INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

API Specification for Hong Kong FPS

Version: 2.8

Description

This document introduces the **OpenAPI specification** which describes the REST APIs for HSBCs Collection of digital payments - HK Faster Payments System (HKFPS).

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

Update Log

- [Dec 15, 2021] **v2.8** Refined several descriptions in content sections.
- [Jan 18, 2021] **v2.7** Added content section Testing
- [Aug 24, 2020] **v2.6** Removed request field `country` in qrCodeRequestModel
- [Jun 22, 2020] **v2.5** Added `NOTICE` message in Payment Status Notification API
- [Jun 8, 2020] **v2.4**
  - Added new request message object `for_nonbill_payment` to Payment Simulation API
  - Added new API - Refund Simulation API
  - Added new Section - Download Swagger
- [Nov 8, 2019] **v2.3** Updated `API Base URL` including both Sandbox and Production
- [Sep 20, 2019] **v2.2** Updated `Disclaimer`
- [Sep 11, 2019] **v2.1**
  - Enhanced Section `API Connectivity`
  - Added Content Section `REFERENCE`
- [Aug 05, 2019] **v2.0**
  - Added New Business Capability - **Refund:**
  - Added Refund Use Case in API Use Case
  - Added New API Refund Request API
  - Added New API Refund Notification API
  - Enhanced Payment Status Enquiry API with new optional fields related to Refund Scenario
- [Jun 5, 2019] **v1.28**
  - Added new request field `tip` in Payment QR Code Creation API
  - Added `proCode` 800050 regarding to the validation of `tip`
- [Mar 27, 2019] **v1.27**
  - Added new field `amtEditInd` in Payment QR Code Creation API
  - Remove object `for_bill_payment` in Simulation API and release optional fields to support Editable Payment Amount scenario
- [Mar 18, 2019] **v1.26** Updated the URL of HKMA QR Code Specification
- [Jan 28, 2019] **v1.25**
  - Added optional request object `for_bill_payment` in Simulation API to support Bill Payment model
  - Enhanced caption of request field `merTimeout` in QR Code Creation API
- [Oct 16, 2018] **v1.24** Added HTTP Header `message_encrypt` description in Simulation API
- [Sep 28, 2018] **v1.23**
  - Content Enhanced in Content Section
  - Updated the link of PHP JOSE library
  - Changed response field `suppInfo` to optional
  - Changed response field `bankTxnId` length to `16`
- [Sep 24, 2018] **v1.22** Changed the sample value of `bankTxnId`
- [Sep 18, 2018] **v1.21** Changed the default testing value of `keyId`
- [Sep 11, 2018] **v1.20**
  - Added message samples of field `returnReason` in response message
  - Updated Image `QR Code Payment Flow`
- [Aug 27, 2018] **v1.19**
  - Modified the possible value of response field `proCode` of QR Code API
  - Changes field name from `debtorName` to `suppInfo` in Notification API
  - Added field `suppInfo` in Enquiry API
  - Added Content Section `Data Type Overview`
- [Aug 15, 2018] **v1.18** Changed Content Type in HTTP Header of Status Notification API to `text/plain`
- [Aug 14, 2018] **v1.17**
  - Modified the example value of field `qrCode` in QR Code API
  - Content enhanced on `Connectivity AT-A-Glance` and `FAQ` section
- [Aug 8, 2018] **v1.16** Removed possible value `999999` from request field `proCode` in Notification API
- [Aug 7, 2018] **v1.15** Revised the whole layout of this document and separate the API onboarding procedure to another document
- [Aug 6, 2018] **v1.14** Modified time format of field `bankTxnTime` in Enquiry & Status Notification API
- [Jul 11, 2018] **v1.13** Changed URL endpoint of `Payment Simulation API`
- [Jun 29, 2018] **v1.12**
  - Updated content section `Getting Started` `Key Renewal` `Message Encryption`
  - Added new API `Payment Simulation`
  - Added new validation rule of `QR Code API`
- [Jun 14, 2018] **v1.11**
  - Changed maxLength of field `posMachineId` `employeeId`
  - Added Possible Value of `proCode` of response of QR Code API & Enquiry API
- [Jun 13, 2018] **v1.10a**
  - Changed datetime format to JSON standard `yyyy-MM-dd'T'HH:mm:ssZ`
  - Removed Possible Value `Fail` from field `proCode` in Enquiry API
  - Added Possible Value `HK` into field `country` in QR Code request API
- [Jun 6, 2018] **v1.9**
  - Changed fields order in QR Code API
  - Added `debtorName` to Notification API
  - Removed `qrCodeRefId`
- [Jun 4, 2018] **v1.8a**
  - Changed maxlength of `qrCode` `employeeId`
  - Added `currency` to Enquiry & Notification API
  - Added `bankTxnId` to Enquiry API
  - Renamed fields `bankTxnTime` `totalAmtPaid` `subAmtPaid`
- [May 31, 2018] **v1.7** Changed API endpoint names & field names including `txnRef` `payMethod` `posMachineId` `employeeId`
- [May 29, 2018] **v1.6** All amount fields are changed from maxlength 16 to 12 according to EMVCo standard
- [May 28, 2018] **v1.5** New Fields added `merTimeout`
- [May 25, 2018] **v1.4** Refined HTTP Response Code
- [May 16, 2018] **v1.3** New Fields added `totalAmt` `subAmt` `fpsTxnTime`
- [May 14, 2018] **v1.2** New Fields added `txnChannel` `posMachineId` `employeeId` `qrCodeRefId`
- [May 4, 2018] **v1.1** Final Revision
- [May 3, 2018] **v1.0** Initial Version

