API Specification for Indonesia Cards and Alternate Payment Methods

Version: 2.3

Description

This document introduces the **OpenAPI specification** describing the REST APIs of the HSBC ASP Mobile Collection for Retail Payments pertaining to Indonesia Cards and Alternate Payments.

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

Update Log

- [Dec 28, 2021] **v2.3** Revised several content sections
- [Nov 2, 2020] **v2.2** Renewed API Retrieve a transaction and Void or Refund a Payment

> **NOTICE:**
> The two APIs in former version (v.2.1) are still usable.

- [Aug 28, 2020] **v2.1**
  - Added `plan_id` for Instalment support
  - Added System Response Code pay_rpn_system_Obj and enq_rpn_sys_Obj
  - Added Possible Values in `payment_option`
- [Jun 22, 2020] **v2.0** Enhanced API Access Authorization, details in How to Connect
- [Nov 27, 2019] **v1.10** Added conditional object in response messages of Enquiry API and Callback Notification
- [Nov 8, 2019] **v1.9** Updated `API Base URL` including both Sandbox and Production
- [Oct 17, 2019] **v1.8** Updated Possible Value of Payment Channel Code
- [Sep 23, 2019] **v1.7**
  - Updated `Disclaimer`
  - Enhanced Section `API Connectivity`
  - Enhanced Section `REFERENCE`
- [Feb 13, 2019] **v1.6**
  - Added optional field `minAmt` `maxAmt` at direct_va_rpn_payment_Obj.
  - Enhanced topic `How to generate a Payment Code`
- [Jan 31, 2019] **v1.5** Removed possible value `03 - Partial Refund Debit` of field `refund_type` at cancel_rqt_payment_Obj.
- [Dec 6, 2018] **v1.4**
  - Moved field `issuing_bank` from object `credit_card` to object `payment`
  - Added new field `wallet_id`
- [Nov 22, 2018] **v1.3** Updated the field definition of field `email` in Page Redirect & Direct VA enquiry API
- [Nov 20, 2018] **v1.2** Added response field `notifyUrl` in Direct VA Enquiry API
- [Nov 19, 2018] **v1.1**
  - Added Direct VA in `Channels and Features`
  - Added new API `Direct VA Enquiry API`
- [Oct 16, 2018] **v1.0**
  - Initial Version for Merchant Distribution
  - Updated the retry logic of Status Notification API
  - Updated Possible Value of Payment Channel Code
  - Added Order Confirmation Section
- [Sep 21, 2018] **v9.0e** Added request field `notifyUrl` in Page Redirect API
- [Sep 10, 2018] **v9.0d** Added request field `customerId` in Page Redirect API
- [Sep 4, 2018] **v9.0c**
  - Removed `recurring` object in request of Page Redirect API
  - Removed possible value `17` in field `payment_option`
  - Modified `order` object to array in request of Page Redirect API
  - Removed field `status_type` in response of Enquiry API
  - Changed format of field `payment_datetime`
  - Modified possible value in field `liability` in Enquiry response
- [Aug 2, 2018] **v9.0b** 1st Revision, enhanced & update field definition of all APIs
- [Jul 19, 2018] **v9.0a** 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

Credit Card

HSBC Mobile Collection accept all credit card issued with these Principals:

- Mastercard
- Visa
- JCB (Only for BNI Acquiring)

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

BIN Filtering

Each card issuer has a BIN (Bank Identification Number) made up of the first 6 digits of the Credit Card number. The conditions set in the filter specify which BIN numbers are allowed to make payments on your site. When a card number blocked by the BIN filter is entered, HSBC's backend server will not process the payment.

Tokenization

Tokenization enables the customer to make a purchase without having to input card details or personal information, apart from the CVV number. By reducing the number of fields the customer needs to fill in, tokenisation enables merchants who have repeat customers - to benefit from a faster checkout. If the card issuer requires 3D secure verification process, the customer still needs to complete the form to make a purchase.

For this process to work, the customer enters all the card information the first time they make a purchase. This card information is stored securely and a token is given to the merchant paired to the customer's login credentials on the merchant website. After this process has been completed, each time they make a payment, they only have to input the CVV.

Instalment

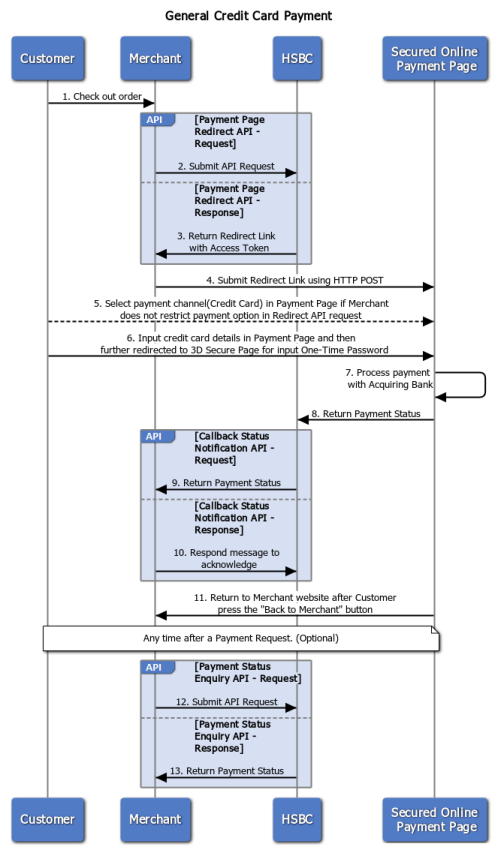
To make an instalment payment, retrieve a `plan_id` from Plans and input it as a request field of the Payment Page Redirect API.

Cancellation

The Payment Cancellation API modifies the status of a successful credit card transaction:

Depending upon the settlement status, the Payment gateway submits either a Void or Refund request to a successful transaction. If the transaction is unsettled, a Void is submitted. If the transaction is settled, a Refund is submitted.

API Use Case (General Payment)



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 field `redirectLink`. The redirect link is inside a `HTML FORM POST`, for details, please refer to the Payment Page Redirect API.
4. The Merchant submits the redirect link using `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> The Merchant can restrict the payment page to show only the Credit Card payment by passing a value `15` in the optional field `payment_option`.

5. If the Merchant does not pass `payment_option` in a Redirect API request, the Customer can select different payment channels.
6. The Customer inputs Credit Card details in the Payment Page and redirects to a 3D Secure (3DS) Page for input One-Time password.
7. The Payment page connects securely to the bank's backend systems to process the payment.
8. HSBC receives payment status once it is updated from the backend system.
9. HSBC triggers the Callback Payment Notification API and sends the payment status back to the Merchant.

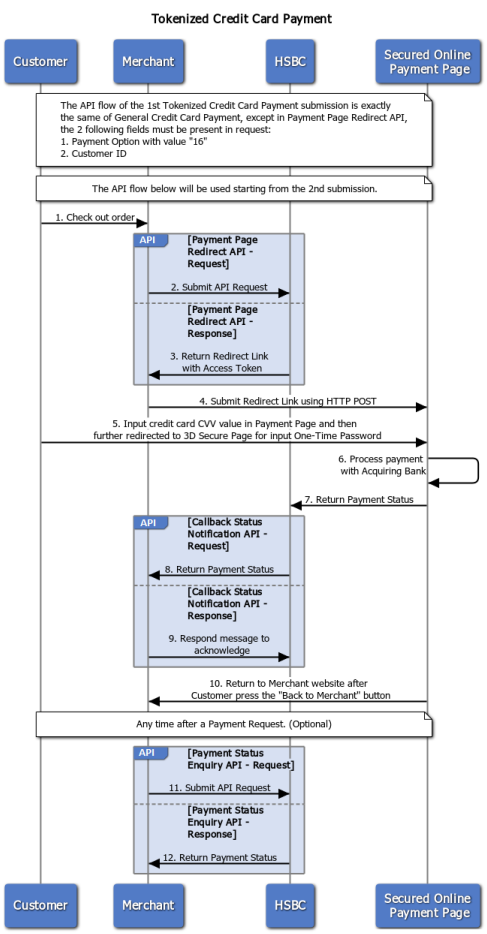
> **NOTE:**
> The Merchant can define the URL to catch the Notification in request field `notifyUrl` in the Payment Page Redirect API.

10. The Merchant responds to acknowledge. Returning different response messages can trigger a different scenario, for details, please see the Callback Payment Notification API.
11. Redirects back to merchant website once the customer presses the "Back to Merchant" button in the payment completion page.

> **NOTE:**
> The Merchant is required to pre-set a URL endpoint to catch this redirection, for more details, please contact HSBC.

12. The Merchant can use a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant receives no acknowledgement after a certain period of time.
13. According to the transaction reference number the Merchant provided, HSBC returns the latest payment status.

