# API Specification for Thailand Cards and Alternate Payment Methods

Version: 2.1

## Description

This document introduces the **OpenAPI specification** describing the REST APIs of HSBC's ASP Omni Collection for Thailand Cards and Alternate Payment Methods.

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

## Update Log

- [Dec 21, 2021] **v2.1** Revised several content sections
- [Nov 1, 2021] **v2.0** Added Void and Refund feature through API
- [Aug 25, 2021] **v1.14**
  - Added Installment Payment Support
  - Enhanced `payment_option` to support multiple options

    > **NOTE:**
    > Field type is changed from String to String Array, yet the old format used in version 1.13 is still supported.

- [Jul 10, 2020] **v1.13**
  - Added Download Swagger section
  - Changed `browser_info` to optional at notif_rqt_system_Obj
  - Changed `payment_channel` `payment_scheme` and `process_by` to conditional at notif_rqt_payment_Obj
- [Nov 8, 2019] **v1.12** Updated `API Base URL` including both Sandbox and Production
- [Sep 20, 2019] **v1.11** Updated `Disclaimer`
- [Sep 16, 2019] **v1.10**
  - Enhanced Section `API Connectivity`
  - Enhanced Section `REFERENCE`
- [Apr 15, 2019] **v1.9** Changed maximum length of request field `email` at Page Redirect API
- [Mar 7, 2019] **v1.8** Added Possible Value at APM Channel Code
- [Feb 26, 2019] **v1.7** Change JSON message object name `credit_card` to `online_cc` and `apm` to `offline`
- [Nov 20, 2018] **v1.6**
  - Added response field `txnSubStatus` in Enquiry API
  - Added Array object `refund` in Enquiry API response
- [Nov 9, 2018] **v1.5**
  - Updated definition of field `email` `currency` `payment_option` `payment_expiry`
  - Updated string definition of `Data Type Overview`
  - Added response field `payment_scheme` `process_by` in Enquiry API
- [Nov 2, 2018] **v1.4** Enhanced `APM Agent Code` and `APM Channel Code` in Reference Section.
- [Nov 1, 2018] **v1.3** Updated the supported Payment Channels in Section `Channels and Features` and `Payment Channel Code`
- [Oct 25, 2018] **v1.2**
  - Changed API DateTime format to `yyyy-MM-dd'T'HH:mm:ss±hh:mm`
  - Changed range of API field `amount`
  - Enhanced Content and Rectified miscellaneous typo
- [Oct 12, 2018] **v1.1** Updated the retry logic of Callback Payment Notification API
- [Oct 3, 2018] **v1.0** Initial Draft Version for Merchant Distribution
- [Jul 16, 2018] **v0.9c**
  - Changed API Endpoint
  - Renamed field `txnId` to `txnRef`
- [May 24, 2018] **v0.9b** Initial Version for Development Distribution
- [May 16, 2018] **v0.9a** Initial Draft

## 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

The HSBC Omni Collection provides a wide range of online payment solutions - enabling online merchants to process payments including credit / debit card, PayPal. e-Wallet and Internet Banking (see the table below). The platform supports implementations with either websites or mobile applications.

Using our APIs services, you can build your own eCommerce website and accept payments including the following payment channels.

| **Credit / Debit Card Payments** |
| --- |
| Visa |
| MasterCard |
| Diners |
| China UnionPay |
| American Express |
| JCB |

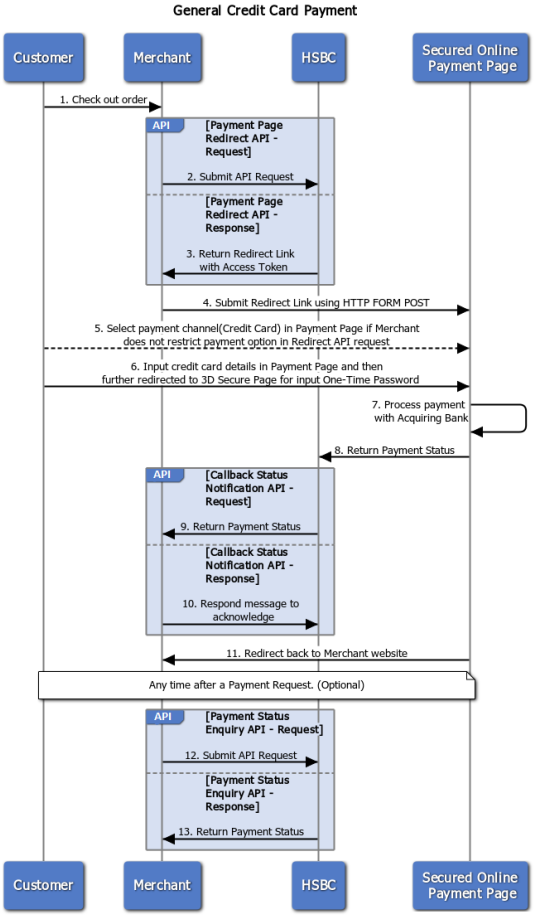
| **Alternative Payments (as known as 1-2-3 Payment)** | **API Mode** |
| --- | --- |
| Direct Debit / Web Pay | Online Payment (Redirect to Bank Website) |
| PayPal | Online Payment (Redirect to PayPal Website) |
| eWallet / QR Code Payment<br>• Alipay<br>• WeChat Pay<br>• LinePay<br>• Samsung Pay<br>• Prompt Pay<br>• mVISA QR<br>• MasterPass QR | Online Payment (QR Code Payment) |

| Alternative Payments (as known as 1-2-3 Payment) | API Mode |
|---|---|
| Cash payments over Convenience Stores or Kiosk<br>• Tesco lotus<br>• BIGC<br>• Pay at Post<br>• MPAY<br>• True Money<br>• CenPay by Central<br>• Boonterm | Offline Payment (Generate Payment Code) |
| Cash payments over Bank Counters or ATM<br>• The Siam Commercial Bank<br>• Krung Thai Bank<br>• TTB Bank<br>• United Overseas Bank<br>• Bank of Ayudhya<br>• Bangkok Bank PCL<br>• Kasikorn Bank | Offline Payment (Generate Payment Code) |

## Credit Card / Debit Card Payments

Credit card transactions by a Thailand Online Merchant usually require additional security from the issuer Bank, this is called 3D Secure. The process asks the credit card holder to enter an Internet PIN, or a One Time PIN(OTP) usually sent to the Credit Card Holder's mobile phone.

### API Use Case




1. The Customer conducts a checkout process in merchant's website.
2. The Merchant submits a Payment Page Redirect API request to HSBC.
3. HSBC returns a JSON response which embeds the redirect link of the Secured Online Payment Page with an access token inside the field `redirectLink`. The redirect link is in a `HTML FORM POST` format. More details are covered in the Payment Page Redirect API.
4. The Merchant submits the redirect link using a `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> By passing the value `C` in the optional field `payment_option`, the Merchant can restrict the payment page to show only Credit Card payments.

5. The Customer can select different credit / debit card brands, e-Wallet, etc (see the table in Credit / Debit Card Payments) in the Payment Gateway - providing the Merchant does not restrict it by passing a value in the API request field `payment_option`. See also the Notice in step 4 above.
6. The Customer Credit Card details in the Payment Page are then redirected to a 3D Secure (3DS) Page to input a One-Time password. 3DS is optional. Please contact HSBC to enable this feature if required.
7. The payment page securely connects to the bank's backend systems to process the payment.
8. HSBC receives the payment status once it is updated from the backend system.
9. HSBC triggers a Callback Payment Notification API and sends the payment status back to the Merchant.

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

10. To acknowledge, the Merchant sends a response to the Callback API. Failure to return a correct response triggers a Notification resend mechanism.
11. A redirect is sent back to merchant website once the payment process is completed in the Payment Gateway.