How to Read this Document

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

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

Use Cases for this API

There are two API use cases in this document:

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

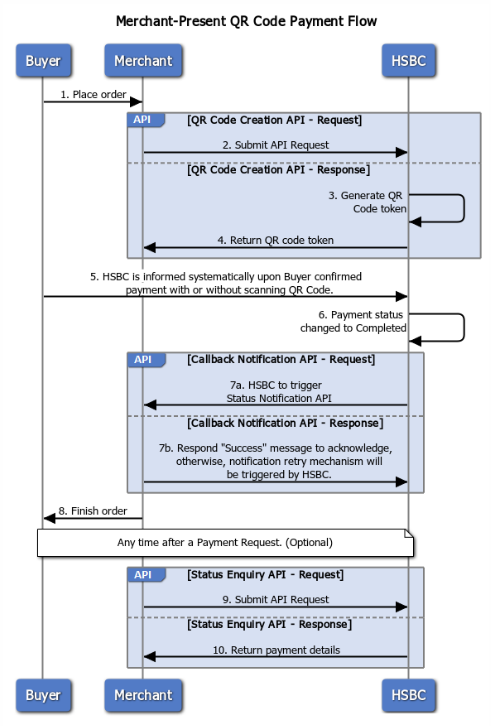
REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

1. A Buyer visits a Merchant's online store to place an order and make a payment. This is done with or without Scanning the Merchant-Presented QR Code which is based on HKMA and HKICL standard (also known as the Common QR code specification).
2. A Buyer requests a Merchant to refund a settled transaction. The refund process will start after the refund request is asynchronously submitted to HSBC.