API Use Case (Payment with Tokenization)



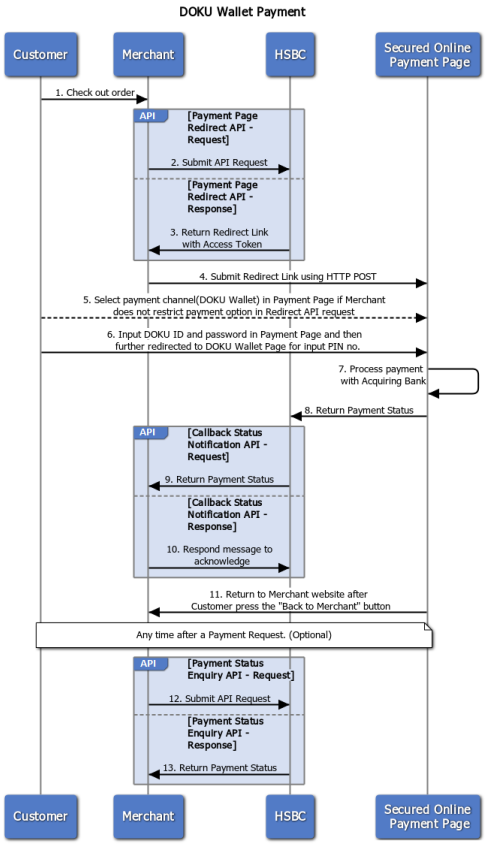
The API Flow for a General & Tokenized Credit Card payment is similar. However, to facilitate the tokenization feature, an optional field `payment_option` (with value ***16***) and `customerId` must always be passed in the Page Redirect API Request message.

The credit card information is stored securely in the backend system and is associated with the `customerId` during the first payment submission. Starting from the second submission, `customerId` is used to retrieve the corresponding credit card information and subsequently the Customer only needs to input the CVV value.

DOKU Wallet

DOKU Wallet is an electronic money service issued by DOKU - an e-payment solution provider. DOKU has a license to issue e-money granted from the Bank of Indonesia.

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 field `redirectLink`. The redirect link is inside a `HTML FORM POST`, for details, please refer to the Payment Page Redirect API.
4. The Merchant submits the redirect link using `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> The Merchant can restrict the payment page to show only DOKU Wallet payment by passing value `04` in the optional field `payment_option`.

5. If the Merchant does not pass `payment_option` in a Redirect API request, the Customer can select different payment channels.
6. The Customer inputs a DOKU ID and password in the Payment Page and redirects to a DOKU Wallet Page to input the DOKU PIN number.
7. The Payment page connects securely to the bank's backend systems to process the payment.
8. HSBC receives payment status once it is updated from the backend system.
9. HSBC triggers the 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 acknowledge. Returning different response messages can trigger a different scenario, for details, please see the Callback Payment Notification API.
11. Redirects back to merchant website once the customer presses the "Back to Merchant" button in the payment completion page.

> **NOTE:**
> The Merchant is required to pre-set a URL endpoint to catch this redirection, for more details, please contact HSBC.

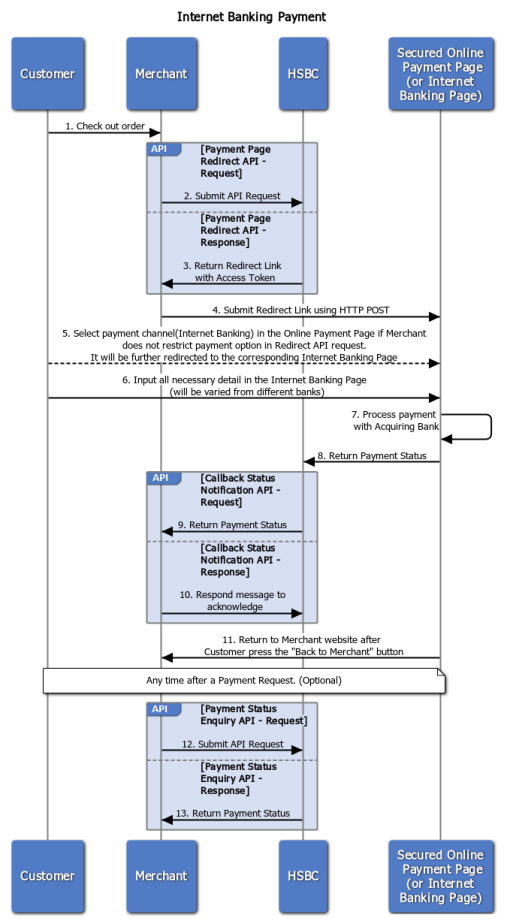
12. The Merchant can use a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant receives no acknowledgement after a certain period of time.
13. According to the transaction reference number the Merchant provided, HSBC returns the latest payment status.

Internet Banking

Mobile collection accepts both bank account debit payment and consumer financing payments. This is categorized as an internet banking payment. Below is the list of internet banking channels:

Internet Banking Channel	Corresponding Payment Option Value
CIMB Clicks	19
Muamalat Internet Banking	25
Danamon Internet Banking	26
Permata Internet Banking	28

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 field `redirectLink`. The redirect link is inside a `HTML FORM POST`, for details, please refer to the Payment Page Redirect API.
4. The Merchant submits the redirect link using `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> The Merchant can give the Customer access to the Internet Banking page directly by passing the Payment Option values in the request message.

5. If the Merchant does not pass `payment_option` in a Redirect API request, the Customer can select different payment channels.

6. The Customer inputs all necessary details in the Internet Banking Page (Bank's website).
7. The Payment is then processed in the bank's website.
8. HSBC receives payment status once it is updated from the backend system.
9. HSBC triggers the 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 acknowledge. Returning different response messages can trigger a different scenario, for details, please see the Callback Payment Notification API.
11. Redirects back to merchant website once the customer presses the "Back to Merchant" button in the payment completion page.

> **NOTE:**
> The Merchant is required to pre-set a URL endpoint to catch this redirection, for more details, please contact HSBC.

12. The Merchant can use a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant receives no acknowledgement after a certain period of time.
13. According to the transaction reference number the Merchant provided, HSBC returns the latest payment status.

Virtual Account

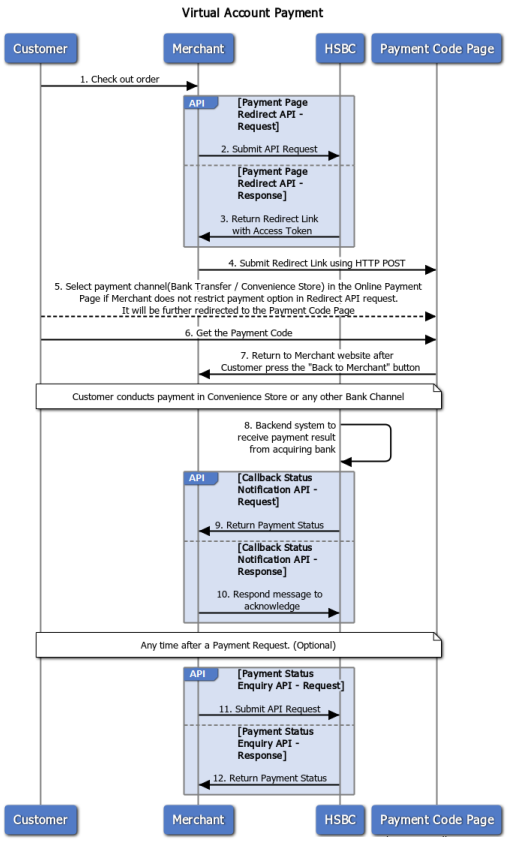
The Virtual Account Payment Code is issued to banks or convenience stores for payments. Payment is initiated from bank's or convenience store's channels such as ATM, mobile banking, internet banking or over the counter / teller. Supported VA Channels are listed below:

VA Channel	Corresponding Payment Option Value
Convenience Store - Indomaret	31
CIMB VA	32
Danamon VA	33
Alfa VA	35
Permata VA	36
Mandiri VA	41
Maybank VA	44

There are two methods of creating a Virtual Account Payment Code:

- **Regular VA** - This is the most common use case where the Merchant can retrieve the Payment Code in the online payment page.
- **Direct VA** - This is an alternative flow. The Merchant is responsible to create the Payment Code.

API Use Case - Regular VA



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 field `redirectLink`. The redirect link is inside a `HTML FORM POST`, for details, please refer to the Payment Page Redirect API.
4. The Merchant submits the redirect link using `HTML FORM POST`. It redirects the Merchant website to the Secure Online Payment Page.

> **NOTE:**
> The Merchant can give the Customer access to the Internet Banking page directly by passing the Payment Option values in the request message.

5. If the Merchant does not pass `payment_option` in a Redirect API request, the Customer can select different payment channels.
6. The Customer gets the payment Code.
7. Redirects back to merchant website once the customer presses the "Back to Merchant" button in the payment completion page.

> **NOTE:**
> The Merchant is required to pre-set a URL endpoint to catch this redirection, for more details, please contact HSBC.

