one required schema among `no. of conditional field`	Fail to pass JSON Conditional Field Check.
500	java.net.ConnectException: Connection refused: connect	**Notices:** Message can be varied depended on the dependent system *(which across the entire system pipeline)* which returns this message. Yet, all reasons can be concluded into Internal Error or System Unavailable.

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

- This is a system time of HSBC API gateway which located in Cloud, timezone is calculated in `GMT+0`

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

- This is a system time of HSBC API gateway which located in Cloud, timezone is calculated in `GMT+0`

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
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
HAL Standard - URL Link

id: string range: (up to 100 chars) optional
HAL Standard - Enitiy ID

rel: string range: (up to 100 chars) required
HAL Standard - Relation of the Resource

method: string range: (up to 100 chars) required
HAL Standard - HTTP Method

Example

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

orderObj: object

PROPERTIES

id: string range: (up to 25 chars) required
Unique Entity ID of Order, identical to `txnRef`

txnRef: string range: (up to 25 chars) required

Example

```
{
  "id": "1234567890123456789012345",
  "txnRef": "1234567890123456789012345",
  "amount": 500000,
```

Transaction Reference defined by Merchant, also considered as the End-to-End ID of all other related transaction

amount: integer range: 1 ≤ x ≤ 9999999999999 required
Order Amount

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

currency: string enum: [MYR] range: (up to 3 chars) required
Order Currency

Possible Value	Definition
MYR	Malaysian Ringgit

goodsDescription: string range: (up to 1280 chars) optional
Goods Description

referenceLabel: string range: (up to 25 chars) optional
Returning Reference Label

storeLabel: string range: (up to 25 chars) optional
Store Label

terminalLabel: string range: (up to 25 chars) optional
Terminal Label

consumerLabel: string range: (up to 25 chars) optional
Consumer Label

loyaltyNumber: string range: (up to 25 chars) optional
Loyalty Number

billNumber: string range: (up to 25 chars) optional
Bill Number

rrn: string range: (up to 20 chars) optional
Recipient Reference Number (Ref-1)

rrn2: string range: (up to 30 chars) optional
Recipient Reference Number (Ref-2)

mobileNumber: string range: (up to 25 chars) optional
Mobile Number

purposeOfTransaction: string range: (up to 25 chars) optional
Purpose Of Transaction

additionalConsumerDataRequest: string range: (up to 3 chars) optional
Addtional Consumer Data Request

geoCoordinates: string range: (up to 25 chars) optional
Geo Coordinates

suppInfo: string range: (up to 140 chars) optional
Supplementary Info

payments: Array< paymentObj > optional
Array of all payments, presents if any payment transaction is processed regarding to the order

```
    "currency": "MYR",
    "goodsDescription": "Goods Description",
    "referenceLabel": "987654321987654321",
    "storeLabel": "987654321",
    "terminalLabel": "987654321",
    "consumerLabel": "987654321",
    "loyaltyNumber": "987654321",
    "billNumber": "987654321",
    "rrn": "987654321",
    "rrn2": "987654321",
    "mobileNumber": "987654321",
    "purposeOfTransaction": "987654321quot;,
    "additionalConsumerDataRequest": "XXX",
    "geoCoordinates": "13.2904027,108.4265113",
    "suppInfo": "Supplementary Info",
    "payments": [
      {
        "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
        "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
        "bankTxnTime": "2018-06-11T14:10:25+07:00",
        "notifyTime": "2018-06-11T14:10:25+07:00",
        "amount": 500000,
        "currency": "MYR"
      }
    ]
  }
```

paymentObj: object

PROPERTIES

id: string range: (up to 35 chars) required
Unique Entity ID of Payment, identical to `bankTxnId`

bankTxnId: string range: (up to 35 chars) required
Bank Transaction ID of the particular payment transaction, returned by system

bankTxnTime: string range: (up to 25 chars) required
Returning HSBC Transaction time for the inward credit payment