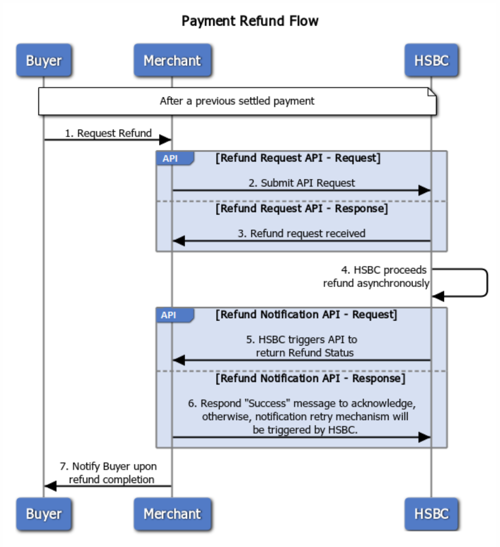
Making a Payment

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


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

1. The Buyer places an order.
2. The Merchant submits a Create QR code request.
3. The HSBC backend system generates a QR Code token or image.
4. HSBC returns the QR Code token or image via the API response.
5. The Merchant converts the QR Code token to a QR Code Image and displays it on its online store. HSBC receives an acknowledgement as soon a the Buyer confirms payment after scanning the QR code. In the case that online store is a Mobile App, the Buyer is directed to its FPS Mobile Payment/Banking App for payment. There is no QR code scanning as the QR Code token is passed to the Buyer's App. Likewise, HSBC is informed when the Buyer confirms payment.
6. The Payment is completed.
7. An acknowledge message is sent back via a Status Notification API immediately after the payment status is changed to Completed at the HSBC backend system.
8. The Merchant notifies a completed order to the buyer.
9. To check the payment status, the Merchant submits an Order Status Enquiry any time after a payment request, or if no acknowledge message is returned after a certain period of time.
10. HSBC returns the payment status via the Status Enquiry API response.

Refund a settled Transaction


Payment Refund Flow
Buyer
Merchant
HSBC
After a previous settled payment
1. Request Refund
API
[Refund Request API - Request]
2. Submit API Request
[Refund Request API - Response]
3. Refund request received
4. HSBC proceeds refund asynchronously
API
[Refund Notification API - Request]
5. HSBC triggers API to return Refund Status
[Refund Notification API - Response]
6. Respond "Success" message to acknowledge, otherwise, notification retry mechanism will be triggered by HSBC.
7. Notify Buyer upon refund completion
Buyer
Merchant
HSBC

1. The Buyer places order.
2. The Merchant submits a Refund Request API request.
3. HSBC responds to the Merchant after the refund request is received.
4. HSBC will processes the refund asynchronously.
5. An acknowledge message is sent back via a Refund Notification API directly after the refund is completed.
6. The Merchant responds to the API to acknowledge, if not HSBC will trigger a retry mechanism.
7. Merchant notifies Buyer upon refund completion.

> ! **NOTE:**
> Merchant can also submit a Status Enquiry API to check refund status anytime after a refund request, or if found no refund acknowledge is received after a certain period of time.

Simulate a Payment and Refund in Sandbox Environment

The Sandbox Environment is offered for testing purpose. It comes with two additional features:

- Bypass Data Encryption which enables the developer to first focus testing on API connectivity.
- On a self-service basis, the developer or tester can submit a Simulation API to simulate a payment or refund.

×

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED

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

How to make an API request
with Plain Message
with Data Encryption

Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS

Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API

Refund
Refund Request API
Refund Notification API

Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA

Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE

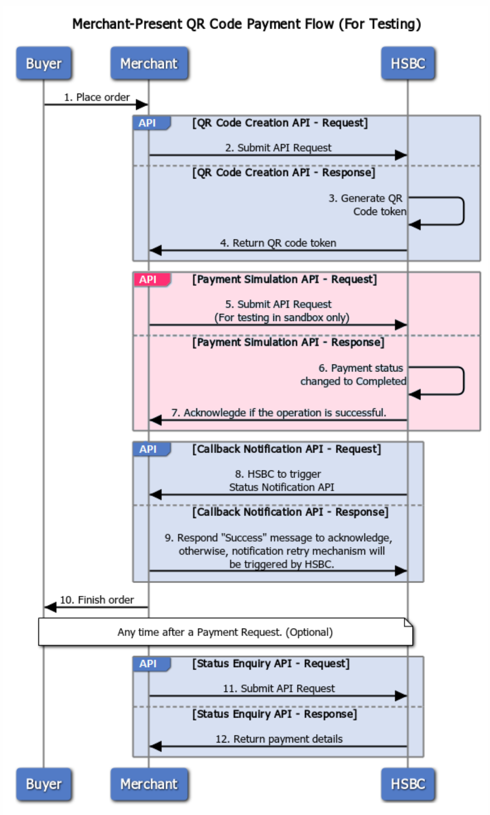
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal

Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

The Payment Simulation API must be submitted directly after a QR Code is requested, please see the flow diagram below:


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

Refund Simulation API should be submitted after the completion of a payment just like the original Refund Request API. Below is the Summary of the Simulation API Usability over different testing scenarios:

Steps	Testing Scenario #1 (Self-Served Basis)	Testing Scenario #2 (with HSBC Support)
i.	Submit QR Code Creation	Submit QR Code Creation
ii.	Submit Payment Simulation API	Contact HSBC support team to scan and pay QR Code with actual App
iii.	Verify Payment Notification	Verify Payment Notification
iv.	Submit Refund Simulation	Submit Refund Request, then contact HSBC support team to reconcile with downstream refund backend systems
v.	Verify Refund Notification	Verify Refund Notification

Payment Models

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

APIs	E/M-Commerce	m-POS	Online Bill Payment
Payment QR Code Creation API	✔	✔	✘
Payment Status Enquiry API	✔	✔	✘
Payment Status Notification API	✔	✔	✔
Payment Simulation API	✔	✔	✔
Refund Request API	✔	✔	Available when Opt In
Refund Notification API	✔	✔	Available when Opt In
Refund Simulation API	✔	✔	Available when Opt In
✔ = Always Available ✘ = Not Available			

How to Connect

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

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

API Gateway URL

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

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

Sandbox

×

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED

How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS

Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA

Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE

Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

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

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

API Authentication

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*	Pass value in API request message body

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

×

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED

How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS

Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA

Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE

Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

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`		

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

Components \ Steps	Message Signing	Message Encryption	Message Decryption	Verify Signature
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:

```
curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mchk-ea-merchantserv
 -H "message_encrypt: false"
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
 -H "Content-Type: application/json"
 -d "{ \"txnRef\": \"0002900F064577105000000001\", \"merId\": \"0002900F0645774\"}"
```

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 Client ID in HTTP header `x-HSBC-client-id`.
4. Put Client Secret in HTTP header `x-HSBC-client-secret`.
5. Set `Content-Type` to JSON format.
6. Plain `json` message payload.

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

Making an API Request with Message Encryption

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

```
curl -X POST "https://devclustercmb.api.p2g.netd2.hsbc.com.hk/glcm-mobilecoll-mchk-ea-merchantserv
 -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
 -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
 -H "Content-Type: application/json"
 -d "eyJraWQiOiIwMDAxIiwiZW5jIjoiQTEyOEdDTSIsImFsZyI6IlJTQS1PQUVQLTI1NiJ9.W4nobHoVXUMOXGM5I-WGPZt8
```

1. Submit the `POST` request to the API URL endpoint.
2. Put Client ID in the HTTP header `x-HSBC-client-id`.
3. Put Client Secret in the HTTP header `x-HSBC-client-secret`.
4. Set `Content-Type` to JSON format.
5. 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 *(For general field)*	AlphaNumeric and Symbols	General field means field which is **NOT** a critical field. HSBC system will execute characters checking upon all string fields we received in order to tackle security vulnerability, such as Cross-site Scripting. Yet, we recommend you to use AlphaNumeric only for most cases.

String *(For critical field)*	`0-9` `a-z` `A-Z` `-` `_` `.`	Critical field is used to be either a key or search criteria in HSBC backend system and hence tight restriction is applied to the allowed characters.
		
		Moreover, the starting and ending space of the string value will be trimmed before stored in HSBC system. For example, string `" example 12 34 "` will be trimmed to `"example 12 34"`.
		
		List of Critical Fields:
		`txnRef`
		`merId`
		`posMachineId`
		`employeeId`

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 presented 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.

×

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
  Make Payment
  Refund
  Testing
  Payment Models

GETTING STARTED

How to Connect
  API Gateway URL
  API Authentication
  User Identification
  Connection Security
  Message Security
    Sign & Encrypt
    Decrypt & Verify
    Summary
How to make an API request
  with Plain Message
  with Data Encryption
Data Type Overview
FAQ
  SSL Connection
  Message Encryption
  JOSE Framework
  QR Code Image

API OPERATIONS

Payments
  Payment QR Code Creation API
  Payment Status Enquiry API
  Payment Status Notification API
Refund
  Refund Request API
  Refund Notification API
Simulation
  Payment Simulation API
  Refund Simulation API