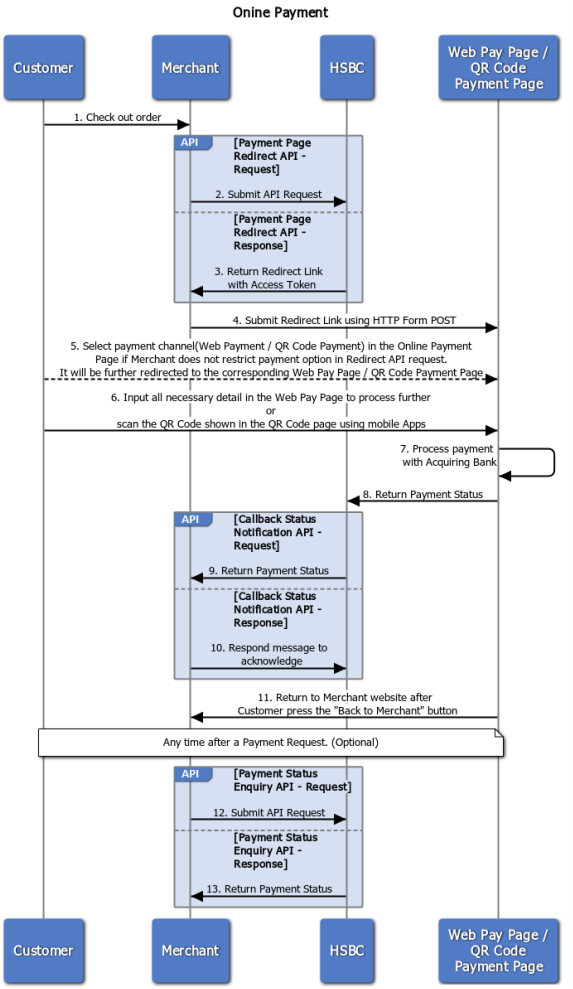
> **NOTE:**
> The Merchant can define the redirect back URL in request field `redirectUrl` in the Payment Page Redirect API.

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

# Online Payments

Payments are processed and confirmed by either a bank or payment agent in real time, or the customer must be online to scan QR code to make a payment during a QR Code payment scenario.

## API Use Case




1. The Customer conducts a checkout process in merchant's website.
2. The Merchant submits a Payment Page Redirect API request to HSBC.
3. HSBC returns a JSON response which embeds the redirect link of the Secured Online Payment Page with an access token inside the field `redirectLink`. The redirect link is in a `HTML FORM POST` format. More details are covered in the Payment Page Redirect API.
4. The Merchant submits the redirect link using a `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> The Merchant can also restrict the payment page to show only a 1-2-3 payment option, or directly show the QR Code page by passing the corresponding value in the `payment_option` field in a Redirect API request.

5. The Customer can select different payment channels - providing the Merchant does not pass a value in `payment_option` in the Redirect API request.
6. The Customer inputs all necessary details into the Web Payment Page (Bank's website) or in a QR Code payment scenario, i.e the customer scans the QR Code to make a payment with their mobile App.
7. The Payment is processed in the acquiring bank's backend system.
8. HSBC receives the payment status as soon as it is updated at the backend system.
9. HSBC triggers a Callback Payment Notification API and sends the payment status back to the Merchant.

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

10. To acknowledge, the Merchant sends a response to the Callback API. Failure to return a correct response triggers a Notification resend mechanism.
11. A redirect is sent back to merchant website once the payment process is completed in the Payment Gateway.

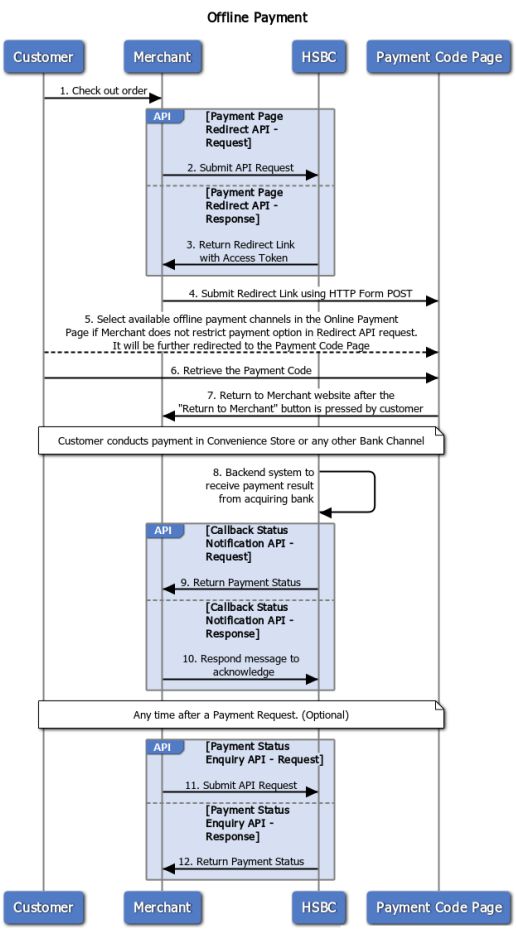
> **NOTE:**
> The Merchant can define the redirect back URL in request field `redirectUrl` in the Payment Page Redirect API.

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

# Offline Payments

In this flow example, a Payment is pending between request and payment.

## API Use Case




1. The Customer conducts a checkout process in merchant's website.
2. The Merchant submits a Payment Page Redirect API request to HSBC.
3. HSBC returns a JSON response which embeds the redirect link of the Secured Online Payment Page with an access token inside the field `redirectLink`. The redirect link is in a `HTML FORM POST` format. More details are covered in the Payment Page Redirect API.
4. The Merchant submits the redirect link using a `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> The Merchant permits Customer access to the Payment Code Page by passing the corresponding Payment Option values in the request message.

5. The Customer can select different payment channels - providing the Merchant does not pass a value in `payment_option` in the Redirect API request.
6. The Customer gets the payment Code.
7. The browser redirects back to merchant website from the Payment/QR Code page.

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

8. HSBC's backend system receives the payment status as soon as the payment process is completed at the acquiring bank.
9. HSBC triggers a Callback Payment Notification API and sends the payment status back to the Merchant.

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

10. The Merchant responds to the API with an acknowledge. Failure to return a proper response triggers the Notification resend mechanism.
11. The Merchant can optionally submit a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant finds no acknowledge message returned after a certain period of time.
12. HSBC returns the latest payment status according to the transaction reference number the Merchant provided.

## Check Status Feature

The Omni collection provides features for the merchant to check the status of every payment transaction. To implement Check Status, please refer to the Status Enquiry API.

## Void & Refund

The Merchant can request a Void API to cancel an existing order where the payment transaction is unsettled.

The Merchant can request a Refund API to refund a settled transaction, i.e. settled on both the issuing and acquiring bank). HSBC accepts Full Refunds and multiple Partial Refunds.

## Order Confirmation

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

# How to Connect

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

| Definition | Components |
|---|---|

| | 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

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

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

| **Sandbox** |
|---|
| https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcth-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) | |
| **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™ |
| **Where to get it?** | • Mutual agreed between Merchant and HSBC | • Mutual agreed between Merchant and HSBC |
| **Implementation** | Define in program coding, see demo in here | |

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

---

## How to Sign and Encrypt Outgoing Message

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

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

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

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

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

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

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

    return jwsObject;
}
```

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

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

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

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

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

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

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

    return jweObject;
}
```

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

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

---

## How to Decrypt Message and Verify Signature of an Incoming Message

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

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

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

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

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

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

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

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

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

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

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

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

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

---

## Summary

| Components \ Steps | Message Signing | Message Encryption | Message Decryption | Verify Signature |
|---|---|---|---|---|
| JWS Object | Signing Algorithm: `RS256` | | | |
| JWE Object | | JWE Algorithm: `RSA_OAEP_256`<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

```
#1 curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcth-ea-merchant
#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.

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

---

## Making API Request with Message Encryption

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

POST GET