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

notifyTime: string range: (up to 25 chars) required
Returning Notification Time

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

amount: integer range: 1 ≤ x ≤ 9999999999999 required
Payment Amount

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

currency: string enum: [MYR] range: (up to 3 chars) required
Payment Currency

Possible Value	Definition
MYR	Malaysian Ringgit

Example

```
{
  "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
  "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
  "bankTxnTime": "2018-06-11T14:10:25+07:00",
  "notifyTime": "2018-06-11T14:10:25+07:00",
  "amount": 500000,
  "currency": "MYR"
}
```

refundObj: object

PROPERTIES

id: string range: (up to 35 chars) required
Unique Rntity ID of Refund, identical to `bankTxnId`

bankTxnId: string range: (up to 35 chars) required
Bank Transaction ID of the particular refund transaction, returned by system

bankTxnTime: string range: (up to 25 chars) required
Returning HSBC Transaction time for the refund transaction

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

Example

```
{
  "id": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
  "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
  "bankTxnTime": "2018-06-11T14:10:25+07:00",
  "amount": 500000,
  "currency": "MYR",
  "status": "SUCCESS"
}
```

amount: integer range: 1 ≤ x ≤ 9999999999999 required
Refund Amount

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

currency: string enum: [MYR] range: (up to 3 chars) required
Refund Currency

Possible Value	Definition
MYR	Malaysian Ringgit

status: string range: (up to 123 chars) required
Status of the Refund Transaction

callbackPtyReqtModel: object

PROPERTIES

system: notif_rqt_system_Obj required
transaction: notif_rqt_txn_Obj required
merchant: notif_rqt_merchant_Obj required
customer: notif_rqt_customer_Obj optional
order: notif_rqt_order_Obj required
bill: notif_rqt_bill_Obj optional
supplementary: notif_rqt_supp_Obj optional

Example

```
{
  "system": {
    "proCode": "000000",
    "proMsg": "Transaction Successful"
  },
  "transaction": {
    "txnRef": "1234567890123456789012345",
    "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
    "bankTxnTime": "2018-06-11T14:10:25+08:00",
    "notifyTime": "2018-06-11T14:10:25+08:00",
    "referenceLabel": "987654321987654321"
  },
  "merchant": {
    "merId": "12345QWERT",
    "storeLabel": "987654321",
    "terminalLabel": "987654321"
  },
  "customer": {
    "consumerLabel": "987654321",
    "loyaltyNumber": "987654321"
  },
  "order": {
    "amount": 500000,
    "currency": "MYR"
  },
  "bill": {
    "billNumber": "987654321",
    "rrn": "987654321",
    "rrn2": "987654321"
  },
  "supplementary": {
    "mobileNumber": "987654321",
    "purposeOfTransaction": "987654321",
    "additionalConsumerDataRequest": "XXX",
    "geoCoordinates": "13.2904027,108.4265113",
    "suppInfo": "Supplementary Info"
  }
}
```

notif_rqt_system_Obj: object

PROPERTIES

proCode: string range: (up to 6 chars) required
Return Process Code

Possible Value	Definition
000000	Transaction Successful

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

Example

```
{
  "proCode": "000000",
  "proMsg": "Transaction Successful"
}
```

notif_rqt_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 25 chars) optional
Transaction Reference defined by Merchant

bankTxnId: string range: (up to 35 chars) required
End-to-End Transaction ID

bankTxnTime: string range: (up to 25 chars) required
HSBC Transaction time for the inward credit payment

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

notifyTime: string range: (up to 25 chars) required
Notification Time

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

referenceLabel: string range: (up to 25 chars) optional
Reference Label

Example

```
{
  "txnRef": "1234567890123456789012345",
  "bankTxnId": "YYYYMMDDBBBBBBBB0300CCSSSSSSSS",
  "bankTxnTime": "2018-06-11T14:10:25+08:00",
  "notifyTime": "2018-06-11T14:10:25+08:00",
  "referenceLabel": "987654321987654321"
}
```

notif_rqt_merchant_Obj: object

Example

PROPERTIES

merId: string range: (up to 15 chars) required
Merchant ID

storeLabel: string range: (up to 25 chars) optional
Store Label

terminalLabel: string range: (up to 25 chars) optional
Terminal Label

```
{
  "merId": "12345QWERT",
  "storeLabel": "987654321",
  "terminalLabel": "987654321"
}
```

notif_rqt_customer_Obj: object

PROPERTIES

consumerLabel: string range: (up to 25 chars) optional
Consumer Label

loyaltyNumber: string range: (up to 25 chars) optional
Loyalty Number

Example

```
{
  "consumerLabel": "987654321",
  "loyaltyNumber": "987654321"
}
```

notif_rqt_order_Obj: object

PROPERTIES

amount: integer range: 1 ≤ x ≤ 999999999999 required
Payment Amount

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

Possible Value	Definition
MYR	Malaysian Ringgit

Example

```
{
  "amount": 500000,
  "currency": "MYR"
}
```

notif_rqt_bill_Obj: object

PROPERTIES

billNumber: string range: (up to 25 chars) optional
Bill Number

rrn: string range: (up to 20 chars) optional
Recipient Reference Number (Ref-1)

rrn2: string range: (up to 30 chars) optional
Recipient Reference Number (Ref-2)