API SCHEMA

Schema Definitions
  commonRespObj
  exceptionModel
  qrCodeRequestModel
  qrCodeResponseModel
  qrCodeResponseModel_response
  txnEnqRequestModel
  txnEnqResponseModel
  txnEnqResponseModel_response
  subAmountObj
  refundAmountObj
  refundRqRequestModel
  refundRqResponseModel
  refundRqResponseModel_response
  paySimRequestModel
  simBillPaymentObj
  paySimResponseModel
  paySimResponseModel_response
  statusReturnRequestModel
  statusReturnResponseModel
  refundNotificationRequestModel
  refundNotificationResponseModel

REFERENCE

Lifecycle of Cryptographic Keys
  Key Generation & Exchange
  Key Maintenance
  Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

Time Zone Control:

Aspect	Format	Definition & Important Notice
In Request Message	`yyyy-MM-dd'T'HH:mm:ssZ`	Time zone is expected to be `GMT+8` (Hong Kong local time). Merchant is required to perform any necessary time zone conversion before submit request if needed.
In Response Message	`yyyy-MM-dd'T'HH:mm:ss±hh:mm`	Timezone returned in `api_gw` object is generated from HSBC API Gateway which located in Cloud and hence is calculated in `GMT+0`. 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 this url does not belong to HSBC, use it at your own discretion. By clicking the url or website, it means you accept these terms and conditions.

Where can I download JOSE libraries for development?

For your reference, you can use 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.

QR Code Image Generation Questions

The QR Code token returned by the API is a String token, the Merchant is required to generate a QR Code image by any QR Code image generator. The Merchant is free to choose any compatible QR Code image creation libraries or mechanism of their own choice. Below are some references:

- Project Nayuki
- CrunchifyQRCode (Java)

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

Payments

Contains resource collections for QR Code payment, enquiry and notification.

Payments

Payment QR Code Creation API

POST `/payment/qrCode`

DESCRIPTION

This API creates QR Code token with the aim of making payment. Once this API request is submitted from merchants, HSBC payment platform will return QR Code token to Merchant according to the Common QR code specification. Merchant needs to convert the QR Code token to QR Code Image and display on its online store. HSBC will be informed systematically upon Buyer confirms payment after scanning QR code. In case the online store is a Mobile App, Buyer will be directed to its FPS Mobile Payment/Banking App for payment and there will be no QR code scanning as the QR Code token will be passed to the Buyer's App. Likewise, HSBC will be informed systematically upon Buyer confirms payment.

REQUEST PARAMETERS

x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
Content-Type required in header	application/json

REQUEST BODY

qrCodeRequestModel *Click link to navigate Schema Definition	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: application/json

Request Example

```
{
  "merId": "0002900F0645774",
  "txnRef": "0002900F06457710500000001",
  "txnChannel": "01",
  "txnTime": "2018-06-11T14:10:25Z",
  "merTimeout": 1440,
  "payMethod": [
    "HKFPS"
  ],
  "currency": "HKD",
  "amount": 1050,
  "tip": 200,
  "amtEditInd": "Y",
  "notifyUrl": "https://merchant.com/returnStatus",
  "goodsDes": "Description of goods.",
  "posMachineId": "00112233-4455-6677-8899-aabbccddeeff",
  "employeeId": "00112233-4455-6677-8899-xxyyzzxxyyzz"
}
```

INTRODUCTION
- Description
- Update Log
- How to Read this Document
- Use Cases for this API
  - Make Payment
  - Refund
  - Testing
  - Payment Models

GETTING STARTED
- How to Connect
  - API Gateway URL
  - API Authentication
  - User Identification
  - Connection Security
  - Message Security
    - Sign & Encrypt
    - Decrypt & Verify
    - Summary
- How to make an API request
  - with Plain Message
  - with Data Encryption
- Data Type Overview
- FAQ
  - SSL Connection
  - Message Encryption
  - JOSE Framework
  - QR Code Image

API OPERATIONS
- Payments
  - Payment QR Code Creation API
  - Payment Status Enquiry API
  - Payment Status Notification API
- Refund
  - Refund Request API
  - Refund Notification API
- Simulation
  - Payment Simulation API
  - Refund Simulation API

