# API Specification for Singapore PAYNOW

Version: 2.9

## Description

This document introduces the **OpenAPI specification** which describes the PAYNOW REST APIs for HSBCs QR code collections. The scope of this document is limited to digital payments in Singapore.

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

## Update Log

- [Jan 18, 2022] **v2.9** Extended the maximum length of field `txnRef` from 25 to 35 chars in statusReturnRequestModel
- [Jan 13, 2022] **v2.8** Revised several content sections
- [Nov 12, 2020] **v2.7** Updated terms of Data Type Overview
- [Jul 15, 2020] **v2.6** Added Download Swagger section
- [Nov 8, 2019] **v2.5** Updated `API Base URL` including both Sandbox and Production
- [Nov 1, 2019] **v2.4** Added new possible value of field `proCode` in qrCodeResponseModel_response
- [Sep 20, 2019] **v2.3** Updated `Disclaimer`
- [Sep 3, 2019] **v2.2**
  - Enhanced Section `GETTING STARTED`
  - Added Content Section `REFERENCE`
- [Jun 14, 2019] **v2.1** Enhanced format of request field `qrExpiry` of Payment QR Code Creation API
- [Mar 28, 2019] **v2.0**
  - Added new field `originatingCustName`
  - Added new optional fields `currency` `amount` in Payment Simulation API
  - Added new API Report Request API
- [Oct 16, 2018] **v1.1** Added HTTP Header `message_encrypt` description in Simulation API
- [Sep 13, 2018] **v1.0** Initial Version for Distribution
- [Aug 31, 2018] **v0.9k** Removed Possible Value `100010` in field `proCode` of Enquiry response
- [Aug 29, 2018] **v0.9j**
  - Modified the possible value of response field `proCode` of QR Code API
- [Aug 16, 2018] **v0.9i** Changed Content Type in HTTP Header of Status Notification API to `text/plain`
- [Aug 9, 2018] **v0.9h** Change maxLength of field `merId` to 15
- [Aug 8, 2018] **v0.9g** Modified time format of field `bankTxnTime` in Enquiry & Status Notification API
- [Jul 17, 2018] **v0.9f** Modified Possible Value of field `proCode` of QR Code & Status Notification API
- [Jul 12, 2018] **v0.9e**
  - Added Conditional field `qrOption`
  - Changed `amtEditInd` to Conditional field
- [Jul 4, 2018] **v0.9d** Changed data format of `qrExpiry`
- [Jun 26, 2018] **v0.9c** Changed field `payMethod` to String
- [Jun 25, 2018] **v0.9b** Added field `amtEditInd`
- [Jun 15, 2018] **v0.9a** Draft 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

There are two API use cases used in this document:

TOPICS

GETTING STARTED

API OPERATIONS

API SCHEMA

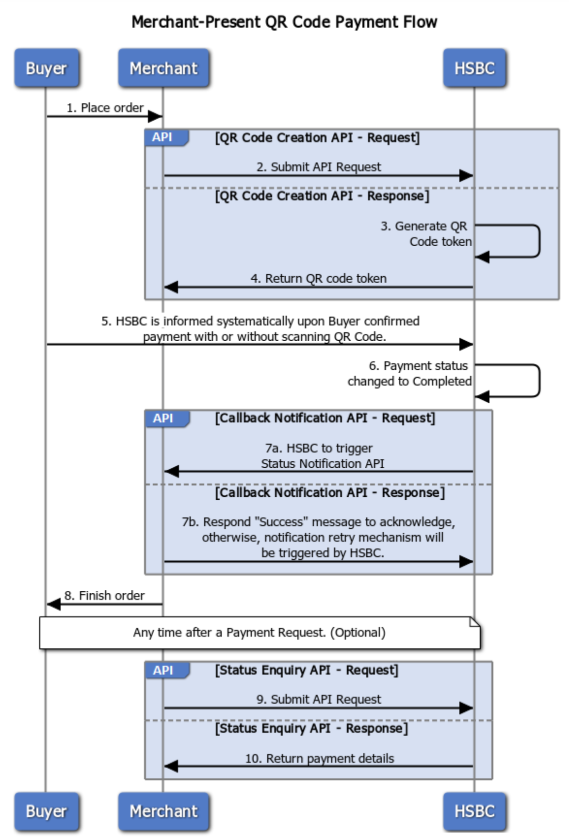
REFERENCE

DISCLAIMER

- The Buyer visits a Merchant's online store to place an order and make a payment by scanning the Merchant-Presented QR Code which is based on the PayNOW specification.
- The Merchant request to HSBC to generate reports on an as-needed basis. After successfully submitting a report request, the Merchant collects reports from their pre-subscribed report collection channels.