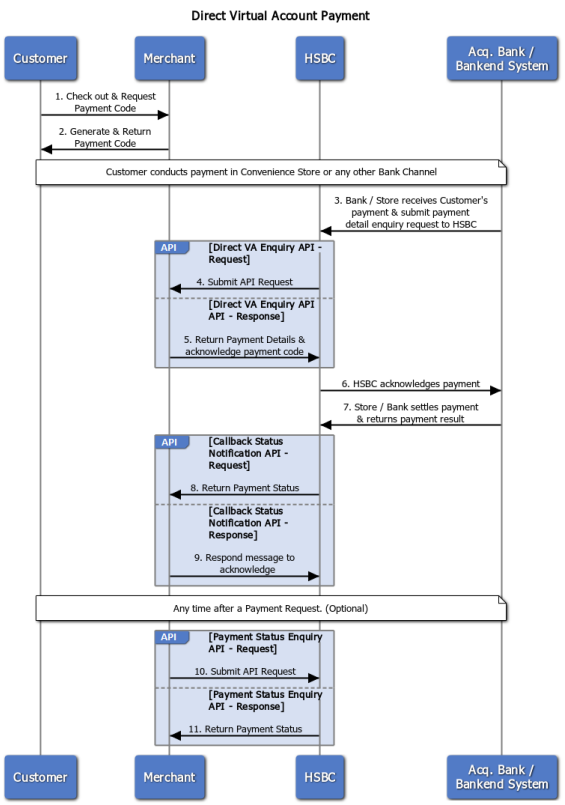
8. HSBC's backend system receives the payment status once the payment process is completed at the acquiring bank.

9. HSBC triggers the 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 uses the API to acknowledge. Returning different response messages can trigger a different scenario, for details, please see the Callback Payment Notification API.
11. The Merchant can use a Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant receives no acknowledgement after a certain period of time.
12. According to the transaction reference number the Merchant provided, HSBC returns the latest payment status.

API Use Case - Direct VA



1. The Customer conducts a checkout process in merchant's website.
2. The Merchant generates a Payment Code and delivers it to the Customer.

> **NOTE:**
> To generate a Payment Code, the pattern must be followed by a set of rules, please see the subsequent section for details.

3. The Customer makes a payment in Convenience Stores / Bank Channels (ATM, Internet Banking, Mobile Banking, Tekker), then a Store / Bank will submit a payment detail enquiry request to HSBC.
4. HSBC then triggers a Direct VA Enquiry API to the Merchant.

> **NOTE:**
> The Merchant is required to pre-set a URL endpoint to catch this API request, for details, please contact HSBC.

5. The Merchant provides payment details as well as an acknowledge payment code by responding using the Direct VA Enquiry API.
6. HSBC relays the acknowledgement to the acquiring bank.
7. The Store / Bank settles the payment and returns a payment completion status to HSBC.
8. HSBC then triggers Callback Payment Notification API and send payment status back to the Merchant.

> **NOTE:**
> The Merchant can define the URL to catch the Notification in response field `notifyUrl` in Direct VA Enquiry API.

9. The Merchant responds using the API to acknowledge. Returning different response messages can trigger a different scenario, for details, please see the Callback Payment Notification API.
10. The Merchant can optionally submit Payment Status Enquiry API at any time after a payment request is submitted. This is useful when the Merchant does not receive an acknowledge after a certain period of time.
11. According to the transaction reference number the Merchant provided, HSBC returns the latest payment status.

How to generate a Payment Code

```
<5 digit acquiring bank code> + <3 digit merchant identity code> + <8 digit merchant-generated random
e.g. 88560691000000123
```

- **5 digit Acquiring Bank Code** has a constant value for different banks / stores (except Indomaret)
- **3 digit Merchant Identity Code** will be different from merchants and is assigned by HSBC
- **8 digit Random Code** is generated by Merchant

Sample of Payment Codes:

Bank / Store	5 digit Acquiring Bank Code	First 8 digit used for Testing
Permata Bank	88560	88560691
Mandiri Bank	88899	88899274
CIMB Bank	51491	51491062
Bank Danamon	89220	89220038
Alfa Group	88888	88888173
Indomaret	*Assigned by HSBC*	83311000

Check Status Feature

Mobile collections have a feature the merchant can use to check the status of every payment transaction. To implement Check Status, please see the Payment Status Enquiry API.

Order Confirmation

In the above 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, where the details can be retrieved from the asynchronous Callback Payment Notification API.

How to Connect

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

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

API Gateway URL

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

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

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

API Authentication

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

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

User Identification

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

Connection Security

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

Message Security - Data Encryption and Signing

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

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

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

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

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

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

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

How to Sign and Encrypt Outgoing Message

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

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

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

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

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

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

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

    return jwsObject;
}
```

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

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

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

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

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

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

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

    return jweObject;
}
```

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

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

How to Decrypt Message and Verify Signature of an Incoming Message

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

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

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

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

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

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

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

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

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

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

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

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

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

Summary

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

How to Make an API Request

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

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

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

Make Your API Request with Plain Messages

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

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

Step 1. Run this command on your platform:

POST GET

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

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` and symbols `, . & ; + : / =` Notice: `@` is specifically allowed in email field	HSBC system will execute characters checking upon all string fields we received in order to tackle security vulnerability, such as Cross-site Scripting. 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. Field & value must be present in the request with valid `JSON` format.
Optional	Annotated with `optional` tag in field definition section. If you don't want to pass fields that are optional, your handler should not pass neither empty strings `{"example":""}` nor blank value `{"example":" "}`.
Conditional	Annotated with `conditional` tag in field definition section. Required under a specific condition whose logic is always provided in the field definition if it is a Conditional Field.

Time Zone Control:

Aspect	Format	Definition & Important Notice
In Request Message	`yyyy-MM-dd'T'HH:mm:ssZ`	Time zone is expected to be `GMT+7` (Jakarta 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:ssxhh:mm`	Timezone returned in `api_gw` object is generated from HSBC API Gateway which located in Cloud and hence is calculated in `GMT+0`. On the other hand, time field in `response` object will be returned together with timezone information. For more details, please read each field definition carefully.

FAQ

SSL Connection Questions

Where can I find the HSBC SSL server certificates?

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

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

Message Encryption Questions

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

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

Javascript Object Signing and Encryption (JOSE) Framework Questions

Where can I get more information about JOSE Framework?

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

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

Where can I download JOSE libraries for development?

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

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

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

Payments

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

Payments

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://staging.doku.com/Suite/Receive" method="post">
<input type="hidden" name="CURRENCY" id="CURRENCY" value="360">
<input type="hidden" name="PAYMENTCHANNEL" id="PAYMENTCHANNEL" value="15">
<input type="hidden" name="SESSIONID" id="SESSIONID" value="NA">
<input type="hidden" name="WORDS" id="WORDS" value="ef6e76f5d55925cbc3bf2240e69446cd745eff25792ce8914
<input type="hidden" name="CHAINMERCHANT" id="CHAINMERCHANT" value="NA">
/* More Input Fields Here... */
</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",
    "plan_id": "BCA-BCA-3",
    "txnDatetime": "2018-06-11T14:10:25Z",
    "notifyUrl": "https://www.your-domain.com/notification"
  },
  "payment": {
    "country": "ID",
    "currency": "IDR",
    "payment_option": "15",
    "amount": 1234050
  },
  "merchant": {
    "merId": "5822"
  },
  "customer": {
    "name": "Tuanku Imam Bonjol",
    "email": "Tuanku.Imam.Bonjol@customer.com",
    "customerId": "CUST0045678"
  },
  "order": {
    "description": [
      {
        "orderName": "Order Item 1",
        "unitAmt": 1500000,
        "unit": 2,
        "subAmt": 3000000
      },
      {
        "orderName": "Order Item 2",
        "unitAmt": 2400000,
        "unit": 3,
        "subAmt": 7200000
      }
    ]
  },
  "other": {
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU.."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "redirectLink": "<Redirection_HTML_Form>"
    }
  }
}
```

Response Example (400 Bad Request)

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

Payments

Retrieve a transaction by Transaction Reference

GET /payment/transaction/{txnRef}

DESCRIPTION

Merchant can optionally initiate payment status enquiry at any time after a payment request is submitted. This is used when Merchant wants to check payment status any time after a payment request or find no acknowledge message returned after a certain period of time. HSBC Mobile Collection will return the latest transaction status according to the transaction reference number Merchant 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 required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json

txnRef: string
required
in path

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

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "transaction": {
      "txnRef": "0002900F064577105001",
      "plan_id": "BCA-BCA-3",
      "result_message": "SUCCESS",
      "response_code": "0000"
    },
    "payment": {
      "amount": 1050,
      "currency": "IDR",
      "payment_option": "15",
      "payment_datetime": "2018-06-11T14:10:25+07:00",
      "approval_code": "987021",
      "issuing_bank": "BCA",
      "payment_code": "12345678",
      "wallet_id": "dokuwalletid12345"
    },
    "credit_card": {
      "cardholder_name": "Tuanku Imam Bonjol",
      "brand": "VISA",
      "mcn": "444411xxxxxx1111",
      "3ds_status": "TRUE",
      "liability": "CUSTOMER"
    },
    "refunds": [
      {
        "refundRef": "REF2900F064577105001",
        "amount": 100,
        "bank_refno": "987654",
        "response_code": "0000",
        "response_message": "REFUNDED",
        "refundDatetime": "2018-12-12T14:10:25+07:00"
      }
    ],
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ],
    "links": [
      {
        "href": "/plan/@id",
        "id": "BCA-BCA-3",
        "rel": "plan",
        "method": "GET"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


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


Payments

Void or Refund a Payment

POST `/payment/refund`

DESCRIPTION

This API allows Merchant to cancel Credit Card Sale transaction regarding of its settlement status by performing either a void or refund request.

REQUEST PARAMETERS

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

REQUEST BODY

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

RESPONSES

200 OK	cancelRespModel	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": "0002900F064577105001"
  },
  "refund": {
    "refundRef": "REF2900F064577105001",
    "amount": 100,
    "reason": "Goods are damaged"
  }
}
```