```
#1 curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mcth-ea-merchant
#2  -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#3  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#4  -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
#5  -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
#6  -H "Content-Type: application/json"
#7  -d "eyJraWQiOiIwMDAxIiwiZW5jIjoiQTEyOEdDTSIsImFsZyI6IlJTQS1PQUVQLTI1NiJ9.W4nobHoVXUMOXGM5I-WC
```

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

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

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

# Data Type Overview

**Data Type Control:**

| Data Type | Allowed Characters | Definition & Important Notice |
|---|---|---|
| String | `0-9` `a-z` `A-Z`<br>***Notice:** Symbols are specifically allowed in any fields which defined with (accept Symbols)* | HSBC system will execute characters checking upon all string fields we received in order to tackle security vulnerability, such as Cross-site Scripting.<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"`. |
| 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":" "}`. |
| 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+7` (Thailand local time). Merchant is required to perform any necessary time zone conversion before submit request if needed. |
| In Response Message | `yyyy-MM-dd'T'HH:mm:ss±hh:mm` | Timezone returned in `api_gw` object is generated from HSBC API Gateway which located in Cloud and hence is calculated in `GMT+0`.<br>On the other hand, time field in `response` object will be returned together with timezone information. For more details, please read each field definition carefully. |

# FAQ

## SSL Connection Questions

### Where can I find the HSBC SSL server certificates?

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

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

## Message Encryption Questions

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

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

## Javascript Object Signing and Encryption (JOSE) Framework Questions

### Where can I get more information about JOSE Framework?

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

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

### Where can I download JOSE libraries for development?

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

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

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

# Payments

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

Payments

## Payment Page Redirect API

POST `/payment/pageRedirect`

DESCRIPTION

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

## How to do Redirection

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

```
<script language="javascript">window.onload=function(){document.pay_form.submit();}</script>
<form id="pay_form" name="pay_form" action="https://demo2.2c2p.com/2C2PFrontEnd/RedirectV3/payment" m
<input type="hidden" name="version" id="version" value="7.5">
<input type="hidden" name="merchant_id" id="merchant_id" value="344764000000003">
<input type="hidden" name="currency" id="currency" value="702">
<input type="hidden" name="result_url_1" id="result_url_1" value="http://localhost/devPortal/V3_UI_PH
<input type="hidden" name="hash_value" id="hash_value" value="ea7892d6fc4722e74b64897eadb5b86b3e91c3
<input type="hidden" name="payment_description" id="payment_description" value="1 night in Mandarin O
<input type="hidden" name="order_id" id="order_id" value="a28P4qT5oJ85">
<input type="hidden" name="amount" id="amount" value="000000002500">
<input type="hidden" name="payment_option" id="payment_option" value="A">
</form>
```

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

REQUEST BODY

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

Request Content-Types: application/json

Request Example

```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "system": {
    "default_lang": "th",
    "redirectUrl": "https://www.example.com/returnStatusFront",
    "notifyUrl": "https://www.example.com/returnStatusBack"
  },
  "payment": {
    "country": "TH",
    "currency": "THB",
    "payment_option": [
      "CC",
      "IPP",
      "123"
    ],
    "amount": 1050,
    "payment_expiry": "2018-07-16T14:10:25Z"
  },
  "merchant": {
    "merId": "JT01"
  },
  "customer": {
    "email": "customer.name@example.com"
  },
  "order": {
    "description": "1 night in Mandarin Oriental Bangkok"
  }
}
```

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "returnCode": "200",
    "returnReason": "Successful operation",
    "responseTime": "2016-11-15T10:00:00.000Z",
    "messageId": "89817674-daOO-4883",
    "sentTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "sysDatetime": "2018-01-05T15:20:45+07:00",
      "redirectLink": "<Encoded_Redirect_Submit_Form>"
    }
  }
}
```

Response Example (400 Bad Request)

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

Payments

# Payment Status Enquiry API

POST `/payment/enquiry`

DESCRIPTION

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

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

REQUEST BODY

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

Request Content-Types: application/json

Request Example

```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "merchant": {
    "merId": "JT01"
  }
}
```

RESPONSES

| | |
|---|---|
| **200 OK** enquiryRespModel | Successful operation. |

Response Content-Types: application/json

Response Example (200 OK)

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

| | |
|---|---|
| **400 Bad Request** commonRespObj | Missing or invalid Parameters. |
| **403 Forbidden** | Authorization credentials are missing or invalid. |
| **404 Not Found** | Empty resource/resource not found. |
| **500 Internal Server Error** | The request failed due to an internal error. |


```
{
  "api_gw": {
    "returnCode": "200",
    "returnReason": "Successful operation",
    "responseTime": "2016-11-15T10:00:00.000Z",
    "messageId": "89817674-daOO-4883",
    "sentTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001",
      "txnDatetime": "2018-01-05T15:20:45+07:00",
      "txnStatus": "Success",
      "txnSubStatus": "A"
    },
    "system": {
      "sysCode": "00",
      "sysMsg": "Success",
      "sysDatetime": "2018-01-05T15:20:45+07:00"
    },
    "payment": {
      "amount": 1050,
      "payment_scheme": "VI",
      "process_by": "VI"
    },
    "online_cc": {
      "approvalCode": "987012",
      "ccRefNo": "11815866",
      "maskedPan": "444411xxxxxx1111",
      "eci": "05"
    },
    "offline": {
      "apmRefNo": "11815866",
      "paidAgent": "BBL",
      "paidChannel": "BANKCOUNTER"
    },
    "ipp": {
      "ippPeriod": 4,
      "ippInterestType": "M",
      "ippInterestRate": 0.75
    },
    "refund": [
      {
        "rfdRefNo": "RFD00000123456789",
        "rfdStatus": "RF",
        "rfdAmount": 5000,
        "rfdDatetime": "2018-12-11T14:10:25+07:00"
      },
      {
        "rfdRefNo": "RFD00000123456790",
        "rfdStatus": "RR",
        "rfdAmount": 15000,
        "rfdDatetime": "2018-12-12T14:10:25+07:00"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


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


Payments

# Void API

`POST` `/payment/void`

## DESCRIPTION

This API is used to void an unsettled transaction.

## 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** optional in header | [Merchant ID]+[JWS ID]+[JWE ID] |
| **Content-Type** required in header | application/json |

## REQUEST BODY

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

Request Content-Types: application/json

Request Example


```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "merchant": {
    "merId": "JT01"
  }
}
```


## RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001",
      "txnDatetime": "2020-01-01T13:00:00+07:00"
    },
    "system": {
      "sysCode": "00",
      "sysMsg": "Success"
    },
    "void": {
      "status": "V",
      "amount": 5000,
      "voidDatetime": "2018-12-12T14:10:25+07:00"
    }
  }
}
```


Response Example (400 Bad Request)


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

Payments

# Refund API

`POST` `/payment/refund`

## DESCRIPTION

This API is used to send a refund request for a previously settled transaction. It supports both full and multiple partial refund. Before requesting a new partial refund, any prior partial refund request must have been settled.

## 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** optional in header | [Merchant ID]+[JWS ID]+[JWE ID] |
| **Content-Type** required in header | application/json |

## REQUEST BODY

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

## RESPONSES

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

Request Content-Types: application/json

Request Example

```
{
  "transaction": {
    "txnRef": "ORD-438UL748T6",
    "rfdAmount": 1050,
    "currency": "THB"
  },
  "merchant": {
    "merId": "JT01"
  }
}
```

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001",
      "txnDatetime": "2020-01-01T13:00:00+07:00"
    },
    "system": {
      "sysCode": "00",
      "sysMsg": "Success"
    },
    "refund": {
      "rfdRefNo": "RFD00000123456789",
      "rfdStatus": "RF",
      "rfdAmount": 5000,
      "rfdDatetime": "2018-12-12T14:10:25+07:00"
    }
  }
}
```

Response Example (400 Bad Request)

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

Payments

# Callback Payment Notification API

`POST` `/<Callback URL predefined by Merchant>`

## DESCRIPTION

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

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

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

> **Endpoint Definition**
> Require Merchant to provide URL endpoint and it will be pre-set at Mobile Collection backend system.

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

## REQUEST PARAMETERS

| | |
|---|---|
| **Content-Type: string** required in header | text/plain |

## REQUEST BODY

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

Request Content-Types: text/plain

Request Example

```
{
  "transaction": {
    "txnRef": "0002900F064577105001",
    "txnDatetime": "2018-01-05T15:20:45+07:00"
  },
  "system": {
    "sysDatetime": "2018-01-05T15:20:45+07:00",
    "browser_info": "Type=Firefox28,Name=Firefox,Ver=28.0"
  },
  "merchant": {
    "merId": "JT01"
  },
  "payment": {
    "amount": 1050,
    "currency": "THB",
    "payment_status": "000",
    "payment_channel": "001",
    "channel_response_code": "9062",
    "channel_response_desc": "Invalid request 3DS value",
    "payment_scheme": "PA",
    "process_by": "VI"
  },
  "online_cc": {
    "approvalCode": "987012",
    "maskedPan": "444411xxxxxx1111",
    "eci": "05"
  },
```