## Making a Payment

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

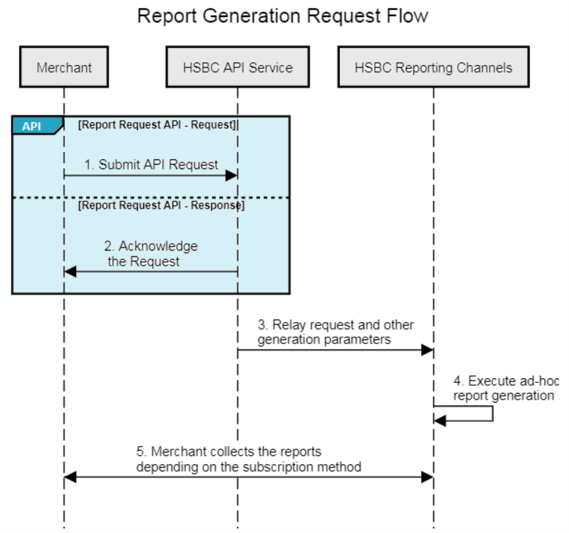

Merchant-Present QR Code Payment Flow
Buyer
Merchant
HSBC
1. Place order
API
[QR Code Creation API - Request]
2. Submit API Request
[QR Code Creation API - Response]
3. Generate QR Code token
4. Return QR code token
5. HSBC is informed systematically upon Buyer confirmed payment with or without scanning 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 converts the QR Code token to a QR Code Image and displays it on its online store. HSBC receives an acknowledgement as soon a the Buyer confirms payment after scanning the QR code.

> ! **NOTE:**
> For testing purpose, the Merchant can choose to call a Payment Simulation API to simulate a payment.

6. The Payment is completed.
7. An acknowledge message is sent back via a Status Notification API immediately after 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 a Status Enquiry API 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.

## Request report generation on as-needed basis


Report Generation Request Flow
Merchant
HSBC API Service
HSBC Reporting Channels
API
[Report Request API - Request]
1. Submit API Request
[Report Request API - Response]
2. Acknowledge the Request
3. Relay request and other generation parameters
4. Execute ad-hoc report generation
5. Merchant collects the reports depending on the subscription method

1. The Merchant submits a Report Request API request.
2. HSBC responds to the Merchant with an acknowledgement.
3. The HSBC API engine relays a request and other generated parameters (e.g. File Format, etc) to HSBC's Reporting Channels.
4. The HSBC Reporting Channels trigger and generate an ad-hoc report.
5. The Merchant collects reports based on their pre-subscribed collection method.

## Summary of API Usability

Here is the Summary of the API Usability over different Payment Models:

| APIs | E-Commerce |
|---|---|
| Payment QR Code Creation API | ✔ |
| Payment Status Enquiry API | ✔ |
| Callback Payment Notification API | ✔ |
| Payment Simulation API | ✔ |
| Report Request API | Opt in |

✔ = Always Available
✘ = Not Available
Opt in = Available when Opt in

# 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 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-mcsg-ea-merchantservices-prod-proxy/v1 |

| Sandbox |
|---|
| https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcsg-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:<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 | | |
|---|---|---|
| **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:<br>`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) |

| **SSL Certificate & Network Whitelist** | | | |
|---|---|---|---|
| **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<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 | |

> ! **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™ |

| keyID of JWS™ & 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 k
  throws UnrecoverableKeyException, KeyStoreException, NoSuchAl
#1  Payload payload = new Payload(messagePayload);

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

#4  PrivateKey privateKey = (PrivateKey) ks.getKey(keyAlias, ke
    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
  "iat": "1625587913"  //Issued At - the time this request is
}
```

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, RS
  throws JOSEException {
#1  Payload jwepayload = new Payload(jwsObject.serialize());

#2  JWEHeader jweheader = new JWEHeader.Builder(JWEAlgorithm.RS
#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, KeyStoreFa
  throws KeyStoreException, NoSuchAlgorithmException, Certifica
         java.text.ParseException, UnrecoverableKeyException, J
#1  JWEObject jweObject = JWEObject.parse(respMsgPayload);

#2  PrivateKey privateKey = (PrivateKey) keyStore.getPrivateKey

    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 k
  throws KeyStoreException, JOSEException, ParseException {
#1  JWSObject jwsObject = JWSObject.parse(signedMessage);

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

#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`<br>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

```
curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.c
 -H "message_encrypt: false"
 -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C
 -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
 -H "Content-Type: application/json"
 -d "{ \"txnRef\": \"PAY-QJZV956664\", \"merId\": \"4229
```

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.co
 -H "message_encrypt: false"
 -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C
 -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.c
 -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C
 -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
 -H "Content-Type: application/json"
 -d "eyJraWQiOiIwMDAxIiwiZW5jIjoiQTEyOEdDTSIsImFsZyI6IlJ
```

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.co
 -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C
 -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. |
| String *(For critical field)* | `0-9` `a-z`<br>`A-Z` `-` `_`<br>`.` | 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>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>**List of Critical Fields:**<br>`merId`<br>`posMachineId`<br>`employeeId` |
| String *(For Restricted field)* | `0-9` `a-z`<br>`A-Z` | Share the same conditions with critical field plus adding extra restriction on characters option.<br>**List of restricted Fields:**<br>`txnRef` |
| 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>Field & value must be present in the request with valid `JSON` format. |
| Optional | Annotated with optional tag in field definition section.<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":" "}`. |

| Field Mandatory Type | Definition & Important Notice |
|---|---|
| Conditional | Annotated with conditional tag in field definition section.<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+8` (Singapore 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>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