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"
    },
    "refund": {
      "refundRef": "REF2900F064577105001",
      "amount": 100,
      "bank_refno": "987654",
      "response_code": "0000",
      "response_message": "REFUNDED"
    }
  }
}
```


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>`

```
e.g. https://www.your-domain.com/notification
```

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. Mobile Collection will trigger this API call and defines the interface with OpenAPI standard. Merchant is required to provide implementation.

> **Retry Mechanism**
> If no success response is received, up to 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 can be pre-set at Mobile Collection backend system or define it in field `notifyUrl` in Payment Page Redirect API.

> **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",
    "plan_id": "BCA-BCA-3",
    "result_message": "SUCCESS",
    "response_code": "0000",
    "status_type": "P"
  },
  "merchant": {
    "merId": "5822"
  },
  "payment": {
    "amount": 1050,
    "payment_option": "15",
    "currency": "IDR",
    "payment_datetime": "2018-06-11T14:10:25+07:00",
    "approval_code": "987021",
    "issuing_bank": "BCA"
  },
  "credit_card": {
    "cardholder_name": "Tuanku Imam Bonjol",
    "brand": "VISA",
    "mcn": "444411xxxxxx1111",
    "3ds_status": "TRUE",
    "liability": "CUSTOMER"
  },
  "va": {
    "payment_code": "8856069112345678"
  },
  "other": {
    "wallet_id": "dokuwalletid12345",
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```

RESPONSES

200 OK — Successful operation.

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

Response Content-Types: application/json

Response Example (200 OK)

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

Alternative Payments

Contains specific resource collections for alternative payment flow

Alternative Payments

Direct VA Enquiry API

POST `/<Callback URL predefined by Merchant>`

```
e.g. https://www.your-domain.com/direct-va/enquiry
```

DESCRIPTION

This is a Callback API. This API is used for Direct Virtual Account payment where the payment code is generated by Merchant. Merchant is required to provide payment details as well as acknowledge payment code by responding this API.

REQUEST PARAMETERS

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

REQUEST BODY

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

Request Content-Types: text/plain

Request Example

```
{
  "merchant": {
    "merId": "5822"
  },
  "payment": {
    "payment_option": "36",
    "payment_code": "8856069112345678"
  }
}
```

RESPONSES

200 OK directVaEnqRespModel	Successful operation. *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)


```
{
  "transaction": {
    "txnRef": "0002900F064577105001",
    "txnDatetime": "2018-06-11T14:10:25Z",
    "txnResponseCode": "0000",
    "notifyUrl": "https://www.your-domain.com/notification"
  },
  "payment": {
    "country": "ID",
    "currency": "IDR",
    "amount": 0,
    "minAmt": 1000000,
    "maxAmt": 2000000,
    "payment_code": "8856069112345678"
  },
  "customer": {
    "name": "Tuanku Imam Bonjol",
    "email": "Tuanku.Imam.Bonjol@customer.com"
  },
  "order": {
    "description": [
      {
        "orderName": "Order Item 1",
        "unitAmt": 1500000,
        "unit": 2,
        "subAmt": 3000000
      },
      {
        "orderName": "Order Item 2",
        "unitAmt": 2400000,
        "unit": 3,
        "subAmt": 7200000
      }
    ]
  }
}
```


Plans

A plan is a foundation on which an instalment payment *(or a recurring payment which will be supported later)* is built. It acts as a reusable template and contains details of the billing information.

To make an instalment or a recurring payment, you must invoke the `plan_id` which represent a specific instalment or recurring plan into the Payment Page Redirect API.

Plans

Retrieve All Plans

`GET` `/plan`

DESCRIPTION

Use this endpoint to fetch all plans or filter the result by additional parameters.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type: string required in header	application/json
total_count: integer in query	*Example: /plan?total_count=12*
customer_bank: string in query	*Example: /plan?customer_bank=BCA*
acquiring_bank: string in query	*Example: /plan?acquiring_bank=BCA*

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "plans": [
      {
        "id": "BCA-BCA-3",
        "type": "I",
        "description": "Monthly Installment Plan #1",
        "period": "month",
        "interval": 1,
        "total_count": 6,
        "customer_bank": "BCA",
        "acquiring_bank": "BCA",
        "create_date": "2020-01-01T13:02:00+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"
}
```


Plans

Retrieve Plan by Plan ID

`GET` `/plan/{plan_id}`

DESCRIPTION

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

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type: string required in header	application/json
plan_id: string required in path	*Data Encryption is enforced.*

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "plans": [
      {
        "id": "BCA-BCA-3",
        "type": "T",
        "description": "Monthly Installment Plan #1",
        "period": "month",
        "interval": 1,
        "total_count": 6,
        "customer_bank": "BCA",
        "acquiring_bank": "BCA",
        "create_date": "2020-01-01T13:02:00+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"
}
```


Schema Definitions

commonRespObj: object

PROPERTIES

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

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

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

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

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

Corr. Return Code	Return Message Sample	Definition
200	Successful operation	A successful API operation in terms of Authorization, Connectivity and valid JSON Message Structure. Any checking failure on Business Logic level will be still considered a successful API operation yet the Business Logic checking result will be returned in `response` object.
400	Client ID - Merchant ID mapping is not correct/updated!	The binding of Client ID, Merchant ID and Merchant Public Certificate is incorrect or not up-to-date.
400	object has missing required properties `field name`	Fail to pass JSON Field Mandatory Check.
400	instance type `data type` does not match any allowed primitive type	Fail to pass JSON Field Type Check.
400	string `field value` is too long	Fail to pass JSON Field Max Length Check
400	instance failed to match at least one required schema among `no. of conditional field`	Fail to pass JSON Conditional Field Check.

Example


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

Corr. Return Code	Return Message Sample	Definition
500	java.net.ConnectException: Connection refused: connect	**Notices:** Message can be varied depended on the dependent system *(which across the entire system pipeline)* which returns this message. Yet, all reasons can be concluded into Internal Error or System Unavailable.

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

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

paymentReqtModel: object

PROPERTIES

transaction: pay_rqt_txn_Obj required
payment: pay_rqt_payment_Obj required
merchant: pay_rqt_merchant_Obj required
customer: pay_rqt_customer_Obj required
order: pay_rqt_order_Obj required
other: pay_rqt_other_Obj optional

Example


```
{
  "transaction": {
    "txnRef": "0002900F064577105001",
    "plan_id": "BCA-BCA-3",
    "txnDatetime": "2018-06-11T14:10:25Z",
    "notifyUrl": "https://www.your-domain.com/notification"
  },
  "payment": {
    "country": "ID",
    "currency": "IDR",
    "payment_option": "15",
    "amount": 1234050
  },
  "merchant": {
    "merId": "5822"
  },
  "customer": {
    "name": "Tuanku Imam Bonjol",
    "email": "Tuanku.Imam.Bonjol@customer.com",
    "customerId": "CUST0045678"
  },
  "order": {
    "description": [
      {
        "orderName": "Order Item 1",
        "unitAmt": 1500000,
        "unit": 2,
        "subAmt": 3000000
      },
      {
        "orderName": "Order Item 2",
        "unitAmt": 2400000,
        "unit": 3,
        "subAmt": 7200000
      }
    ]
  },
  "other": {
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```


pay_rqt_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 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 30 characters
- A uniqueness checking will be taken place based on EACH `merId`

plan_id: string range: (up to 100 chars) optional
The entity ID of a Plan. Having this ID means the corresponding payment request is an instalment payment.

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

- Client system time. The timezone is expected to be `GMT+7` (Jakarta local time). Merchant is required to perform timezone conversion if needed. Format: `yyyy-MM-dd'T'HH:mm:ssZ`

notifyUrl: string range: (up to 255 chars) required
Merchant's URL to receive Status Notification sent by HSBC

- Merchant is required to provide implementation in order to process the JSON message, please see Status Notification API for details.