```
  "offline": {
    "paidAgent": "BBL",
    "paidChannel": "BANKCOUNTER"
  },
  "ipp": {
    "ippPeriod": 4,
    "ippInterestType": "M",
    "ippInterestRate": 0.75
  }
}
```

RESPONSES

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

# Schema Definitions

## commonRespObj: object

PROPERTIES

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

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

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

| Possible Value | Definition |
|---|---|
| 200 | Successful operation |
| 400 | Bad Request (With detail message in field `returnReason`) |
| 500 | Internal Error.<br><br>**Important Notices:**<br>If any tier comes before the API Cloud Foundry is unavailable, such as the API Gateway, there will be no json respond message returned.<br><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><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
Time of request received by HSBC system from client, only for HSBC internal reference use

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

Example

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

## paymentReqtModel: object

PROPERTIES

**transaction:** pay_rqt_txn_Obj required
**system:** pay_rqt_system_Obj required
**payment:** pay_rqt_payment_Obj required
**merchant:** pay_rqt_merchant_Obj required
**customer:** pay_rqt_customer_Obj optional
**order:** pay_rqt_order_Obj required

Example

```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "system": {
    "default_lang": "th",
    "redirectUrl": "https://www.example.com/returnStatusFront",
    "notifyUrl": "https://www.example.com/returnStatusBack"
  },
  "payment": {
    "country": "TH",
    "currency": "THB",
    "payment_option": [
      "CC",
      "IPP",
      "123"
    ],
    "amount": 1050,
    "payment_expiry": "2018-07-16T14:10:25Z"
  },
  "merchant": {
    "merId": "JT01"
  },
  "customer": {
    "email": "customer.name@example.com"
  },
  "order": {
    "description": "1 night in Mandarin Oriental Bangkok"
  }
}
```

## pay_rqt_txn_Obj: object

Example

PROPERTIES

**txnRef:** string range: (up to 20 chars) required
Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

Example

```
{
  "txnRef": "0002900F064577105001"
}
```

## pay_rqt_system_Obj: object

PROPERTIES

**default_lang:** string enum: [ en, id, ja, my, th, vi, zh ] range: (up to 2 chars) optional
To specify secure online payment page localization

| Possible Value | Definition |
|---|---|
| en | English (default) |
| id | Bahasa Indonesia |
| ja | Japanese |
| my | Burmese |
| th | Thai |
| vi | Vietnamese |
| zh | Simplified Chinese |

**redirectUrl:** string (accept Symbols) range: (up to 255 chars) optional
Define Frontend return url for redirecting customer back to merchant after completing the payment

**notifyUrl:** string (accept Symbols) range: (up to 255 chars) required
Define Backend return url for receiving payment result notification from HSBC after payment completed.

- This URL will also be used to notify merchant when offline payment (such as CASH payments) is completed.

Example

```
{
  "default_lang": "th",
  "redirectUrl": "https://www.example.com/returnStatusFront",
  "notifyUrl": "https://www.example.com/returnStatusBack"
}
```

## pay_rqt_payment_Obj: object

PROPERTIES

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

| Possible Value | Definition |
|---|---|
| TH | Thailand |

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

| Possible Value | Definition |
|---|---|
| THB | Thai baht |

**payment_option:** string[] optional
Restrict to show payment methods / channels in the online Payment Page. If no value is provided, by default, all available options will be shown.

- Please refer to Payment Channel Option Section.

ITEMS

string enum: [ CC, FULL, ALIPAY, LINE, PAYPAL, SSPAY, UPOP, WECHAT, 123, IPP, IMBANK, WEBPAY ]

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

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. ฿15000.00 = 1500000

**payment_expiry:** string range: (up to 20 chars) optional
To specify payment expiry date/time for APM payments (Offline Payment)

- Format: `yyyy-MM-dd'T'HH:mm:ssZ`
- Time zone is expected to be `GMT+7` (Thailand local time)

Example

```
{
  "country": "TH",
  "currency": "THB",
  "payment_option": [
    "CC",
    "IPP",
    "123"
  ],
  "amount": 1050,
  "payment_expiry": "2018-07-16T14:10:25Z"
}
```

## pay_rqt_merchant_Obj: object

PROPERTIES

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

Example

```
{
  "merId": "JT01"
}
```

## pay_rqt_customer_Obj: object

PROPERTIES

**email:** string (accept Symbols) range: (up to 150 chars) optional
Customer's email address to receive payment receipt from HSBC

Example

```
{
  "email": "customer.name@example.com"
}
```

## pay_rqt_order_Obj: object

PROPERTIES

**description:** string (accept Symbols) range: (up to 255 chars) required
Payment detail description.

Example

```
{
  "description": "1 night in Mandarin Oriental Bangkok"
}
```

## paymentRespModel: object

Example

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**transaction:** pay_rpn_txn_Obj required

**system:** pay_rpn_system_Obj required

Example


```
{
  "api_gw": {
    "returnCode": "200",
    "returnReason": "Successful operation",
    "responseTime": "2016-11-15T10:00:00.000Z",
    "messageId": "89817674-daOO-4883",
    "sentTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "sysDatetime": "2018-01-05T15:20:45+07:00",
      "redirectLink": "<Encoded_Redirect_Submit_Form>"
    }
  }
}
```


## pay_rpn_txn_Obj: object

PROPERTIES

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

Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

Example


```
{
  "txnRef": "0002900F064577105001"
}
```


## pay_rpn_system_Obj: object

PROPERTIES

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

System Return Code

| Possible Value | Definition |
|---|---|
| 000000 | Request Successful |
| 999999 | Request Failed |

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

Corresponding Text Message of Process Return Code

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

Time of sending out this response

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

**redirectLink:** string range: (up to 1024 chars) required

Encoded Redirect Link with all form submit parameters

Example


```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful",
  "sysDatetime": "2018-01-05T15:20:45+07:00",
  "redirectLink": "<Encoded_Redirect_Submit_Form>"
}
```


## enquiryReqtModel: object

PROPERTIES

**transaction:** enq_rqt_txn_Obj required

**merchant:** enq_rqt_merchant_Obj required

Example


```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "merchant": {
    "merId": "JT01"
  }
}
```


## enq_rqt_txn_Obj: object

PROPERTIES

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

Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

Example


```
{
  "txnRef": "0002900F064577105001"
}
```


## enq_rqt_merchant_Obj: object

PROPERTIES

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

Merchant ID

Example


```
{
  "merId": "JT01"
}
```


## enquiryRespModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**transaction:** enq_rpn_txn_Obj required

**system:** enq_rpn_system_Obj required

**payment:** enq_rpn_payment_Obj required

**online_cc:** enq_rpn_online_cc_Obj optional

**offline:** enq_rpn_offline_Obj optional

**ipp:** enq_rpn_ipp_Obj optional

For Installment Payment

**refund:** Array< enq_rpn_refund_Obj > optional

Example


```
{
  "api_gw": {
    "returnCode": "200",
    "returnReason": "Successful operation",
    "responseTime": "2016-11-15T10:00:00.000Z",
    "messageId": "89817674-daOO-4883",
    "sentTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001",
      "txnDatetime": "2018-01-05T15:20:45+07:00",
      "txnStatus": "Success",
      "txnSubStatus": "A"
    },
    "system": {
      "sysCode": "00",
      "sysMsg": "Success",
      "sysDatetime": "2018-01-05T15:20:45+07:00"
    },
    "payment": {
```