×

TOPICS

Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Report Request
Summary

GETTING STARTED

How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary

How to make API request
with Plain Message
with Data Encryption

Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework

API OPERATIONS

Payments
QR Code Creation
Status Enquiry
Callback Status Notification

Simulation
Payment Simulation

Report Management
Report Request

API SCHEMA

Schema Definitions
subAmountObj
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
paySimRequestModel
paySimResponseModel
paySimResponseModel_response
reportRqtRequestModel
reportRqtResponseModel
reportRqtResponseModel_response

REFERENCE

Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal

Download Swagger

DISCLAIMER

Disclaimer

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

# Payments

Contains resource collections for QR Code Creation, payment enquiry and callback notification

Payments

## Payment QR Code Creation API

POST `/payment/qrCode`

### DESCRIPTION

This API 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 PayNOW specification. The Merchant then displays the QR Code to Buyer, for payment initiation process.

### REQUEST PARAMETERS

| | |
|---|---|
| **Authorization**<br>optional<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>optional<br>in header | [Merchant ID]+[JWS ID]+[JWE ID] |
| **Content-Type**<br>required<br>in header | application/json |

### REQUEST BODY

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

### RESPONSES

| | |
|---|---|
| **200 OK**<br>qrCodeResponseModel | Successful operation.<br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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. |

Request Content-Types: application/json

Request Example

```
{
  "merId": "S123456S0010001",
  "txnRef": "SGHSBC000001234567F064577",
  "txnChannel": "01",
  "txnTime": "2018-06-11T14:10:25Z",
  "qrExpiry": "20180625121010",
  "payMethod": "SGQR",
  "amtEditInd": "Y",
  "qrOption": "02",
  "country": "SG",
  "currency": "SGD",
  "amount": 1050,
  "notifyUrl": "https://merchant.com/returnStatus",
  "goodsDes": "Description of goods.",
  "posMachineId": "00112233-4455-6677-8899-aabbccddeeff",
  "employeeId": "00112233-4455-6677-8899-xxyyzzxxyyzz"
}
```

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": {
    "txnRef": "SGHSBC000001234567F064577",
    "currency": "SGD",
    "amount": 1050,
    "proCode": "000000",
    "proMsg": "Transaction Successful",
    "qrCode": "QR_CODE_DATA"
  }
}
```

Response Example (400 Bad Request)

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

Payments

## Payment Status Enquiry API

POST `/payment/enquiry`

### DESCRIPTION

The Merchant can initiate payment status enquiry any time after a payment request is submitted. HSBC will return the latest transaction status based on the transaction ID provided by the Merchant.

### REQUEST PARAMETERS

| | |
|---|---|
| **Authorization**<br>optional<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>optional<br>in header | [Merchant ID]+[JWS ID]+[JWE ID] |
| **Content-Type**<br>required<br>in header | application/json |

### REQUEST BODY

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

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "merId": "S123456S0010001"
}
```

### RESPONSES

| | |
|---|---|
| **200 OK**<br>txnEnqResponseModel | Successful operation.<br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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": "SGHSBC000001234567F064577",
    "proCode": "000000",
    "proMsg": "Payment Success",
    "currency": "SGD",
    "totalAmtPaid": 2100,
    "arrayOfSubAmt": [
      {
        "bankTxnId": "GPS0000123456789",
        "bankTxnTime": "2018-06-11T14:10:25+08:00",
        "subAmtPaid": 1050,
        "originatingCustName": "Sean Mante"
      },
      {
        "bankTxnId": "GPS0000123456999",
        "bankTxnTime": "2018-06-11T15:11:12+08:00",
        "subAmtPaid": 1050,
        "originatingCustName": "Sean Mante"
      }
    ]
  }
}
```

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

## Callback Payment Notification API

POST `/<Callback URL predefined by Merchant>`

### DESCRIPTION

Payment status will be returned to the Merchant by asynchronous callback once HSBC receives a payment request. Once HSBC receives the payment result from the clearing system, it will push the result back to the Merchant by calling this API.

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

> ! **Retry Mechanism** If unsuccessful response, 4 retries will be triggered in every 2 minutes. There will be maximum 5 API calls including the 1st attempt.

> ! **Endpoint Definition** Field `notifyUrl` from QR Code Creation API will be used as URL endpoint of the corresponding transaction.

> ! **Exception Handling** Only successful case will be returned. The Merchant can submit an Payment Status Enquiry API request if there is no acknowledge message returned after a certain period of time.

REQUEST PARAMETERS

| | |
|---|---|
| **Content-Type**<br>required<br>in header<br>string | text/plain |

REQUEST BODY

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

Request Content-Types: text/plain

Request Example