Example


```
{
  "txnRef": "0002900F064577105001",
  "plan_id": "BCA-BCA-3",
  "txnDatetime": "2018-06-11T14:10:25Z",
  "notifyUrl": "https://www.your-domain.com/notification"
}
```


pay_rqt_payment_Obj: object

PROPERTIES

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

Possible Value	Definition
ID	Indonesia

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

Possible Value	Definition
IDR	Indonesian Rupiah

payment_option: string enum: [04, 15, 16, 19, 22, 25, 26, 28, 31, 32, 33, 34, 35, 36, 37, 38, 41, 44, 50, 51, 53] range: (up to 2 chars) optional
Merchant to pass which payment option is displayed in the redirected Payment Page

- Merchant can pass the following value to restrict the payment option displayed in the Online Payment Page, otherwise, all payment options will be shown in the Online Payment Page.
- Please see Payment Channel Code for details.

amount: integer range: 1 ≤ x ≤ 99999999999999 required
Payment Amount

- Format: Eliminate punctuation and sign, support 2 decimal places, e.g. Rp 12340.50 = 1234050

Example


```
{
  "country": "ID",
  "currency": "IDR",
  "payment_option": "15",
  "amount": 1234050
}
```


pay_rqt_merchant_Obj: object

Example

PROPERTIES

merId: string range: (up to 4 chars) required
Merchant ID

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

Example

```
{
  "merId": "5822"
}
```

pay_rqt_customer_Obj: object

PROPERTIES

name: string range: (up to 128 chars) required
Buyer's / Customer's Name

email: string range: (up to 254 chars) required
Providing Buyer's / Customer's email for Fraud Screening purpose

customerId: string range: (up to 16 chars) conditional
Customer ID

- Merchant to assign an unique ID refer to a specific Customer
- This is used for Credit Card tokenization feature. The credit card information will be securely stored in backend system and associated with the `customerId` during the first payment submission. Starting from the 2nd submission, `customerId` will be used to retrieve the corresponding credit card information and Customer only needs to input CVV value.
- Required when `payment_option = "16"`

Example

```
{
  "name": "Tuanku Imam Bonjol",
  "email": "Tuanku.Imam.Bonjol@customer.com",
  "customerId": "CUST0045678"
}
```

pay_rqt_order_Obj: object

PROPERTIES

description: Array< descriptionObj > range: (up to 10 items) required

ITEMS

descriptionObj

Example

```
{
  "description": [
    {
      "orderName": "Order Item 1",
      "unitAmt": 1500000,
      "unit": 2,
      "subAmt": 3000000
    },
    {
      "orderName": "Order Item 2",
      "unitAmt": 2400000,
      "unit": 3,
      "subAmt": 7200000
    }
  ]
}
```

descriptionObj: object

PROPERTIES

orderName: string range: (up to 50 chars) required
Order Item Name / Description

unitAmt: integer range: 1 ≤ x ≤ 99999999999999 required
Unit Amount of each item

- Format: Eliminate punctuation and sign, support 2 decimal places, e.g. Rp 15000.00 = 1500000

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

subAmt: integer range: 1 ≤ x ≤ 99999999999999 required
Sub Amount of the Sum of one particular item with mulitple orders. Namly, Unit Amount x Unit

- Format: Eliminate punctuation and sign, support 2 decimal places, e.g. Rp 600000.00 = 60000000

Example

```
{
  "orderName": "Order Item 1",
  "unitAmt": 1500000,
  "unit": 40,
  "subAmt": 60000000
}
```

pay_rqt_other_Obj: object

PROPERTIES

udfs: Array< udfsObj > range: (up to 20 objects) optional
Array of User Defined Fields

Example

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

udfsObj: object

PROPERTIES

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

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

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

Example

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

paymentRespModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required

PROPERTIES

transaction: pay_rpn_txn_Obj required
system: pay_rpn_system_Obj required

Example

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "transaction": {
      "txnRef": "0002900F064577105001"
    },
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "redirectLink": "<Redirection_HTML_Form>"
    }
  }
}
```

pay_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Returning back the original Transaction Reference No. provided by merchant

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
900020	Plan ID Not Found
900030	Duplicate Transaction Reference
999999	System Error

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

redirectLink: string range: (up to 4096 chars) required
Return Redirect Link in a `HTML Form` format, Merchant is required to use HTTP Form POST to submit the redirection

Example

```
{
  "sysCode": "000000",
  "sysMsg": "Request Successful",
  "redirectLink": "<Redirection_HTML_Form>"
}
```

enquiryRespModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required

PROPERTIES

system: enq_rpn_sys_Obj required
transaction: enq_rpn_txn_Obj required
payment: enq_rpn_payment_Obj required
credit_card: enq_rpn_creditcard_Obj optional
Returned if it is a credit card payment
refunds: Array< enq_rpn_refund_Obj > optional
Returned if refund request is submitted previously
udfs: Array< udfsObj > optional
Array of User Defined Fields
links: Array< hallLinkObj > optional
Collection of related resources

Example

```
{
  "api_gw": {
    "messageId": "89817674-da0O-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful"
    },
    "transaction": {
      "txnRef": "0002900F064577105001",
      "plan_id": "BCA-BCA-3",
      "result_message": "SUCCESS",
      "response_code": "0000"
    },
    "payment": {
      "amount": 1050,
      "currency": "IDR",
      "payment_option": "15",
      "payment_datetime": "2018-06-11T14:10:25+07:00",
      "approval_code": "987021",
      "issuing_bank": "BCA",
      "payment_code": "12345678",
      "wallet_id": "dokuwalletid12345"
    },
    "credit_card": {
      "cardholder_name": "Tuanku Imam Bonjol",
      "brand": "VISA",
      "mcn": "44441xxxxxx1111",
      "3ds_status": "TRUE",
      "liability": "CUSTOMER"
    },
    "refunds": [
      {
        "refundRef": "REF2900F064577105001",
        "amount": 100,
        "bank_refno": "987654",
        "response_code": "0000",
        "response_message": "REFUNDED",
        "refundDatetime": "2018-12-12T14:10:25+07:00"
      }
    ],
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ],
    "links": [
      {
        "href": "/plan/@id",
        "id": "BCA-BCA-3",
        "rel": "plan",
        "method": "GET"
      }
    ]
  }
}
```

enq_rpn_sys_Obj: object

PROPERTIES

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

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

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

Example

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

enq_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required

Example

```
{
  "txnRef": "0002900F064577105001",
```

Returning back the original Transaction Reference No. provided by merchant

plan_id: string range: (up to 100 chars) optional
Return Corresponding Plan ID

result_message: string enum: [SUCCESS, FAILED] range: (up to 20 chars) required
Result Message. Main identifier of transaction success or failed.

Possible Value	Definition
SUCCESS	Transaction is Success
FAILED	Transaction is Failed

response_code: string range: (up to 4 chars) required
Transaction Response Code. Provide detailed response message regarding to different payment options.

- Please see all Possible Value in Response Code Section.

```
  "plan_id": "BCA-BCA-3",
  "result_message": "SUCCESS",
  "response_code": "0000"
}
```

enq_rpn_payment_Obj: object

PROPERTIES

amount: integer range: 1 ≤ x ≤ 99999999999999 required
Returing Payment Amount

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

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

Possible Value	Definition
IDR	Indonesian Rupiah

payment_option: string enum: [04, 15, 16, 19, 22, 25, 26, 28, 31, 32, 33, 34, 35, 36, 37, 38, 41, 44, 50, 51, 53] range: (up to 2 chars) required
Returning back payment option. Please see Payment Channel Code for details.

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

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

approval_code: string range: (up to 20 chars) required
Approval Code / Transaction Number from bank

issuing_bank: string range: (up to 100 chars) required
Issuing Bank

payment_code: string range: (up to 22 chars) optional
Virtual Account identifier for VA transaction. Value exists if the corresponding transaction option make use of payment code.

wallet_id: string range: (up to 128 chars) optional
DOKU Wallet ID/Email

Example

```
{
  "amount": 1050,
  "currency": "IDR",
  "payment_option": "15",
  "payment_datetime": "2018-06-11T14:10:25+07:00",
  "approval_code": "987021",
  "issuing_bank": "BCA",
  "payment_code": "12345678",
  "wallet_id": "dokuwallet1d12345"
}
```

enq_rpn_creditcard_Obj: object

PROPERTIES

cardholder_name: string range: (up to 50 chars) required
Cardholder Name

brand: string range: (up to 10 chars) required
VISA or MASTERCARD

mcn: string range: (up to 16 chars) required
Masked Credit Card Number

- First 6 and last 4 digits of credit card number

3ds_status: string enum: [TRUE, FALSE] range: (up to 5 chars) required
Status of 3D Secure

liability: string enum: [CUSTOMER, MERCHANT, NA] range: (up to 10 chars) required
Liability. The responsible party in the event of chargeback.

Possible Value	Definition
CUSTOMER	If card transaction is done with 3D Secure
MERCHANT	If card transaction is done without 3D Secure (This is an exceptional case when Merchant requests HSBC to turn off 3D Secure for a particular reason)
NA	For Non-Credit Card transaction

Example

```
{
  "cardholder_name": "Tuanku Imam Bonjol",
  "brand": "VISA",
  "mcn": "44411xxxxxx1111",
  "3ds_status": "TRUE",
  "liability": "CUSTOMER"
}
```

enq_rpn_refund_Obj: object

PROPERTIES

refundRef: string range: (up to 30 chars) required
Returning back the original Refund Transaction Reference No. provided by merchant

amount: integer range: 1 ≤ x ≤ 99999999999999 required
Refund Amount

bank_refno: string range: (up to 30 chars) required
Reference Number comes from bank

response_code: string range: (up to 4 chars) required
Response Code

- `0000` means Success, others Failed

response_message: string enum: [VOIDED, REFUNDED, FAILED] range: (up to 20 chars) required
Response Message

refundDatetime: 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 Indonesia local time. Format: `yyyy-MM-dd'T'HH:mm:ss±hh:mm`

Example

```
{
  "refundRef": "REF2900F064577105001",
  "amount": 100,
  "bank_refno": "987654",
  "response_code": "0000",
  "response_message": "REFUNDED",
  "refundDatetime": "2018-12-12T14:10:25+07:00"
}
```

halLinkObj: object

PROPERTIES

href: string range: (up to 100 chars) required
URL Endpoint of the related resource

id: string range: (up to 100 chars) optional
Entity ID of the related resource where it replaces the `@id` in the URI.

Example