Example

```
{
  "billNumber": "987654321",
  "rrn": "987654321",
  "rrn2": "987654321"
}
```

notif_rqt_supp_Obj: object

PROPERTIES

mobileNumber: string range: (up to 25 chars) optional
Mobile Number

purposeOfTransaction: string range: (up to 25 chars) optional
Purpose Of Transaction

additionalConsumerDataRequest: string range: (up to 3 chars) optional
Addtional Consumer Data Request

geoCoordinates: string range: (up to 25 chars) optional
Geo Coordinates

suppInfo: string range: (up to 140 chars) optional
Supplementary Info

Example

```
{
  "mobileNumber": "987654321",
  "purposeOfTransaction": "987654321",
  "additionalConsumerDataRequest": "XXX",
  "geoCoordinates": "13.2904027,108.4265113",
  "suppInfo": "Supplementary Info"
}
```

callbackRespModel: object

PROPERTIES

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

Example

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

paySimRequestModel: object

PROPERTIES

system: pay_sim_rqt_system_Obj required
transaction: pay_sim_rqt_txn_Obj required
merchant: notif_rqt_merchant_Obj required
customer: notif_rqt_customer_Obj optional
order: notif_rqt_order_Obj required
bill: notif_rqt_bill_Obj optional
supplementary: notif_rqt_supp_Obj optional

Example

```
{
  "system": {
    "is_notification_encrypted": "Y"
  },
  "transaction": {
    "txnRef": "12345678901234567890123456",
    "referenceLabel": "987654321987654321",
    "txnChannel": "01"
  },
  "merchant": {
    "merId": "12345QWERT",
      "storeLabel": "987654321",
      "terminalLabel": "987654321"
    },
    "customer": {
      "consumerLabel": "987654321",
      "loyaltyNumber": "987654321"
    },
    "order": {
      "amount": 500000,
      "currency": "MYR"
    },
    "bill": {
      "billNumber": "987654321",
      "rrn": "987654321",
      "rrn2": "987654321"
    },
    "supplementary": {
      "mobileNumber": "987654321",
      "purposeOfTransaction": "987654321",
      "additionalConsumerDataRequest": "XXX",
      "geoCoordinates": "13.2904027,108.4265113",
      "suppInfo": "Supplementary Info"
    }
  }
```

pay_sim_rqt_system_Obj: object

PROPERTIES

is_notification_encrypted: string enum: [Y, N] range: (up to 1 chars) required
Flag to indicate if the Status Notification message is encrypted or not

Example

```
{
  "is_notification_encrypted": "Y"
}
```

pay_sim_rqt_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 25 chars) required
Transaction Reference defined by Merchant

referenceLabel: string range: (up to 25 chars) optional
Reference Label

txnChannel: string enum: [01, 02] range: (up to 2 chars) required
Transaction Channel

Possible Value	Definition
01	e-Commerce / m-Commerce
02	POS

Example

```
{
  "txnRef": "1234567890123456789012345",
  "referenceLabel": "987654321987654321",
  "txnChannel": "01"
}
```

paySimResponseModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required

PROPERTIES

txnRef: string range: (up to 25 chars) required
Transaction Reference No. provided by merchant

proCode: string range: (up to 6 chars) required
Process Return Code

Possible Value	Definition
000000	Payment Success
900030	Duplicate Transaction Reference

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

Example

```
{
  "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": {
    "txnRef": "1234567890123456789012345",
    "proCode": "000000",
    "proMsg": "Payment Success"
  }
}
```

Lifecycle of Cryptographic Keys

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

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

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

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

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

Key Generation and Certificate Exchange with HSBC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

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

<Other Information>

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

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

<Other Information>

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

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

<Other Information>

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

Certificates and Keys Maintenance

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

Component	Storage	Validity
Merchant's Private Key	Private Key should be maintained and handled with the most secure approach that a Merchant can apply. The most common and yet secure enough approach is: • **key password** - Do not save the password in plain text or hard-coded in application. Recommend to encrypt it by any Password Encryption Tools • **key storage** - Store inside password-protected key repository, such as `JKS` or `PKCS12` keystore. Keystore password should also be encrypted.	No restriction on the Validity Period. However, if Merchant suspects there is any chance that the key is leaked or for any other security reason, a new Private Key and its associated Public Key Certificate should be generated.