```
{
  "merId": "S123456S0010001",
  "txnRef": "SGHSBC000001234567F064577",
  "currency": "SGD",
  "amount": 1050,
  "originatingCustName": "Sean Mante",
  "proCode": "000000",
  "proMsg": "Completed",
  "bankTxnId": "T18050209732",
  "bankTxnTime": "2018-06-11T14:10:25+08:00"
}
```

RESPONSES

| | |
|---|---|
| **200 OK**<br>statusReturnResponseModel | Successful operation.<br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |

Response Content-Types: application/json

Response Example (200 OK)

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

## Simulation

Contains resource collections for simulating specific scenario which is only available in sandbox environment

Simulation

## Payment Simulation API

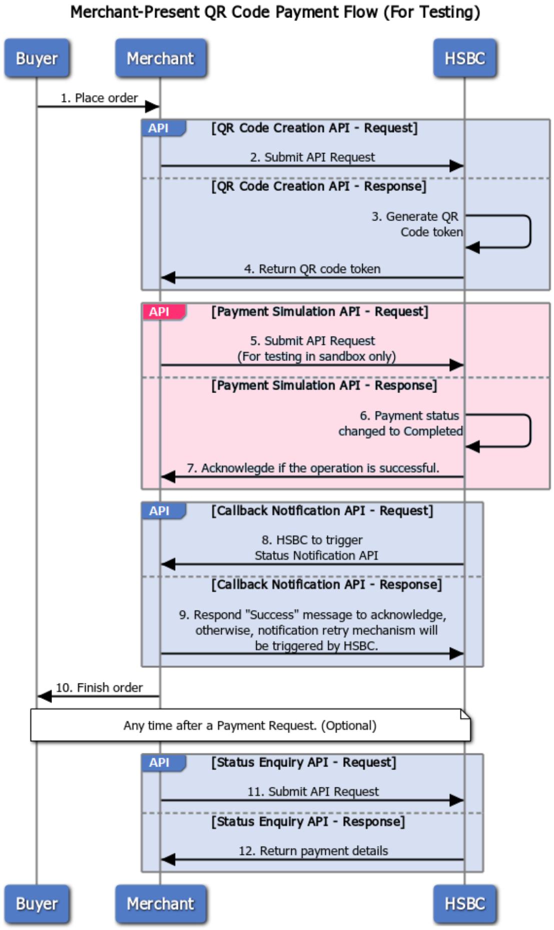
POST `/sg/payment/simulation`

**A different Host URL is being used for this API, please refer to API Gateway URL for details.*

DESCRIPTION

During the client system development phase, the Merchant can call this API to simulate a payment of buyer. This API is only available in sandbox environment.

We use the below Testing Flow diagram to illustrate the Concept:


Merchant-Present QR Code Payment Flow (For Testing)
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
API
[Payment Simulation API - Request]
5. Submit API Request (For testing in sandbox only)
[Payment Simulation API - Response]
6. Payment status changed to Completed
7. Acknowlegde if the operation is successful.
API
[Callback Notification API - Request]
8. HSBC to trigger Status Notification API
[Callback Notification API - Response]
9. Respond "Success" message to acknowledge, otherwise, notification retry mechanism will be triggered by HSBC.
10. Finish order
Any time after a Payment Request. (Optional)
API
[Status Enquiry API - Request]
11. Submit API Request
[Status Enquiry API - Response]
12. Return payment details
Buyer
Merchant
HSBC

## REQUEST PARAMETERS

| | |
|---|---|
| **Content-Type**<br>required<br>in header<br>string | application/json |
| **message_encrypt**<br>required<br>in header<br>string | false |

## REQUEST BODY

| | |
|---|---|
| paySimRequestModel | Message Encryption is not needed for this message payload |

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "merId": "S123456S0010001",
  "currency": "SGD",
  "amount": 1050,
  "originatingCustName": "Sean Mante",
  "is_notification_encrypted": "Y"
}
```

## RESPONSES

| | |
|---|---|
| **200 OK**<br>paySimResponseModel | Successful operation. |
| **400 Bad Request**<br>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": "SGHSBC000001234567F064577",
    "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"
  }
}
```

## SECURITY

| Schema | Scopes |
|---|---|
| Client ID | |
| Client Secret | |

# Report Management

Contains resource collections for reporting

Report Management

## Report Request API

POST `/reporting/request`

DESCRIPTION

Merchant can request HSBC to generate reports on an as-needed basis. After successfully submit a report request, Merchant can collect their reports from their pre-subscribed report collection channels.

REQUEST PARAMETERS

| | |
|---|---|
| **Authorization**<br>optional<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>optional<br>in header | [Merchant ID]+[JWS ID]+[JWE ID] |
| **Content-Type**<br>required<br>in header | application/json |

Request Content-Types: application/json

Request Example