```
{
  "href": "/plan/@id",
  "id": "{Entity_ID}",
  "rel": "{Related_Entity_Name}",
  "method": "GET"
}
```

rel: string range: (up to 100 chars) required
Related entity name

method: string range: (up to 100 chars) required
HTTP Method the related resource

> **INFORMATION:**
> This object and fields make use of Hypertext Application Language (HAL) standard

cancelReqtModel: object

PROPERTIES

transaction: cancel_rqt_txn_Obj required
refund: cancel_rqt_refund_Obj required

Example

```
{
  "transaction": {
    "txnRef": "0002900F064577105001"
  },
  "refund": {
    "refundRef": "REF2900F064577105001",
    "amount": 100,
    "reason": "Goods are damaged"
  }
}
```

cancel_rqt_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 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 30 characters

Example

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

cancel_rqt_refund_Obj: object

PROPERTIES

refundRef: string range: (up to 30 chars) required
Unique ID referred to a specific refund transaction

- Required Merchant to generate a unique ID for each refund transaction in alphanumeric format with up to a maximum of 30 characters

amount: integer range: 1 ≤ x ≤ 99999999999999 required
Total Voild / Refund Amount

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

Example

```
{
  "refundRef": "REF2900F064577105001",
  "amount": 100,
  "reason": "Goods are damaged"
}
```

cancelRespModel: object

PROPERTIES

api_gw: commonRespObj required
response: object required
PROPERTIES
transaction: cancel_rpn_txn_Obj required
refund: cancel_rpn_rfd_Obj required

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"
    },
    "refund": {
      "refundRef": "REF2900F064577105001",
      "amount": 100,
      "bank_refno": "987654",
      "response_code": "0000",
      "response_message": "REFUNDED"
    }
  }
}
```

cancel_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 chars) required
Returning back the original Transaction Reference No. provided by merchant

Example

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

cancel_rpn_rfd_Obj: object

PROPERTIES

refundRef: string range: (up to 30 chars) required
Returning back the original Refund Transaction Reference No. provided by merchant

amount: integer range: 1 ≤ x ≤ 99999999999999 required
Returning Total Voided / Refunded Amount

bank_refno: string range: (up to 30 chars) required
Reference Number comes from bank

response_code: string range: (up to 4 chars) required
Response Code

- `0000` means Success, others Failed

response_message: string enum: [VOIDED, REFUNDED, FAILED] range: (up to 20 chars) required
Response Message

Example

```
{
  "refundRef": "REF2900F064577105001",
  "amount": 100,
  "bank_refno": "987654",
  "response_code": "0000",
  "response_message": "REFUNDED"
}
```

statusRtnReqtModel: object

PROPERTIES

Example

transaction: notif_rqt_txn_Obj required
merchant: notif_rqt_merchant_Obj required
payment: notif_rqt_payment_Obj required
credit_card: notif_rqt_creditcard_Obj optional
Return when the transaction is a credit card payment

va: notif_rqt_va_Obj optional
Return when the transaction is a VA payment

other: notif_rqt_other_Obj optional
Return when the transaction is any other payment methods besides credit card and VA payment or contains user-defined meta data


```
{
  "transaction": {
    "txnRef": "0002900F06457705001",
    "plan_id": "BCA-BCA-3",
    "result_message": "SUCCESS",
    "response_code": "0000",
    "status_type": "P"
  },
  "merchant": {
    "merId": "5822"
  },
  "payment": {
    "amount": 1050,
    "payment_option": "15",
    "currency": "IDR",
    "payment_datetime": "2018-06-11T14:10:25+07:00",
    "approval_code": "987021",
    "issuing_bank": "BCA"
  },
  "credit_card": {
    "cardholder_name": "Tuanku Imam Bonjol",
    "brand": "VISA",
    "mcn": "444411xxxxxx1111",
    "3ds_status": "TRUE",
    "liability": "CUSTOMER"
  },
  "va": {
    "payment_code": "8856069112345678"
  },
  "other": {
    "wallet_id": "dokuwalletid12345",
    "udfs": [
      {
        "definition": "Product Image in Base64 format",
        "value": "iVBORw0KGgoAAAANSUhEU..."
      },
      {
        "definition": "Special Notes from Customer",
        "value": "Customer is a non-smoker"
      }
    ]
  }
}
```


notif_rqt_txn_Obj: object

PROPERTIES

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

plan_id: string range: (up to 100 chars) optional
Returning Plan ID

result_message: string enum: [SUCCESS, FAILED] range: (up to 20 chars) conditional
Result Message

- Required when `status_type is NOT "O"`

response_code: string range: (up to 4 chars) conditional
Transaction Response Code

- Please see all Possible Value in Response Code Section.
- Required when `status_type is NOT "O"`

status_type: string enum: [O, P, G, T] range: (up to 1 chars) required
Status Type

Possible Value	Definition
O	Offline Payment, return when the payment is ATM or VA
P	Purchase, always return for ron-recurring
G	Notify Registration, for recurring
T	Notify Update, for recurring

Example


```
{
  "txnRef": "0002900F06457705001",
  "plan_id": "BCA-BCA-3",
  "result_message": "SUCCESS",
  "response_code": "0000",
  "status_type": "P"
}
```


notif_rqt_merchant_Obj: object

PROPERTIES

merId: string range: (up to 4 chars) required
Returning Merchant ID

Example


```
{
  "merId": "5822"
}
```


notif_rqt_payment_Obj: object

PROPERTIES

amount: integer range: 1 ≤ x ≤ 99999999999999 required
Amount of the corresponding Payment Action

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

payment_option: string enum: [04, 15, 16, 19, 22, 25, 26, 28, 31, 32, 33, 34, 35, 36, 37, 38, 41, 44, 50, 51, 53] range: (up to 2 chars) required
Returning back Payment Option the customer chose to settle the payment. Please see Payment Channel Code for details.

currency: string enum: [IDR] range: (up to 3 chars) conditional
Payment Currency. Format: ISO 4217 Alpha

- Required when `status_type is NOT "O"`

Possible Value	Definition
IDR	Indonesian Rupiah

payment_datetime: string range: (up to 25 chars) conditional
Time of generation of the corresponding payment

- Required when `status_type is NOT "O"`
- Format：yyyy-MM-dd'T'HH:mm:ss±hh:mm

approval_code: string range: (up to 20 chars) conditional
Transaction number from bank

- Required when `status_type is NOT "O"`

issuing_bank: string range: (up to 200 chars) required
Issuing Bank

Example


```
{
  "amount": 1050,
  "payment_option": "15",
  "currency": "IDR",
  "payment_datetime": "2018-06-11T14:10:25+07:00",
  "approval_code": "987021",
  "issuing_bank": "BCA"
}
```


notif_rqt_creditcard_Obj: object

PROPERTIES

Example

cardholder_name: string range: (up to 50 chars) conditional
Cardholder Name

- Conditional Field, only when payment is Credit Card transaction

brand: string range: (up to 10 chars) conditional
VISA or MASTERCARD

- Conditional Field, only when payment is Credit Card transaction

mcn: string range: (up to 16 chars) conditional
Masked Credit Card Number

- Conditional Field, only when payment is Credit Card transaction
- First 6 and last 4 digits of credit card number

3ds_status: string range: (up to 5 chars) conditional
Status of 3D Secure

- Conditional Field, only when payment is Credit Card transaction

liability: string enum: [CUSTOMER, MERCHANT] range: (up to 10 chars) conditional
Liability

- Conditional Field, only when payment is Credit Card transaction

```
{
  "cardholder_name": "Tuanku Imam Bonjol",
  "brand": "VISA",
  "mcn": "444411xxxxxx1111",
  "3ds_status": "TRUE",
  "liability": "CUSTOMER"
}
```

notif_rqt_va_Obj: object

PROPERTIES

payment_code: string range: (up to 22 chars) conditional
Virtual Account identifier for VA transaction.

- Return value if payment channel is Virtual Account

Example

```
{
  "payment_code": "8856069112345678"
}
```

notif_rqt_other_Obj: object

PROPERTIES

wallet_id: string range: (up to 128 chars) conditional
DOKU Wallet ID/Email

- Return value if payment channel is DOKU Wallet

udfs: Array< udfsObj > range: (up to 20 objects) optional
Array of User Defined Fields

Example

```
{
  "wallet_id": "dokuwalletid12345",
  "udfs": [
    {
      "definition": "Product Image in Base64 format",
      "value": "iVBORw0KGgoAAAANSUhEU..."
    },
    {
      "definition": "Special Notes from Customer",
      "value": "Customer is a non-smoKer"
    }
  ]
}
```

statusRtnRespModel: object

PROPERTIES

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

Possible Value	Definition
SUCCESS	To acknowledge a successful transaction **IMPORTANT NOTICE:** By default HSBC will IGNORE merchant's response but merchant have an option to reverse the payment if merchant's response is not appropriate or time out occurs. Please contact HSBC for details.