```
    "amount": 1050,
    "payment_scheme": "VI",
    "process_by": "VI"
  },
  "online_cc": {
    "approvalCode": "987012",
    "ccRefNo": "11815866",
    "maskedPan": "444411xxxxxx1111",
    "eci": "05"
  },
  "offline": {
    "apmRefNo": "11815866",
    "paidAgent": "BBL",
    "paidChannel": "BANKCOUNTER"
  },
  "ipp": {
    "ippPeriod": 4,
    "ippInterestType": "M",
    "ippInterestRate": 0.75
  },
  "refund": [
    {
      "rfdRefNo": "RFD00000123456789",
      "rfdStatus": "RF",
      "rfdAmount": 5000,
      "rfdDatetime": "2018-12-11T14:10:25+07:00"
    },
    {
      "rfdRefNo": "RFD00000123456790",
      "rfdStatus": "RR",
      "rfdAmount": 15000,
      "rfdDatetime": "2018-12-12T14:10:25+07:00"
    }
  ]
 }
}
```


## enq_rpn_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 20 chars) required
Returning unique Transaction Reference No.

**txnDatetime:** string range: (up to 25 chars) optional
Returning the time of receiving the corresponding transaction

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

**txnStatus:** string enum: [ Success, Failed, Settled, Refunded, Voided, Pending, Closed ] range: (up to 10 chars) required
Main Status of the transaction requested.

**txnSubStatus:** string range: (up to 3 chars) optional
Subdivided Status of the transaction requested. For reference purposes. Please see Transaction Status Code for details.

Example


```
{
  "txnRef": "0002900F064577105001",
  "txnDatetime": "2018-01-05T15:20:45+07:00",
  "txnStatus": "Success",
  "txnSubStatus": "A"
}
```


## enq_rpn_system_Obj: object

PROPERTIES

**sysCode:** string range: (up to 3 chars) required
System Result Code

- For all possible value and definition, please refer to System Result Code section

**sysMsg:** string range: (up to 100 chars) required
Corresponding Text Message of System Result Code

**sysDatetime:** string range: (up to 25 chars) required
Time of receiving the corresponding enquiry request

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

Example


```
{
  "sysCode": "00",
  "sysMsg": "Success",
  "sysDatetime": "2018-01-05T15:20:45+07:00"
}
```


## enq_rpn_payment_Obj: object

PROPERTIES

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

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. ฿15000.00 = 1500000

**payment_scheme:** string range: (up to 2 chars) required
Payment scheme code, please see Payment Scheme

**process_by:** string range: (up to 2 chars) required
Payment scheme code which process the transaction, please see Payment Scheme

Example


```
{
  "amount": 1050,
  "payment_scheme": "VI",
  "process_by": "VI"
}
```


## enq_rpn_online_cc_Obj: object

PROPERTIES

**approvalCode:** string range: (up to 6 chars) optional
Transaction Approval Code

| Payment Channel | Definition |
|---|---|
| Online Webpay | Approval code provided by from WebPay provider |
| Credit Card | Approval code provided by Credit Card Host |

**ccRefNo:** string range: (up to 30 chars) optional
Reference number

| Payment Channel | Definition |
|---|---|
| Online Webpay | Trace no. for online web payment |
| Credit Card | Trace no. for credit card payment |

**maskedPan:** string range: (up to 16 chars) optional
Masked Credit Card Number

| Payment Channel | Definition |
|---|---|
| Online Webpay | Not in use |
| Credit Card | First 6 and last 4 digits of credit card number |

**eci:** string range: (up to 2 chars) optional
ECI value

Example


```
{
  "approvalCode": "987012",
  "ccRefNo": "11815866",
  "maskedPan": "444411xxxxxx1111",
  "eci": "05"
}
```

| Payment Channel | Definition |
|---|---|
| Online Webpay | Not in use |
| Credit Card | ECI value for credit card payment, 3DSecure result code<br>• For details please see here |

## enq_rpn_offline_Obj: object

PROPERTIES

**apmRefNo:** string range: (up to 30 chars) optional
Payment Code for offline APM payment

**paidAgent:** string range: (up to 30 chars) optional
Return APM agent code that customer made payment with, please see APM Agent Code for details.

**paidChannel:** string range: (up to 30 chars) optional
Return APM Channel Code that customer made payment with, please see APM Channel Code for details.

Example


```
{
  "apmRefNo": "11815886",
  "paidAgent": "BBL",
  "paidChannel": "BANKCOUNTER"
}
```


## enq_rpn_ipp_Obj: object

PROPERTIES

**ippPeriod:** integer range: 1 ≤ x ≤ 99 optional
IPP tenor

**ippInterestType:** string enum: [ M, C ] range: (up to 1 chars) optional
IPP Interest Type

| Possible Value | Definition |
|---|---|
| M | Interest Paid by Merchant |
| C | Interest Paid by Customer |

**ippInterestRate:** number *(double)* optional
IPP interest rate

Example


```
{
  "ippPeriod": 4,
  "ippInterestType": "M",
  "ippInterestRate": 0.75
}
```


## enq_rpn_refund_Obj: object

PROPERTIES

**rfdRefNo:** string range: (up to 30 chars) optional
Refund reference number returned when refund request was completed

**rfdStatus:** string range: (up to 3 chars) optional
Refund status of the transaction requested, please see Transaction Status Code for details.

**rfdAmount:** integer range: 1 ≤ x ≤ 9999999999999 optional
Returning Refund Amount

- Refund Amount should not exceed the value of total transaction amount
- Support multiple partial refund

**rfdDatetime:** string range: (up to 25 chars) optional
Time of sending out this request

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

Example


```
{
  "rfdRefNo": "RFD00000123456789",
  "rfdStatus": "RF",
  "rfdAmount": 5000,
  "rfdDatetime": "2018-12-12T14:10:25+07:00"
}
```


## voidReqtModel: object

PROPERTIES

**transaction:** void_rqt_txn_Obj required
**merchant:** void_rqt_merchant_Obj required

Example


```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "merchant": {
    "merId": "JT01"
  }
}
```


## void_rqt_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 20 chars) required
Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

Example


```
{
  "txnRef": "0002900F064577105001"
}
```


## void_rqt_merchant_Obj: object

PROPERTIES

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

Example


```
{
  "merId": "JT01"
}
```


## voidRespModel: object

PROPERTIES

**api_gw:** commonRespObj required
**response:** object required

PROPERTIES

Example


```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
```

**transaction:** void_rpn_txn_Obj required
**system:** void_rpn_system_Obj required
**void:** void_rpn_void_Obj optional
Return if request is successful


```
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001",
      "txnDatetime": "2020-01-01T13:00:00+07:00"
    },
    "system": {
      "sysCode": "00",
      "sysMsg": "Success"
    },
    "void": {
      "status": "V",
      "amount": 5000,
      "voidDatetime": "2018-12-12T14:10:25+07:00"
    }
  }
}
```


## void_rpn_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 20 chars) required
Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

**txnDatetime:** string required
Time of the original transaction being created

Example


```
{
  "txnRef": "0002900F064577105001",
  "txnDatetime": "2020-01-01T13:00:00+07:00"
}
```


## void_rpn_system_Obj: object

PROPERTIES

**sysCode:** string range: (up to 3 chars) required
System Return Code

- For all possible value and definition, please refer to response code section

**sysMsg:** string range: (up to 100 chars) required
Corresponding Text Message of Process Return Code

Example


```
{
  "sysCode": "00",
  "sysMsg": "Success"
}
```


## void_rpn_void_Obj: object

PROPERTIES

**status:** string range: (up to 3 chars) required
Void status of the transaction requested, please see Status Code for details.

**amount:** integer range: 1 = x = 9999999999999 required
Amount of the original transaction being voided

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

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

Example


```
{
  "status": "V",
  "amount": 5000,
  "voidDatetime": "2018-12-12T14:10:25+07:00"
}
```


## refundReqtModel: object

PROPERTIES

**transaction:** refund_rqt_txn_Obj required
**merchant:** refund_rqt_merchant_Obj required

Example


```
{
  "transaction": {
    "txnRef": "ORD-438UL748T6",
    "rfdAmount": 1050,
    "currency": "THB"
  },
  "merchant": {
    "merId": "JT01"
  }
}
```


## refund_rqt_txn_Obj: object

PROPERTIES

**txnRef:** string range: (up to 20 chars) required
Pass Transaction Reference that refers to one specific transaction

**rfdAmount:** integer range: 1 = x = 9999999999999 required
Merchant provides requested Refund Amount

- Refund Amount must not exceed original Payment Amount

> ! NOTE: Do not use comma or dot. For example: Input `10000` instead of `100.00`

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

| Possible Value | Definition |
|---|---|
| THB | Thai Baht |

Example