```
{
  "merId": "S123456S0010001",
  "reportName": "DETAIL REPORT",
  "reportFormat": "PDF",
  "reportDate": "20190320"
}
```

REQUEST BODY

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

RESPONSES

| | |
|---|---|
| **200 OK**<br>reportRqtResponseModel | Successful operation.<br>*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.* |
| **400 Bad Request**<br>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": {
    "proCode": "000000",
    "proMsg": "Report Request Successful"
  }
}
```

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

## subAmountObj: object

PROPERTIES

**bankTxnId:** string range: (up to 16 chars) required

HSBC transaction reference id for the inward credit payment

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

HSBC Transaction time for the inward credit payment

Example

```
{
  "bankTxnId": "T18050209732",
  "bankTxnTime": "2018-06-11T14:10:25+08:00",
  "subAmtPaid": 1050,
  "originatingCustName": "Sean Mante"
}
```

- Bank system local time. A `GMT+8` timezone information is appended to the end of the timestamp to indicate this time is a Singapore local time. Format：`yyyy-MM-dd'T'HH:mm:ss±hh:mm`

**subAmtPaid:** integer range: 1 ≤ x ≤ 999999999999 required
Amount of money paid by payer per payment submission

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

**originatingCustName:** string range: (up to 140 chars) required
Returning Ordering/Originating Customer Name

## 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.
- Developer is suggested to first catch the native HTTP Return Code before trying to look into the `returnCode` and `returnReturn` inside the json message.

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

**sentTime:** string range: (up to 27 chars) required

Example

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

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

## exceptionModel: object

PROPERTIES

**api_gw:** commonRespObj required

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"
  }
}
```

## qrCodeRequestModel: object

PROPERTIES

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

- Distributed by HSBC to the merchant for identifying each merchant's identity

**txnRef:** string range: (up to 25 chars) required
Unique ID referred to a specific transaction

- Requires Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 25 characters
- This txnRef will be embedded in the QR code generated, and will be shown to Payer for confirmation during scan and pay of QR code. If PayNow scheme is selected by the payer to settle the payment, HSBC will return this txnRef field in outbound API call to merchant for payment status notification

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

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

**txnTime:** string range: (up to 20 chars) required
Time of sending out this request transaction

- Client system time. The timezone is expected to be `GMT+8` (Singapore local time). Merchant is required to perform timezone conversion if needed. Format: `yyyy-MM-dd'T'HH:mm:ssZ`

**qrExpiry:** string range: (up to 14 chars) optional
QR code Expiry

- Format： `yyyyMMdd` or `yyyyMMddHHmmss`
- When payer scan the QR code and the transaction date is later than QR expiry date, payer bank will not proceed the with PayNow payment to the merchant
- If this field is not provided, it means no expiry date will be specified in the generated QR code. Otherwise, the value must be equal or later than today's date

**payMethod:** string enum: [ SGQR, PAYNOW ] range: (up to 10 chars) required
Payment Method

| Possible Value | Definition |
|---|---|
| SGQR | Singapore QR Code for E-Payments |
| PAYNOW | PayNow QR Code Payment |

**amtEditInd:** string enum: [ Y, N ] range: (up to 1 chars) conditional
Amount Editable Indicator

- To indicate if the amount can be edited by payer or not Please refer to PayNOW spec for details. Can specify Y/N here
- Required when `payMethod = "PAYNOW"`

> ! **NOTICE:** This option is only for **PayNow** but not applicable to SGQR

Example

```
{
  "merId": "S123456S0010001",
  "txnRef": "SGHSBC000001234567F064577",
  "txnChannel": "01",
  "txnTime": "2018-06-11T14:10:25Z",
  "qrExpiry": "20180625121010",
  "payMethod": "SGQR",
  "amtEditInd": "Y",
  "qrOption": "02",
  "country": "SG",
  "currency": "SGD",
  "amount": 1050,
  "notifyUrl": "https://merchant.com/returnStatus",
  "goodsDes": "Description of goods.",
  "posMachineId": "00112233-4455-6677-8899-aabbccddeeff",
  "employeeId": "00112233-4455-6677-8899-xxyyzzxxyyzz"
}
```

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

- Required when `payMethod = "PAYNOW"`

> ! **NOTICE:** This option is only for **PayNow** but not applicable to SGQR

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

**country:** string enum: [ SG ] range: (up to 2 chars) required
Country Code (Format: ISO 3166-1 alpha-2)

| Possible Value | Definition |
|---|---|
| SG | Singapore |

**currency:** string enum: [ SGD ] range: (up to 3 chars) required
Payment Currency (Format: ISO 4217 Alpha)

| Possible Value | Definition |
|---|---|
| SGD | Singapore Dollar |

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

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

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

**goodsDes:** string range: (up to 128 chars) optional
Description of goods.

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

- Required when `txnChannel = "01"`

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

- Required when `txnChannel = "01"`

## qrCodeResponseModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** qrCodeResponseModel_response required

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": "SGHSBC000001234567F064577",
    "currency": "SGD",
    "amount": 1050,
    "proCode": "000000",
    "proMsg": "Transaction Successful",
    "qrCode": "QR_CODE_DATA"
  }
}
```