Example

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

directVaEnqReqtModel: object

PROPERTIES

merchant: direct_va_rqt_merchant_Obj required
payment: direct_va_rqt_payment_Obj required

Example

```
{
  "merchant": {
    "merId": "5822"
  },
  "payment": {
    "payment_option": "36",
    "payment_code": "8856069112345678"
  }
}
```

direct_va_rqt_merchant_Obj: object

PROPERTIES

merId: string range: (up to 4 chars) required
Returning Merchant ID

Example

```
{
  "merId": "5822"
}
```

direct_va_rqt_payment_Obj: object

PROPERTIES

payment_option: string range: (up to 2 chars) required
Returning payment option used by the customer that makes payment, please see Payment Channel Code for details.
payment_code: string range: (up to 22 chars) required
Returning Payment Code for Merchant's verification

Example

```
{
  "payment_option": "36",
  "payment_code": "8856069112345678"
}
```

directVaEnqRespModel: object

PROPERTIES

Example

transaction: direct_va_rpn_txn_Obj required
payment: direct_va_rpn_payment_Obj required
customer: direct_va_rpn_customer_Obj required
order: direct_va_rpn_order_Obj required


```
{
  "transaction": {
    "txnRef": "0002900F06457710S001",
    "txnDatetime": "2018-06-11T14:10:25Z",
    "txnResponseCode": "0000",
    "notifyUrl": "https://www.your-domain.com/notification"
  },
  "payment": {
    "country": "ID",
    "currency": "IDR",
    "amount": 0,
    "minAmt": 1000000,
    "maxAmt": 2000000,
    "payment_code": "885606911234567S"
  },
  "customer": {
    "name": "Tuanku Imam Bonjol",
    "email": "Tuanku.Imam.Bonjol@customer.com"
  },
  "order": {
    "description": [
      {
        "orderName": "Order Item 1",
        "unitAmt": 1500000,
        "unit": 2,
        "subAmt": 3000000
      },
      {
        "orderName": "Order Item 2",
        "unitAmt": 2400000,
        "unit": 3,
        "subAmt": 7200000
      }
    ]
  }
}
```


direct_va_rpn_txn_Obj: object

PROPERTIES

txnRef: string range: (up to 30 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 30 characters
- A uniqueness checking will be taken place based on EACH `merId`

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

- Client system time. The timezone is expected to be `GMT+7` (Jakarta local time). Merchant is required to perform timezone conversion if needed. Format: `yyyy-MM-dd'T'HH:mm:ssZ`

txnResponseCode: string range: (up to 4 chars) required
Response Code regarding to Merchant's checking upon the corresponding transaction

Possible Value	Definition
0000	Success
3000	Bill not found
3001	Decline
3002	Bill already paid
3004	Account number / Bill was expired
3006	VA Number not found
9999	Internal Error / Failed

notifyUrl: string range: (up to 255 chars) required
Merchant's URL to receive Status Notification sent by HSBC

- Merchant is required to provide implementation in order to process the JSON message, please see Status Notification API for details.

Example


```
{
  "txnRef": "0002900F06457710S001",
  "txnDatetime": "2018-06-11T14:10:25Z",
  "txnResponseCode": "0000",
  "notifyUrl": "https://www.your-domain.com/notification"
}
```


direct_va_rpn_payment_Obj: object

PROPERTIES

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

Possible Value	Definition
ID	Indonesia

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

Possible Value	Definition
IDR	Indonesian Rupiah

amount: integer range: 0 ≤ x ≤ 99999999999999 required
Payment Amount

- Format: Eliminate punctuation and sign, support 2 decimal places, e.g. Rp 12340.50 = 1234050

minAmt: integer range: 1 ≤ x ≤ 99999999999999 optional
Minimum Amount for Open Amount Scheme

- Format: Eliminate punctuation and sign, support 2 decimal places, e.g. Rp 12340.50 = 1234050

NOTICE:
Merchant can implement Open Amount Scheme by giving an amount range (sending both `minAmt` and `maxAmt`) rather than a fixed amount. (sending `amount` only)

USAGE:
- Sending fields `minAmt` and `maxAmt` together with valid values means it is an Open Amount Scheme
- `amount` must be set to `0`

The following scenario will be considered failed:
- `amount` field is not set to `0` but sending `minAmt` and `maxAmt`
- Either `minAmt` or `maxAmt` is missing
- Value of `minAmt` is greater than `maxAmt` or vice versa

maxAmt: integer range: 1 ≤ x ≤ 99999999999999 optional
Maximum Amount for Open Amount Scheme

- Format: Eliminate punctuation and sign, support 2 decimal places, e.g. Rp 12340.50 = 1234050

payment_code: string range: (up to 22 chars) required
Merchant returns Payment Code to acknowledge

Example


```
{
  "country": "ID",
  "currency": "IDR",
  "amount": 0,
  "minAmt": 1000000,
  "maxAmt": 2000000,
  "payment_code": "8856069112345678"
}
```

direct_va_rpn_customer_Obj: object

PROPERTIES

name: string range: (up to 50 chars) required
Buyer's / Customer's Name

email: string range: (up to 100 chars) required
Providing Buyer's / Customer's email for Fraud Screening purpose

Example

```
{
  "name": "Tuanku Imam Bonjol",
  "email": "Tuanku.Imam.Bonjol@customer.com"
}
```

direct_va_rpn_order_Obj: object

PROPERTIES

description: Array< descriptionObj > required

ITEMS

descriptionObj

Example

```
{
  "description": [
    {
      "orderName": "Order Item 1",
      "unitAmt": 1500000,
      "unit": 2,
      "subAmt": 3000000
    },
    {
      "orderName": "Order Item 2",
      "unitAmt": 2400000,
      "unit": 3,
      "subAmt": 7200000
    }
  ]
}
```

getPlanRespModel: object

PROPERTIES

api_gw: commonRespObj required

response: object required

PROPERTIES

system: systemGetObj required

plans: Array< planObj > required
Array of plan(s)

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "RETURN_MESSAGE",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "system": {
      "sysCode": "000000",
      "sysMsg": "Request Successful",
      "no_of_record": 99,
      "no_of_page": 1
    },
    "plans": [
      {
        "id": "BCA-BCA-3",
        "type": "I",
        "description": "Monthly Installment Plan #1",
        "period": "month",
        "interval": 1,
        "total_count": 6,
        "customer_bank": "BCA",
        "acquiring_bank": "BCA",
        "create_date": "2020-01-01T13:02:00+07:00"
      }
    ]
  }
}
```

systemGetObj: object

PROPERTIES

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

Possible Value	Definition
000000	Request Successful
900030	Duplicate Key ID
900080	Invalid Certificate Format
999999	System Error

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

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

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

Example

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

planObj: object

PROPERTIES

id: string range: (up to 100 chars) required
Plan ID

type: string enum: [I] range: (up to 1 chars) required
Plan Type

Possible Value	Definition
I	Installment
R	Recurring

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

period: string enum: [month] range: (up to 25 chars) required
Period

interval: integer range: 1 ≤ x ≤ 99 required
Interval

total_count: integer range: 1 ≤ x ≤ 999 required
Total Count

customer_bank: string range: (up to 50 chars) required
Customer's Bank

acquiring_bank: string range: (up to 50 chars) required
Merchant's Bank

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

Example