```
{
  "txnRef": "ORD-438UL748T6",
  "rfdAmount": 1050,
  "currency": "THB"
}
```


## refund_rqt_merchant_Obj: object

PROPERTIES

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

Example


```
{
  "merId": "JT01"
}
```

## refundRespModel: object

PROPERTIES

**api_gw:** commonRespObj required

**response:** object required

PROPERTIES

**transaction:** refund_rpn_txn_Obj required

**system:** refund_rpn_system_Obj required

**refund:** refund_rpn_refund_Obj optional

Return if request is successful

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": {
    "transaction": {
      "txnRef": "0002900F064577105001",
      "txnDatetime": "2020-01-01T13:00:00+07:00"
    },
    "system": {
      "sysCode": "00",
      "sysMsg": "Success"
    },
    "refund": {
      "rfdRefNo": "RFD00000123456789",
      "rfdStatus": "RF",
      "rfdAmount": 5000,
      "rfdDatetime": "2018-12-12T14:10:25+07:00"
    }
  }
}
```


## refund_rpn_txn_Obj: object

PROPERTIES

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

Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

**txnDatetime:** string required

Time of the original transaction being created

Example


```
{
  "txnRef": "0002900F064577105001",
  "txnDatetime": "2020-01-01T13:00:00+07:00"
}
```


## refund_rpn_system_Obj: object

PROPERTIES

**sysCode:** string range: (up to 3 chars) required

System Return Code

- For all possible value and definition, please refer to response code section

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

Corresponding Text Message of Process Return Code

Example


```
{
  "sysCode": "00",
  "sysMsg": "Success"
}
```


## refund_rpn_refund_Obj: object

PROPERTIES

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

Refund reference number returned when refund request was completed

**rfdStatus:** string range: (up to 3 chars) required

Refund status of the transaction requested, please see Status Code for details.

**rfdAmount:** integer range: 1 = x = 9999999999999 required

Returning Refund Amount

- Refund Amount should not exceed the value of total transaction amount
- Support multiple partial refund

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

Time of sending out this request

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

Example


```
{
  "rfdRefNo": "RFD00000123456789",
  "rfdStatus": "RF",
  "rfdAmount": 5000,
  "rfdDatetime": "2018-12-12T14:10:25+07:00"
}
```


## statusRtnReqtModel: object

PROPERTIES

**transaction:** notif_rqt_txn_Obj required

**system:** notif_rqt_system_Obj required

**merchant:** notif_rqt_merchant_Obj required

**payment:** notif_rqt_payment_Obj required

**online_cc:** notif_rqt_online_cc_Obj optional

**offline:** notif_rqt_offline_Obj optional

**ipp:** notif_rqt_ipp_Obj optional

For Installment Payment

Example


```
{
  "transaction": {
    "txnRef": "0002900F064577105001",
    "txnDatetime": "2018-01-05T15:20:45+07:00"
  },
  "system": {
    "sysDatetime": "2018-01-05T15:20:45+07:00",
    "browser_info": "Type=Firefox28,Name=Firefox,Ver=28.0"
  },
  "merchant": {
    "merId": "JT01"
  },
  "payment": {
    "amount": 1050,
    "currency": "THB",
    "payment_status": "000",
    "payment_channel": "001",
    "channel_response_code": "9062",
    "channel_response_desc": "Invalid request 3DS value",
    "payment_scheme": "PA",
    "process_by": "VI"
  },
  "online_cc": {
    "approvalCode": "987012",
    "maskedPan": "444411xxxxxx1111",
    "eci": "05"
  },
  "offline": {
    "paidAgent": "BBL",
    "paidChannel": "BANKCOUNTER"
  },
  "ipp": {
    "ippPeriod": 4,
    "ippInterestType": "M",
    "ippInterestRate": 0.75
  }
}
```


## notif_rqt_txn_Obj: object

Example

PROPERTIES

**txnRef:** string range: (up to 20 chars) required
Unique ID referred to a specific transaction

- Required Merchant to generate a unique ID for each transaction in alphanumeric format with up to a maximum of 20 characters

**txnDatetime:** string range: (up to 25 chars) required
The time of receiving the corresponding transaction

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

Example

```
{
  "txnRef": "0002900F064577105001",
  "txnDatetime": "2018-01-05T15:20:45+07:00"
}
```

## notif_rqt_system_Obj: object

PROPERTIES

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

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

**browser_info:** string range: (up to 50 chars) optional
Client Browser information

Example

```
{
  "sysDatetime": "2018-01-05T15:20:45+07:00",
  "browser_info": "Type=Firefox28,Name=Firefox,Ver=28.0"
}
```

## notif_rqt_merchant_Obj: object

PROPERTIES

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

Example

```
{
  "merId": "JT01"
}
```

## notif_rqt_payment_Obj: object

PROPERTIES

**amount:** integer range: 1 ≤ x ≤ 99999999999 required
Returning Payment Amount of the corresponding transaction

- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. ฿15000.00 = 1500000

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

- Format: ISO 4217 Alpha (e.g. THB = Thai baht)

**payment_status:** string range: (up to 3 chars) required
Response code of a Generic Payment status, please see Payment Status Code

**payment_channel:** string range: (up to 3 chars) conditional
Code of a specific Payment channel which is chosen for the corresponding transaction, please see Payment channel code

> ! **Condition:** `payment_status` is not `003`

**channel_response_code:** string range: (up to 4 chars) required
System Response Code, please refer to System Response Code section

**channel_response_desc:** string range: (up to 255 chars) required
Description of System Response Code

**payment_scheme:** string range: (up to 2 chars) conditional
Payment scheme code, please see Payment Scheme

> ! **Condition:** `payment_status` is not `003`

**process_by:** string range: (up to 2 chars) conditional
Payment scheme code which process the transaction, please see Payment Scheme

> ! **Condition:** `payment_status` is not `003`

Example

```
{
  "amount": 1050,
  "currency": "THB",
  "payment_status": "000",
  "payment_channel": "001",
  "channel_response_code": "0062",
  "channel_response_desc": "Invalid request 3DS value",
  "payment_scheme": "PA",
  "process_by": "VI"
}
```

## notif_rqt_online_cc_Obj: object

PROPERTIES

**approvalCode:** string range: (up to 6 chars) optional
Transaction Approval Code

| Payment Channel | Definition |
|---|---|
| Online Webpay | Approval code provided by from WebPay provider |
| Credit Card | Approval code provided by Credit Card Host |

**maskedPan:** string range: (up to 16 chars) optional
Masked Credit Card Number

| Payment Channel | Definition |
|---|---|
| Online Webpay | Not in use |
| Credit Card | First 6 and last 4 digits of credit card number |

**eci:** string range: (up to 2 chars) optional
ECI value

| Payment Channel | Definition |
|---|---|
| Online Webpay | Not in use |
| Credit Card | ECI value for credit card payment, 3DSecure result code<br>• For details please see here |

Example

```
{
  "approvalCode": "987012",
  "maskedPan": "444411xxxxxx1111",
  "eci": "05"
}
```

## notif_rqt_offline_Obj: object

PROPERTIES

**paidAgent:** string range: (up to 30 chars) optional
APM Agent Code that customer made payment with, please see APM Agent Code.

**paidChannel:** string range: (up to 30 chars) optional
APM Channel Code that customer made payment with, please see APM Channel Code.