## qrCodeResponseModel_response:

PROPERTIES

**txnRef:** string range: (up to 25 chars) required
Returning back the original Transaction Reference No. provided by merchant

**currency:** string range: (up to 3 chars) required
Returning back Payment Currency

- Format: ISO 4217 Alpha (e.g. SGD = Singapore Dollar)

**amount:** integer range: 1 ≤ x ≤ 999999999999 required

Example

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "currency": "SGD",
  "amount": 1050,
  "proCode": "000000",
  "proMsg": "Transaction Successful",
  "qrCode": "QR_CODE_DATA"
}
```

Returning back Payment Amount

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

| Possible Value | Definition |
|---|---|
| 000000 | Transaction Successful |
| 900020 | Merchant ID Not Found |
| 900030 | Duplicate Transaction Reference |
| 900040 | Payment Currency and Settlement Currency is Not Matched |
| 900050 | Transaction Channel Not Available with the corresponding Merchant |

- Other than "000000", all other return codes indicate a fail case.

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

**qrCode:** string range: (up to 20000 chars) conditional
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)

## txnEnqRequestModel: object

PROPERTIES

**txnRef:** string range: (up to 25 chars) required
Merchant to provide transaction ID that referring to a specific transaction

**merId:** string range: (up to 15 chars) required
Merchant to provide Merchant ID for identification

Example

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "merId": "S123456S0010001"
}
```

## txnEnqResponseModel: object

PROPERTIES

**api_gw:** commonRespObj required
**response:** txnEnqResponseModel_response required

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": "SGHSBC000001234567F064577",
    "proCode": "000000",
    "proMsg": "Payment Success",
    "currency": "SGD",
    "totalAmtPaid": 2100,
    "arrayOfSubAmt": [
      {
        "bankTxnId": "GPS0000123456789",
        "bankTxnTime": "2018-06-11T14:10:25+08:00",
        "subAmtPaid": 1050,
        "originatingCustName": "Sean Mante"
      },
      {
        "bankTxnId": "GPS0000123456999",
        "bankTxnTime": "2018-06-11T15:11:12+08:00",
        "subAmtPaid": 1050,
        "originatingCustName": "Sean Mante"
      }
    ]
  }
}
```

## txnEnqResponseModel_response:

PROPERTIES

**txnRef:** string range: (up to 25 chars) required
Returning back the original Transaction Reference No. provided by merchant

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

| Possible Value | Definition |
|---|---|

Example

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "proCode": "000000",
  "proMsg": "Payment Success",
  "currency": "SGD",
  "totalAmtPaid": 2100,
  "arrayOfSubAmt": [
    {
      "bankTxnId": "GPS0000123456789",
```

| Possible Value | Definition |
|---|---|
| 000000 | Payment Success |
| 900010 | Transaction Record Not Found |
| 900020 | Merchant ID Not Found |

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

**currency:** string range: (up to 3 chars) required
Payment Currency

- Format: ISO 4217 Alpha

**totalAmtPaid:** integer range: 1 ≤ x ≤ 999999999999 required
Total amount of money paid by payer

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

> ! **NOTICE:** Customer is able to submit multiple payments to merchant regarding to the same transaction. This amount can be varied if field `amtEditInd` is set to `Y`

**arrayOfSubAmt:** Array< subAmountObj > conditional
List of Sub-Amount Object

- Exist if field `totalAmtPaid > 0`

```
        "bankTxnTime": "2018-06-11T14:10:25+08:00",
        "subAmtPaid": 1050,
        "originatingCustName": "Sean Mante"
      },
      {
        "bankTxnId": "GPS0000123456999",
        "bankTxnTime": "2018-06-11T15:11:12+08:00",
        "subAmtPaid": 1050,
        "originatingCustName": "Sean Mante"
      }
    ]
  }
```

## statusReturnRequestModel: object

PROPERTIES

**merId:** string range: (up to 15 chars) required
Returning back the Merchant ID for Merchant identification.

**txnRef:** string range: (up to 35 chars) required
Returning back the Transaction Reference No.

**currency:** string range: (up to 3 chars) required
Returning back the Payment Currency

- Format: ISO 4217 Alpha (e.g. SGD = Singapore Dollar)

**amount:** integer range: 1 ≤ x ≤ 999999999999 required
Returning back the Payment Amount

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

**originatingCustName:** string range: (up to 140 chars) required
Returning Ordering/Originating Customer Name

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

| Possible Value | Definition |
|---|---|
| 000000 | Completed |

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

**bankTxnId:** string range: (up to 16 chars) required
Returning HSBC transaction reference id for the inward credit payment

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

- Bank system local time. A `GMT+8` timezone information is appended to the end of the timestamp to indicate this time is a Singapore local time.
  Format : `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

Example

```
{
  "merId": "S123456S0010001",
  "txnRef": "SGHSBC000001234567F064577",
  "currency": "SGD",
  "amount": 1050,
  "originatingCustName": "Sean Mante",
  "proCode": "000000",
  "proMsg": "Completed",
  "bankTxnId": "T18050209732",
  "bankTxnTime": "2018-06-11T14:10:25+08:00"
}
```