```
{
  "id": "BCA-BCA-3",
  "type": "I",
  "description": "Monthly Installment Plan #1",
  "period": "month",
  "interval": 1,
  "total_count": 6,
  "customer_bank": "BCA",
  "acquiring_bank": "BCA",
  "create_date": "2020-01-01T13:02:00+07:00"
}
```

Lifecycle of Cryptographic Keys

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

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

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

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

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

Key Generation and Certificate Exchange with HSBC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

> **NOTE:**

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

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

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

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

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

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

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

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

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

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

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

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

<Other Information>

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

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

<Other Information>

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

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

<Other Information>

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

Certificates and Keys Maintenance

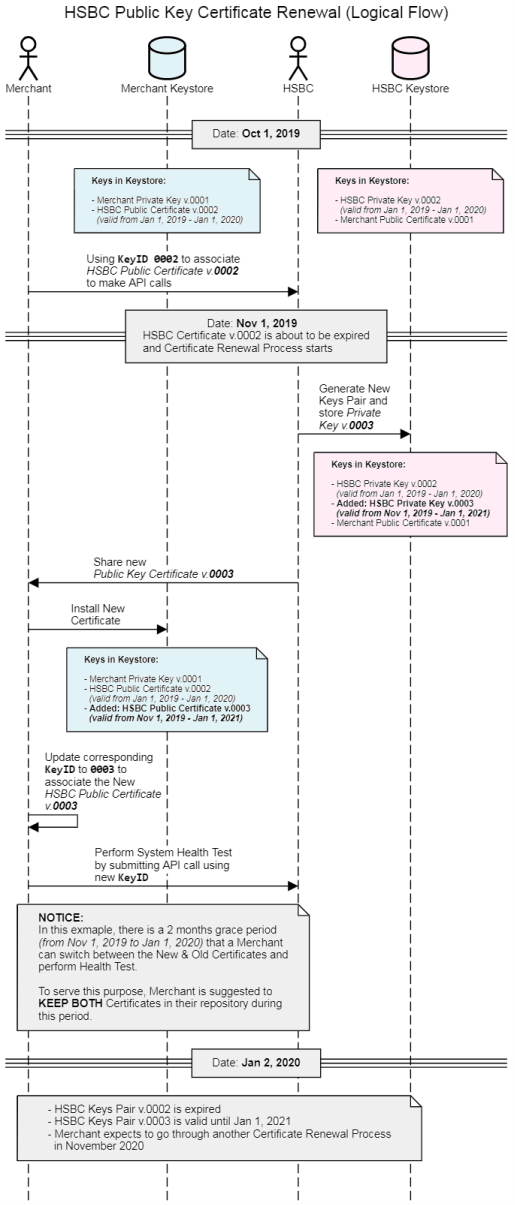
Here are some recommendations to Merchant of how to properly maintain certificates and keys:

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

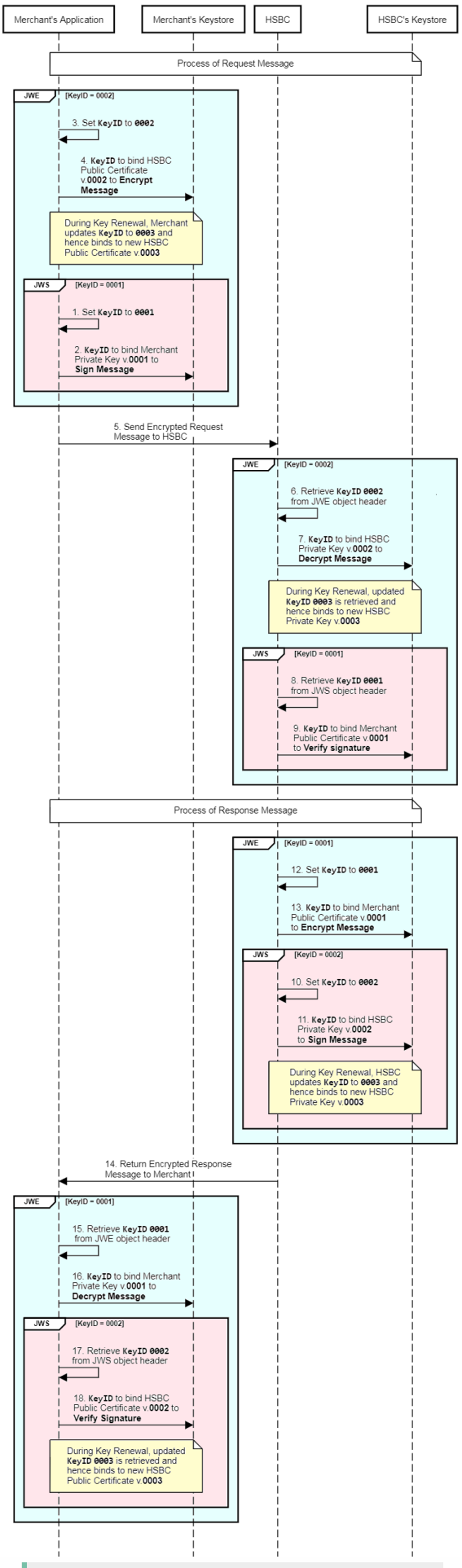
Certificates and Keys Renewal

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

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



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



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

Payment Channel Code

Possible Value	Definition
04	DOKU Wallet
15	Credit Card
16	Credit Card Tokenization
19	CIMB Clicks
22	Sinarmas VA
25	Muamalat Internet Banking
26	Danamon Internet Banking
28	Permata Internet Banking
31	Indomaret
32	CIMB VA
33	Danamon VA
34	BRI VA
35	Alfa VA
36	Permata VA
37	Consumer Finance - Kredivo
38	BNI VA
41	Mandiri VA
44	Maybank VA
50	Link Aja
51	Internet Banking - Jenius
53	OVO

Response Code

> **IMPORTANT NOTICES:** This response codes are not meant for customer. It's for merchant's benefits. Merchant must always treat declined transactions as the customer is required to contact their Bank Issuer for the declined credit card. This data is very confidential. Do not reproduce in any kind to public view. Fail to do so will void merchant's account.

General Response Code

Possible Value	Definition
0000	Successful approval
0001	Failed
5501	Payment channel not registered
5502	Merchant Payment Channel is disabled
5503	Maximum attempt 3 times
5504	WORDS not match
5505	Invalid parameter
5506	Notify failed
5507	Invalid parameter detected / Customer click cancel process
5508	Re-enter transaction
5509	Payment code already expired
5510	Cancel by customer
5511	Not an error, transaction has not been paid
5512	Insufficient parameter
5513	Voided by system
5514	High Risk or Rejected by fraud system
5515	Duplicate PNR
5516	Transaction not found
5517	Error in authorization process
5518	Error parsing XML
5519	Stop at 3D secure page
5520	Register / Scheduler Transaction failed
5521	Invalid merchant
5522	Rates were not found
5523	Failed to get transaction status
5524	Failed to reverse transaction
5525	Transaction can not be processed
5526	Transaction timeout to or from acquirer
5527	Transaction will be processed as Off Us installment
5529	Invalid merchant
5530	Internal server error
5531	Pairing code does not exist
5532	Invalid payment channel
5533	Failed to inquiry list of fund
5534	Invalid pairing code
5535	Invalid token
5536	Time out
5537	Invalid currency
5538	Invalid purchase currency
5539	3D Secure enrollment check failed
5540	3D Secure authentication failed
5541	Form type is not valid
5542	Duplicate transaction ID

Possible Value	Definition
5543	Please check 3D Secure result
5544	Failed to delete token
5545	Failed to void
5547	BIN is not allowed in promo
5548	Invalid parameter
5549	Invalid domain
5550	Invalid IP address
5551	Merchant does not use 1-click service
5552	Invalid token expire
5553	Failed to tokenize / generate merchant key
5554	Off-Us Reward process
5555	Undefined error / Failed to decrypt
5556	Tokenization Failed
5557	Merchant does not have tokenization credential
5558	Cannot capture to System
5559	Batch list is null
5560	Batch ID is already available
5561	OCO ID is not exist
5562	Failed settlement
5563	Failed parsing PARES
5564	Batch ID not found in transaction
5565	Feature for master merchant id
5566	Invalid amount parameter captured
5567	Failed do re-notify
5568	Failed Refund
5569	Void/Refund amount is not Valid
5570	Transaction is canceled by merchant
5571	Failed Register Paycode
5572	No response
5573	Failed create bill
5574	Merchant not found or not active
5575	Transaction already voided
003D	Wrong OTP or customer did not continue the transaction at 3D Secure page
00BA	Blocked by Acquirer
00BB	BIN blocking, because card origin was not allowed to go through the payment
0098	3D Secure failure -- The card does not support 3D Secure

Credit Card Response Code

Credit card transaction response code can be checked here

> ! HSBC Mobile Collection will add `00` prefix in front of response code. Eg : Do Not Honor `05` will be `0005`.

DOKU Wallet Response Code

Possible Value	Definition
0E01	Failed Get Merchant
0E02	Master Merchant Inactive
0E03	Invalid Words from Merchant
0E04	Invalid Merchant
0E05	Failed to Process Payment
0E06	Payment Method not Defined
0E07	Failed to Execute Pre Auth Plugins
0E08	Failed to Execute Post Auth Plugins
0E09	Invalid Pay ID
0E10	Error Pay ID
0E11	Failed to Execute Pre Trans Pre Trans MIP Plugins
0E12	Verify Response STOP From Merchant
0E13	Failed Verify to Merchant
0E14	Failed Send Payment Cash Wallet
0E15	Notify Response STOP From Merchant
0E16	Failed Notify to Merchant
0E18	Failed to Execute Post Trans MIP Plugins
0E19	Not Enough Cash Balance and Don't Have Credit Card
0E20	Spender Does Not Have Link to Credit Card
0E21	Error Check 3D Secure Credit Card
0E22	PIN/OTP Is Not Valid
0E23	Please Input CVV2
0E24	Invalid Session
0E25	Failed to Send Link Authentication to Card Holder
0E26	Insufficient Params
0E27	Failed to Execute Pre Trans CIP Plugins
0E28	Failed to Execute Post Trans CIP Plugins
0E29	Failed to Send Payment MIP Credit Card
0E30	You Do Not Have PIN

Possible Value	Definition
0E31	Duplicate Invoice No
0E32	URL Not Found
0E33	Customer Not Found
0E34	Void Process Failed
0E35	Failed Send ONE TIME PIN to your email
0E36	Failed Send Link to create PIN to your email
0E37	This Spender Cannot Transact in This Merchant
0E38	You Have Reach Your DOKU ID Transaction Limit
0E39	Process MIP Transaction Failed
0E99	Error System

Alfa / Indomaret / Permata / Mandiri Response Code

Possible Value	Definition
0001	Decline (internal error)
0013	Invalid amount
0014	Bill not found
0066	Decline
0088	Bill already paid

Download Swagger

Click here to download Swagger 2.0 file in YAML format.

Disclaimer

IMPORTANT NOTICE

This document is issued by The Hongkong and Shanghai Banking Corporation Limited, Hong Kong ("HSBC"). HSBC does not warrant that the contents of this document are accurate, sufficient or relevant for the recipient's purposes and HSBC gives no undertaking and is under no obligation to provide the recipient with access to any additional information or to update all or any part of the contents of this document or to correct any inaccuracies in it which may become apparent. Receipt of this document in whole or in part shall not constitute an offer, invitation or inducement to contract. The recipient is solely responsible for making its own independent appraisal of the products, services and other content referred to in this document. This document should be read in its entirety and should not be photocopied, reproduced, distributed or disclosed in whole or in part to any other person without the prior written consent of the relevant HSBC group member. Copyright: HSBC Group 2019. ALL RIGHTS RESERVED.