Example

```
{
  "paidAgent": "BBL",
  "paidChannel": "BANKCOUNTER"
}
```

## notif_rqt_ipp_Obj: object

PROPERTIES

**ippPeriod:** integer range: 1 ≤ x ≤ 99 optional
IPP tenor

**ippInterestType:** string enum: [ M, C ] range: (up to 1 chars) optional
IPP Interest Type

| Possible Value | Definition |
|---|---|
| M | Interest Paid by Merchant |
| C | Interest Paid by Customer |

**ippInterestRate:** number *(double)* optional
IPP interest rate

Example

```
{
  "ippPeriod": 4,
  "ippInterestType": "M",
  "ippInterestRate": 0.75
}
```

## statusRtnRespModel: object

PROPERTIES

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

Example

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

# Lifecycle of Cryptographic Keys

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

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

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

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

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

---

## Key Generation and Certificate Exchange with HSBC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

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

<Other Information>

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

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

<Other Information>

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

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

<Other Information>

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

## Certificates and Keys Maintenance

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

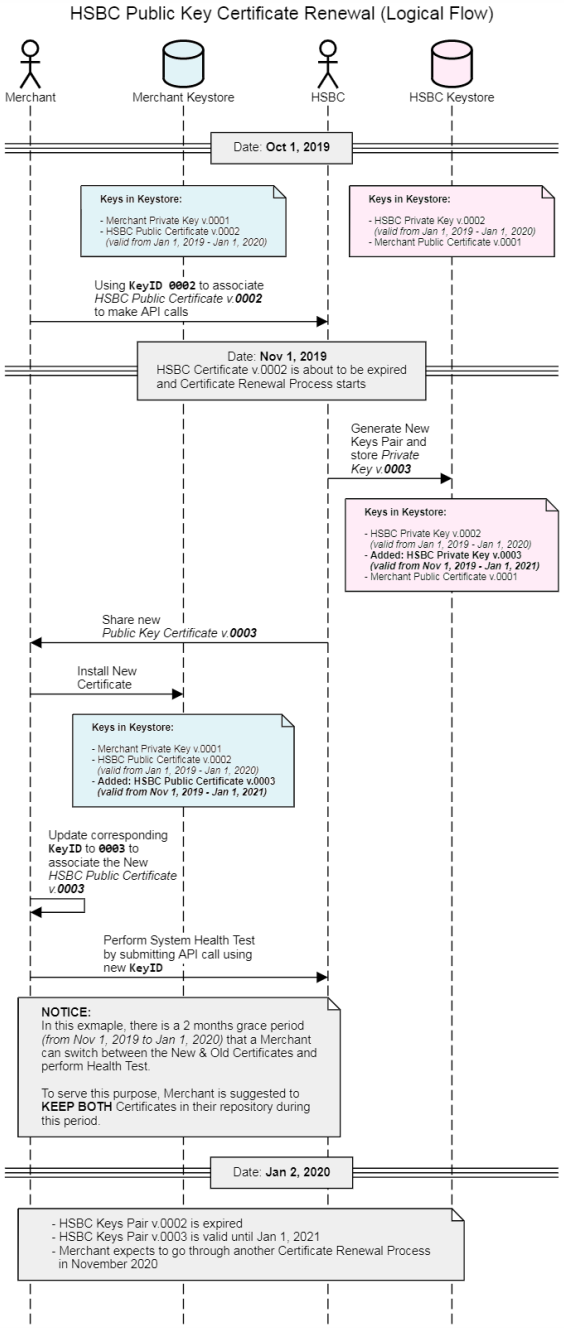
| Component | Storage | Validity |
|---|---|---|

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

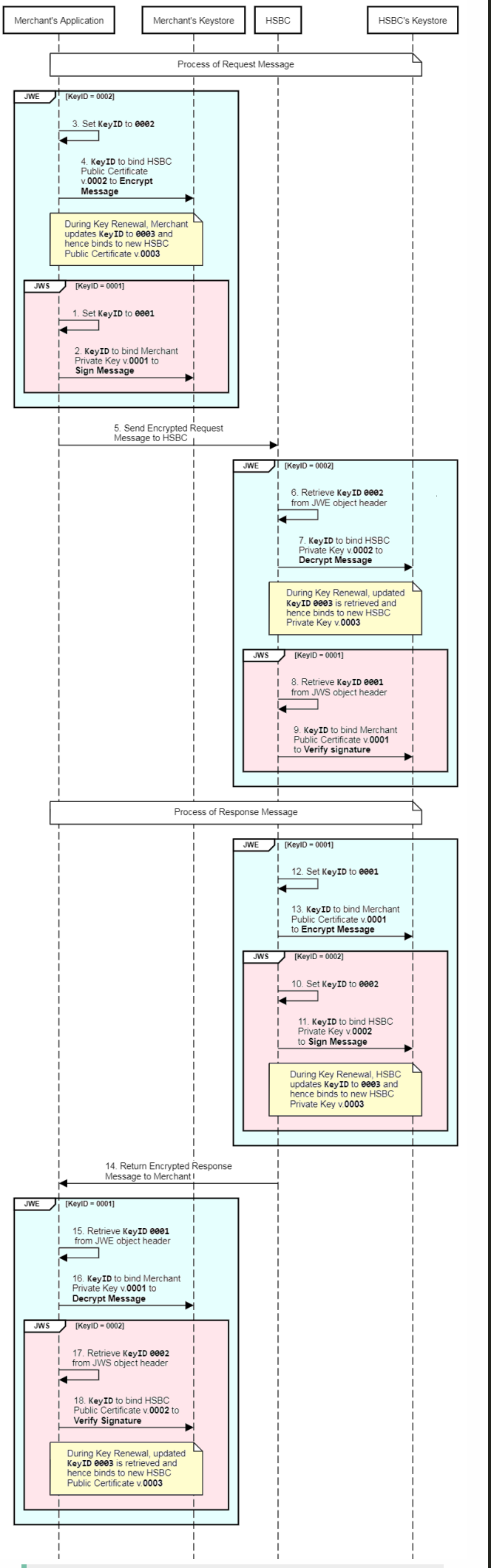
## Certificates and Keys Renewal

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.




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




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

# Payment Channel Option

| Possible Value | Definition |
|---|---|
| CC | Credit card payment |
| FULL | Full amount payment |
| ALIPAY | Alipay payment |
| LINE | LINE Pay Payment |
| PAYPAL | Paypal payment |
| SSPAY | Samsung Pay |
| UPOP | Unionpay payment |
| WECHAT | WeChat payment |
| 123 | 1-2-3 (APM) payment |
| IPP | Installment Payment Plan |
| IMBANK | Internet / Mobile banking |
| WEBPAY | Web Pay / Direct Debit |

# System Response Code

## Credit Cards / Debit Cards

| Response Code | Definition |
|---|---|
| 00 | Success. |
| 9000 | Payment Failed. |
| 9001 | unrecognized version number. |
| 9002 | authentication failed. |
| 9003 | The http request must be POST method. |
| 9004 | The invalid request. |
| 9005 | missing mandatory fields or parameters. |
| 9006 | The string length of the input parameters has exceeded more than it's specified. |
| 9007 | merchant_id is not found. |
| 9008 | The currency code is invalid or incorrect. |
| 9009 | invalid amount. |
| 9010 | invalid email format. |
| 9011 | Invalid url. |
| 9012 | The value of invoice_no is invalid. |
| 9018 | The duplicate order_id request. |
| 9019 | The current request has inconsistent parameters' value with regard to the previous request with the same order_id. |
| 9020 | Duplicate payment request. The payment has been processed before. |
| 9021 | Transaction reject: The payment is currently in process for this same transaction. |
| 9022 | transaction has expired. |
| 9023 | The credit card number can't be blank. |
| 9024 | The credit card number is invalid. |
| 9025 | The credit card expiry can't be blank. |
| 9026 | The credit card expiry date is invalid. Enter a non-expired card. |
| 9027 | The credit card expiry date is invalid. Enter a non-expired card. |
| 9028 | The credit card verification code (cvc/cvv) can't be blank. |
| 9029 | The CVV is invalid. It must be a number. |
| 9030 | The credit card holder name can't be blank. |
| 9031 | The card holder name can't be more than 50 characters. |
| 9032 | The card holder name only accept characters `-_,'.A-Za-z&` |
| 9033 | The issuing bank name can't be blank. |
| 9034 | The issuing bank name has unaccepted characters `- ~;!@#$%^&*<> {}/|:` |
| 9035 | The issuing bank name can't be more than 50 characters. |
| 9036 | The issuing bank country can't be blank. |
| 9037 | The selected issuing bank country is invalid. |
| 9038 | invalid merchant configuration. |
| 9039 | User 2 Factors (3D) authentication failed. |
| 9040 | The request is invalid. The payment_token is invalid. |
| 9041 | invalid transaction_id. |
| 9042 | Invalid hash value. |
| 9043 | Payment authorization failed. |
| 9044 | Invalid order id. |
| 9050 | MPI server unable to check. |
| 9051 | MPI server host error. |
| 9052 | The duplicate payment authorization request. |
| 9054 | Routing Failed. |
| 9055 | Session has been expired due to idle over time limit. |
| 9056 | Invalid promotion code value. |
| 9057 | Invalid payment option. |
| 9058 | Invalid IPP interest type. |
| 9059 | Invalid payment expiry. |
| 9060 | QuickPay does not exists. |
| 9061 | Stored card unique id or masked card number are invalid. |
| 9062 | Invalid request 3DS value. |