Component	Storage	Validity
Merchant's Public Key Certificate	Since Public Key Certificate is publicly distributed, a comparative moderate secure storage approach is acceptable. Merchant can store the physical file in any system's file system or store all keys and certificates in one single key repository for a centralised key management.	For a self-signed Certificate, the same condition has been mentioned as above. However, the validity period of a CA-signed Certificate is depended on the purchase plan of the issuing CA. The most common standard is 1 to 2 years.
HSBC's Public Key Certificate	Same as the above	1 Year **NOTE:** Technically, the validity period is usually 1 Year plus 1 to 2 months more. The spare period is a buffer for a merchant to switch a "to-be-expired" Certificate to the new one during the Certificate Renewal Process. More technical detail will be covered in later section.

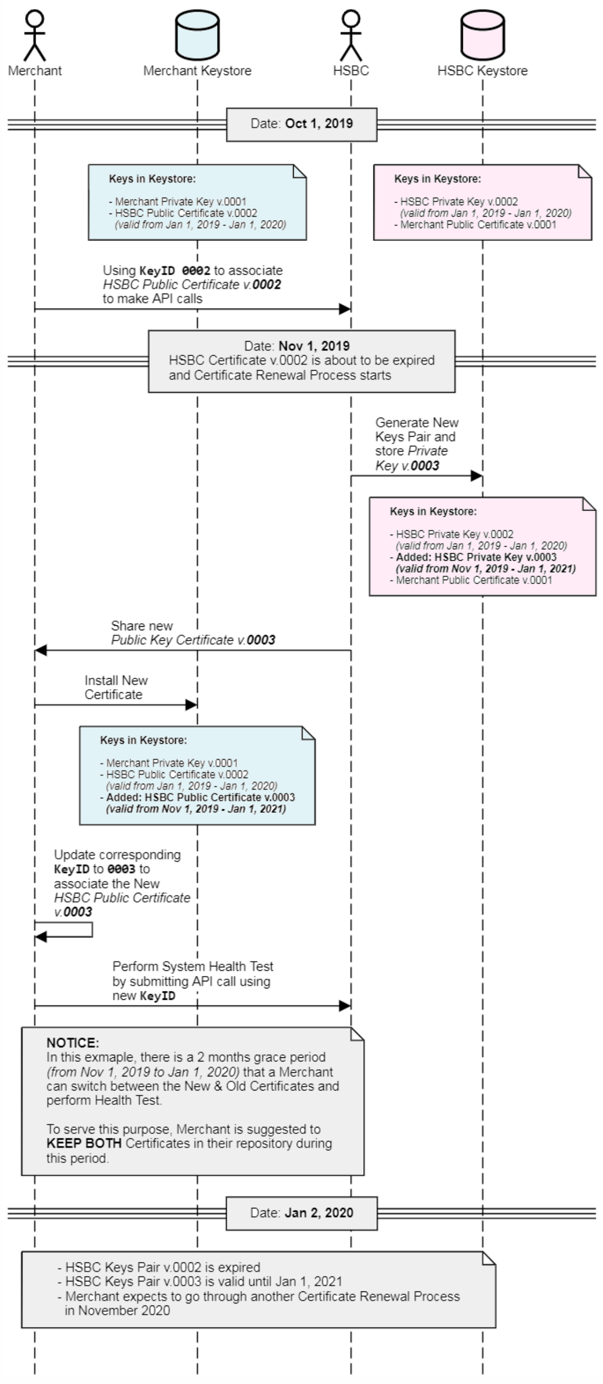
Certificates and Keys Renewal

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

! **SOME RULES YOU SHOULD KNOW:**

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

HSBC Public Key Certificate Renewal (Logical Flow)



Keys in Keystore:
- Merchant Private Key v.0001
- HSBC Public Certificate v.0002 *(valid from Jan 1, 2019 - Jan 1, 2020)*