## statusReturnResponseModel: object

PROPERTIES

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

Example

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

## paySimRequestModel: object

PROPERTIES

**txnRef:** string range: (up to 25 chars) required
Merchant to provide transaction ID that referring to a specific transaction

**merId:** string range: (up to 15 chars) required
Merchant to provide Merchant ID for identification

**currency:** string enum: [ SGD ] range: (up to 3 chars) optional
Payment Currency (Format: ISO 4217 Alpha)

| Possible Value | Definition |
|---|---|
| SGD | Singapore Dollar |

**amount:** integer range: 1 ≤ x ≤ 999999999999 optional
Payment Amount

- Original Payment Amount can be overridden by passing this Amount in Payment Simulation
- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

**originatingCustName:** string range: (up to 140 chars) optional
Ordering/Originating Customer Name

**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

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "merId": "S123456S0010001",
  "currency": "SGD",
  "amount": 1050,
  "originatingCustName": "Sean Mante",
  "is_notification_encrypted": "Y"
}
```

## paySimResponseModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** paySimResponseModel_response 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": {
    "txnRef": "SGHSBC000001234567F064577",
    "proCode": "000000",
    "proMsg": "Payment Success"
  }
}
```

## paySimResponseModel_response:

PROPERTIES

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

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

| Possible Value | Definition |
|---|---|
| 000000 | Payment Success |
| 900010 | Transaction Record Not Found |
| 900020 | Merchant ID Not Found |

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

Example

```
{
  "txnRef": "SGHSBC000001234567F064577",
  "proCode": "000000",
  "proMsg": "Payment Success"
}
```

## reportRqtRequestModel: object

PROPERTIES

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

- Distributed by HSBC to the merchant for identifying each merchant's identity

Example

```
{
  "merId": "S123456S0010001",
  "reportName": "DETAIL REPORT",
  "reportFormat": "PDF",
  "reportDate": "20190320"
}
```

**reportName:** string enum: [ DETAIL REPORT ] range: (up to 255 chars) required
Report Name

- Allow to request one report at one time

| Possible Value | Definition |
|---|---|
| DETAIL REPORT | Daily Detail Report |

**reportFormat:** string enum: [ TXT, CSV, XLS, PDF ] range: (up to 3 chars) required
Report Format

- Allow to request one report at one time

| Possible Value | Definition |
|---|---|
| TXT | Text Format |
| CSV | Comma-Separated Values Format |
| XLS | Microsoft Excel |
| PDF | Adobe PDF |

**reportDate:** string range: (up to 8 chars) required
The Date of the report requested

- Format: yyyyMMdd
- If the date is today, report records will be shown up to the request time

## reportRqtResponseModel: object

PROPERTIES

**api_gw:** commonRespObj required
**response:** reportRqtResponseModel_response required

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": {
    "proCode": "000000",
    "proMsg": "Report Request Successful"
  }
}
```

## reportRqtResponseModel_response:

PROPERTIES

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

| Possible Value | Definition |
|---|---|
| 000000 | Report Request Successful |
| 900020 | Merchant ID Not Found |
| 900030 | Report Date Invalid (Invalid Format or Later than today) |

- Other than "000000", all other return codes indicate a fail case.

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

Example