| Response Code | Definition |
|---|---|
| 9063 | Non-3DS transaction is not allowed. |
| 9064 | Invalid next charge date. |
| 9065 | Invalid recurring interval. |
| 9066 | Invalid recurring count. |
| 9067 | Invalid recurring amount. |
| 9068 | Invalid recurring accumulate amount. |
| 9069 | Invalid recurring flag. |
| 9070 | Invalid recurring accumulate flag. |
| 9071 | Invalid recurring order prefix. |
| 9072 | Invalid charge on date. |
| 9073 | Invalid next recurring charge date. |
| 9074 | Invalid Statement Descriptor Value. |
| 9079 | Stored card unique id is invalid. |
| 9080 | Merchant not allowed for tokenization. |
| 9081 | Merchant not allowed for tokenization without authorization. |
| 0034 | Fraud system reject. |
| 0035 | Payment failed. |
| 0036 | Payment is cancelled. |
| 0037 | Invalid merchant configuration or merchant is not registered. |
| 0055 | MPI reject. |
| 0062 | Corporate Bin Block. |
| 0096 | Bank Host not available. |
| 0099 | reserved error code. |

### Cash Payment / Direct Debit

| Response Code | Definition |
|---|---|
| 000 | Success (PAID) - only for WEB PAY channel |
| 001 | Success (PENDING) - for all other channels |
| 002 | Timeout |
| 003 | Invalid message |
| 004 | Invalid profile (merchant) id |
| 005 | Duplicated invoice no |
| 006 | Invalid amount |
| 007 | Insufficient balance |
| 008 | Invalid currency code |
| 009 | Payment expired |
| 010 | Payment canceled |
| 011 | Invalid payee id |
| 012 | Invalid customer id |
| 013 | Account does not exists |
| 014 | Authentication failed |
| 015 | Success (PAID) more than transaction amount (offline) - DEPRECATED |
| 016 | Success (Paid) less than transaction amount (offline) - DEPRICATED |
| 017 | Success (Paid) expired - DEPRICATED |
| 998 | Internal error |
| 999 | System error |

## System Result Code

| Possible Value | Definition |
|---|---|
| 00 | Success |
| 01 | Stored card ID cannot be found |
| 02 | Invalid Request |
| 03 | Invalid Merchant ID |
| 04 | Invalid Stored Card Unique ID |
| 05 | Invalid Customer Email |
| 10 | Missing Compulsory Values |
| 11 | Request validation failed. |
| 12 | Transaction status is not valid to perform your action. |
| 13 | Invalid hash value. |
| 14 | Invalid merchant id. |
| 15 | Invalid invoice no. |
| 16 | Requested transaction doesn't exist. |
| 17 | Request type is invalid. |
| 18 | Invalid Action Amount. |
| 21 | Void not allowed. |
| 25 | Void failed. |
| 30 | Unable to refund more than transaction amount. |
| 31 | Settlement not allowed. |
| 32 | Settlement is not required. |
| 33 | Partial settlement not allowed. |
| 34 | Settlement rejected. |
| 35 | Settlement failed. |

| Possible Value | Definition |
|---|---|
| 40 | Refund amount is more than transaction amount. |
| 41 | Refund not allowed. |
| 42 | Refund pending. |
| 43 | Partial Refund not allowed. |
| 44 | Refund rejected. |
| 45 | Refund failed. |
| 46 | Insufficient funds to perform refund. |
| 47 | Sub Merchant refund amount is more than transaction amount. |
| 48 | Sub merchant has insufficient funds to perform refund. |
| 96 | Unable to decrypt. |
| 97 | Process is not supported. |
| 98 | Request is not available |
| 99 | Unable to complete the request. |

## Transaction Status Code

| Possible Value | Definition |
|---|---|
| A | Approved. |
| AP | Approval Pending (APM). |
| AE | Approved after Expired (APM). |
| AL | Approved with less amount (APM). |
| AM | Approved with more amount (APM). |
| PF | Payment Failed. |
| AR | Authentication Rejected (MPI Reject). |
| FF | Fraud Rule Rejected. |
| IP | Rejected (Invalid Promotion). |
| ROE | Rejected (Routing Rejected). |
| RP | Refund Pending. |
| RF | Refund confirmed. |
| RR | Refund Rejected. |
| RR1 | Refund Rejected – insufficient balance. |
| RR2 | Refund Rejected – invalid bank information. |
| RR3 | Refund Rejected – bank account mismatch. |
| RS | Ready for Settlement. |
| S | Settled |
| T | Credit Adjustment |
| V | Voided / Canceled |
| VP | Void Pending |

## Payment Status Code

| Possible Value | Definition |
|---|---|
| 000 | Payment Successful |
| 002 | Payment Rejected |
| 003 | Payment was canceled by user |

## Payment Channel Code

| Possible Value | Definition |
|---|---|
| 001 | Credit and debit cards |
| 002 | Cash payment channel |
| 003 | Direct debit |
| 004 | Others |

## Payment Scheme / Process By

| Possible Value | Definition |
|---|---|
| AL | ALIPAY |
| AM | AMEX |
| AP | ALTERNATIVE PAYMENT |
| DI | DISCOVER |
| DN | DINNER |
| JC | JCB |
| KP | KCP |
| LP | LINEPAY |
| MA | MASTER CARD |
| MP | MPU |
| PA | PAYPAL |
| UP | CHINA UNION PAY |

| Possible Value | Definition |
|---|---|
| VI | VISA |
| WC | WECHAT |
| EQ | QR Gateway |
| EVI | QR Gateway - VISA |
| EMA | QR Gateway - MASTER |
| ETQ | QR Gateway - Thai QR |

## APM Agent Code

| Agent code | Agent name | ATM | Bank counter | iBanking | Webpay | Over the counter | Mobile banking | Kiosk |
|---|---|---|---|---|---|---|---|---|
| BAY | Bank of Ayudhya (Krungsri) | ✔ | ✔ | ✔ | ✔ | | ✔ | |
| BBL | Bangkok Bank | ✔ | ✔ | ✔ | ✔ | | ✔ | |
| KTB | Krung Thai Bank | ✔ | ✔ | ✔ | ✔ | | ✔ | |
| SCB | Siam Commercial Bank | ✔ | ✔ | ✔ | ✔ | | ✔ | |
| TTB | TTB Bank | ✔ | ✔ | ✔ | ✔ | | | |
| UOB | United Overseas Bank | ✔ | ✔ | ✔ | ✔ | | | |
| KBANK | Kasikorn Bank | ✔ | ✔ | ✔ | | | | |
| CIMB | CIMB Thai Bank | ✔ | ✔ | ✔ | | | | |
| BIGC | Big C Supercenter | | | | | ✔ | | |
| MPAY | mPay Station by AIS | | | | | ✔ | | |
| PAYATPOST | Pay@Post by Thailandpost | | | | | ✔ | | |
| TESCO | Tesco Lotus Counter Service | | | | | ✔ | | |
| TRUEMONEY | True Money Shop | | | | | ✔ | | |
| CENPAY | CenPay by Central | | | | | ✔ | | |
| BOONTERM | Boonterm | | | | | | | ✔ |

## APM Channel Code

| Possible Value (Used in Status Enquiry) | Possible Value (Used in Payment Notification) | Definition |
|---|---|---|
| ATM | ATM | ATM Machine |
| ATM | ATM | Kiosk Machines |
| OTC | OVERTHECOUNTER | Cash Over the Counter (BANK) |
| OTC | OVERTHECOUNTER | Cash Over the Counter (NON-BANK) |
| IMB | IBANKING | Internet Banking |
| IMB | MOBILEBANKING | Mobile Banking |
| DDB | WEBPAY | Web Payment |

## Download Swagger

Click here to download Swagger 2.0 file in YAML format.

## Disclaimer

***IMPORTANT NOTICE***