Keys in Keystore:
- HSBC Private Key v.0002 *(valid from Jan 1, 2019 - Jan 1, 2020)*
- Merchant Public Certificate v.0001

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***

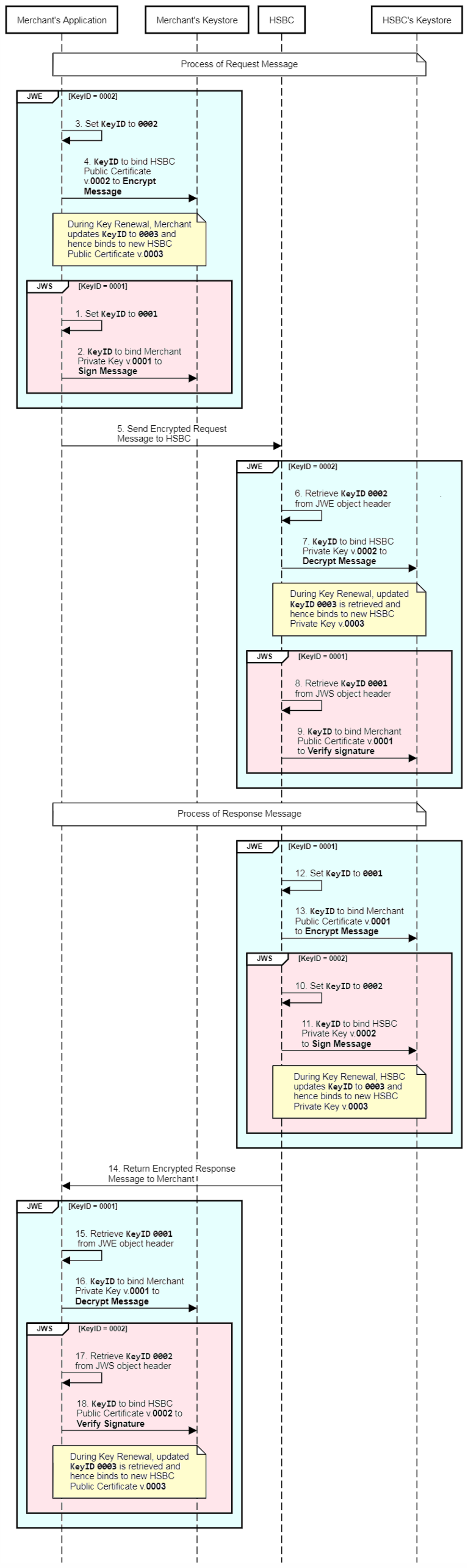
Perform System Health Test by submitting API call using new **KeyID**

NOTICE:
In this exmaple, there is a 2 months grace period *(from Nov 1, 2019 to Jan 1, 2020)* that a Merchant can switch between the New & Old Certificates and perform Health Test.

To serve this purpose, Merchant is suggested to **KEEP BOTH** Certificates in their repository during this period.

- HSBC Keys Pair v.0002 is expired
- HSBC Keys Pair v.0003 is valid until Jan 1, 2021
- Merchant expects to go through another Certificate Renewal Process in November 2020

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


Merchant's Application
Merchant's Keystore
HSBC
HSBC's Keystore
Process of Request Message
JWE [KeyID = 0002]
3. Set KeyID to 0002
4. KeyID to bind HSBC Public Certificate v.0002 to Encrypt Message
During Key Renewal, Merchant updates KeyID to 0003 and hence binds to new HSBC Public Certificate v.0003
JWS [KeyID = 0001]
1. Set KeyID to 0001
2. KeyID to bind Merchant Private Key v.0001 to 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
11. KeyID to bind HSBC Private Key v.0002 to Sign Message
During Key Renewal, HSBC updates KeyID to 0003 and hence binds to new HSBC Private Key v.0003
14. Return Encrypted Response Message to Merchant
JWE [KeyID = 0001]
15. Retrieve KeyID 0001 from JWE object header
16. KeyID to bind Merchant Private Key v.0001 to Decrypt Message
JWS [KeyID = 0002]
17. Retrieve KeyID 0002 from JWS object header
18. KeyID to bind HSBC Public Certificate v.0002 to Verify Signature
During Key Renewal, updated KeyID 0003 is retrieved and hence binds to new HSBC Public Certificate v.0003

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

Disclaimer

IMPORTANT NOTICE