```
{
  "proCode": "000000",
  "proMsg": "Report Request Successful"
}
```

# 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 "y
  [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 |
|---|---|---|

| Alias name | Corresponding Object | Remark |
|---|---|---|
| merchant_key_pair | • Merchant's Private Key<br>• Merchant's Public Certificate (Self-Signed) | These tw objects appear tc be one entry in a JAVA Keystore Merchant can still export them separatel into two objects (files) on your loca file systei dependin on your applicatic design. |
| merchant_signed_cert_0001 | • Merchant's Public Certificate (CA-Signed) | Not exist Merchant skips stej #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. |

×

TOPICS

Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Report Request
Summary

GETTING STARTED

How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary

How to make API request
with Plain Message
with Data Encryption

Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework

API OPERATIONS

Payments
QR Code Creation
Status Enquiry
Callback Status Notification

Simulation
Payment Simulation

Report Management
Report Request

API SCHEMA

Schema Definitions
subAmountObj
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
paySimRequestModel
paySimResponseModel
paySimResponseModel_response
reportRqtRequestModel
reportRqtResponseModel
reportRqtResponseModel_response

REFERENCE

Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal

Download Swagger

DISCLAIMER

Disclaimer

| 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.<br>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<br>**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.

TOPICS

GETTING STARTED

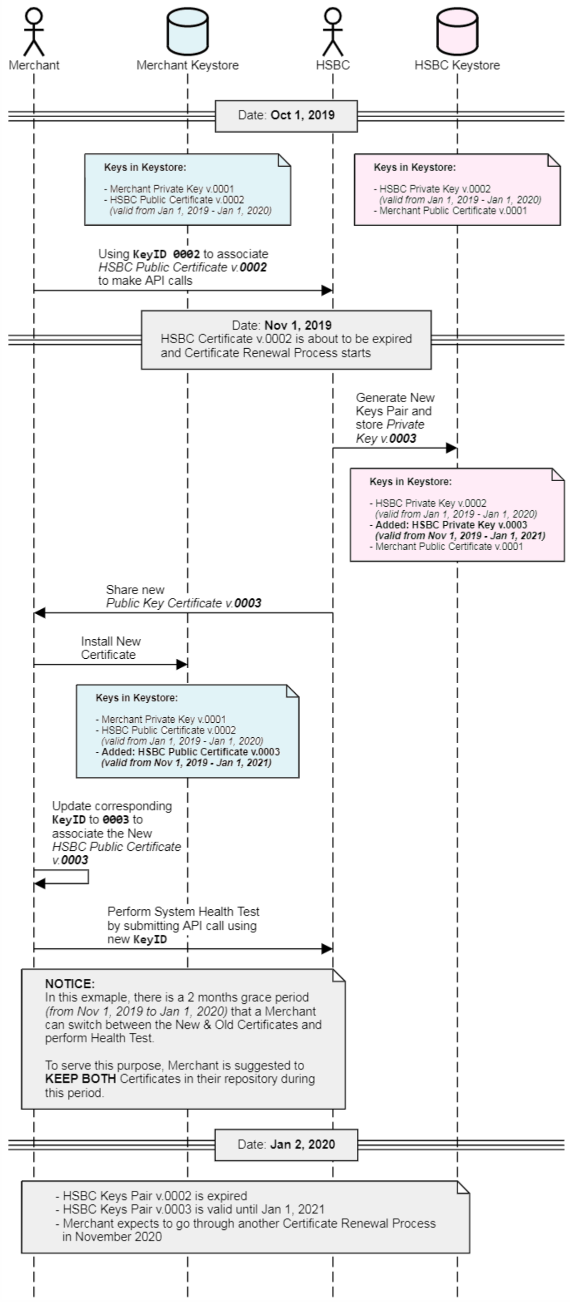
API OPERATIONS

API SCHEMA

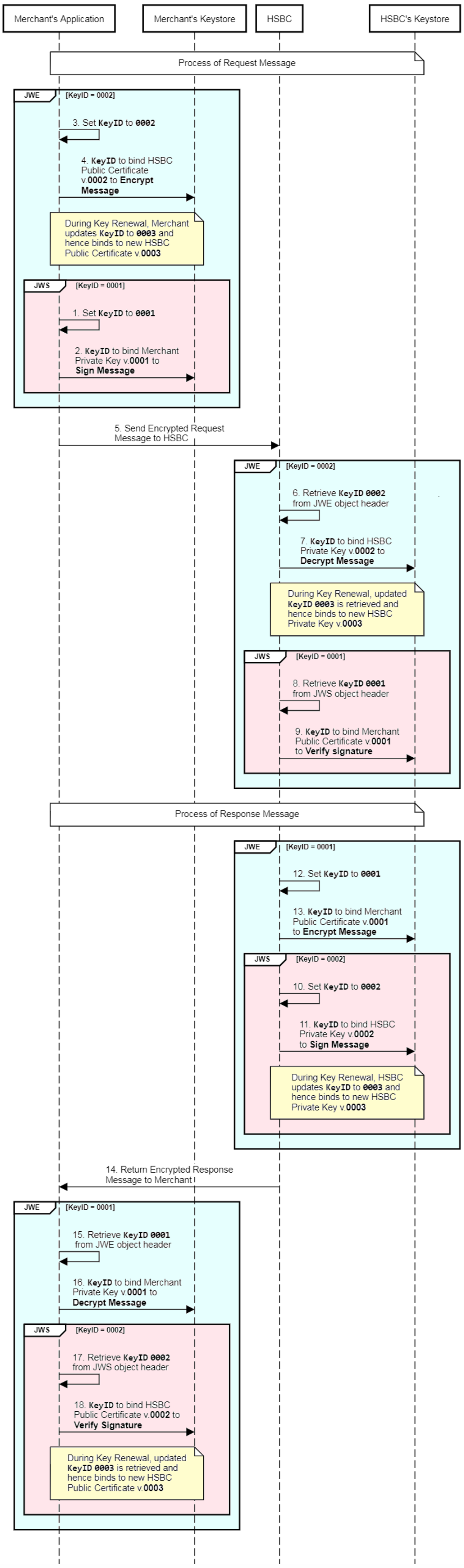
REFERENCE

DISCLAIMER

HSBC Public Key Certificate Renewal (Logical Flow)


Merchant
Merchant Keystore
HSBC
HSBC Keystore
Date: **Oct 1, 2019**
**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.
Date: **Jan 2, 2020**
- 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.

## 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. Copyright: HSBC Group 2019. ALL RIGHTS RESERVED.