API SCHEMA
- Schema Definitions
  - commonRespObj
  - exceptionModel
  - qrCodeRequestModel
  - qrCodeResponseModel
  - qrCodeResponseModel_response
  - txnEnqRequestModel
  - txnEnqResponseModel
  - txnEnqResponseModel_response
  - subAmountObj
  - refundAmountObj
  - refundRqRequestModel
  - refundRqResponseModel
  - refundRqResponseModel_response
  - paySimRequestModel
  - simBillPaymentObj
  - paySimResponseModel
  - paySimResponseModel_response
  - statusReturnRequestModel
  - statusReturnResponseModel
  - refundNotificationRequestModel
  - refundNotificationResponseModel

REFERENCE
- Lifecycle of Cryptographic Keys
  - Key Generation & Exchange
  - Key Maintenance
  - Key Renewal
- Refund Transaction Reference
- Download Swagger

DISCLAIMER
- Disclaimer

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "txnRef": "0002900F06457710500000001",
    "currency": "HKD",
    "amount": 1050,
    "proCode": "000000",
    "proMsg": "Transaction Successful",
    "qrCode":
"00020101021226460012hk.com.hkicl021020180711010512180731112742520400005303344540510.505802HK5902NA6
  }
}
```

Response Example (400 Bad Request)

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

Payments

Payment Status Enquiry API

POST `/payment/enquiry`

DESCRIPTION

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

x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]

REQUEST PARAMETERS

Content-Type required in header	application/json

REQUEST BODY

txnEnqRequestModel *Click link to navigate Schema Definition	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774"
}
```

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "txnRef": "0002900F06457710500000001",
    "proCode": "000000",
    "proMsg": "Payment Success",
    "currency": "HKD",
    "totalAmtPaid": 5000,
    "totalAmtRefunded": 2500,
    "totalAmtPendingRefund": 2000,
    "arrayOfSubAmt": [
      {
        "bankTxnId": "HC80502097323467",
        "bankTxnTime": "2018-06-11T14:10:25+08:00",
        "subAmtPaid": 2500,
        "suppInfo": "Supplementary Information",
        "arrayOfRefundAmt": [
          {
            "refundTxnId": "HC1196080000319R",
            "refundTxnTime": "2018-06-11T18:12:44+08:00",
            "subAmtRefund": 2500,
            "refundStatus": "SUCCESS"
          }
        ]
      },
      {
        "bankTxnId": "HC80502057680987",
        "bankTxnTime": "2018-06-11T15:11:12+08:00",
        "subAmtPaid": 2500,
        "suppInfo": "Supplementary Information",
        "arrayOfRefundAmt": [
          {
            "refundTxnId": "HC1196080002256R",
            "refundTxnTime": "2018-06-11T21:20:11+08:00",
            "subAmtRefund": 2000,
            "refundStatus": "PENDING"
          }
        ]
      }
    ]
  }
}
```

Response Example (400 Bad Request)

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

Payments

Payment Status Notification API

POST `/<Callback URL predefined by Merchant>`

DESCRIPTION

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
  Make Payment
  Refund
  Testing
  Payment Models

GETTING STARTED

How to Connect
  API Gateway URL
  API Authentication
  User Identification
  Connection Security
  Message Security
    Sign & Encrypt
    Decrypt & Verify
    Summary
How to make an API request
  with Plain Message
  with Data Encryption
Data Type Overview
FAQ
  SSL Connection
  Message Encryption
  JOSE Framework
  QR Code Image

API OPERATIONS

Payments
  Payment QR Code Creation API
  Payment Status Enquiry API
  Payment Status Notification API
Refund
  Refund Request API
  Refund Notification API
Simulation
  Payment Simulation API
  Refund Simulation API

API SCHEMA

Schema Definitions
  commonRespObj
  exceptionModel
  qrCodeRequestModel
  qrCodeResponseModel
  qrCodeResponseModel_response
  txnEnqRequestModel
  txnEnqResponseModel
  txnEnqResponseModel_response
  subAmountObj
  refundAmountObj
  refundRqRequestModel
  refundRqResponseModel
  refundRqResponseModel_response
  paySimRequestModel
  simBillPaymentObj
  paySimResponseModel
  paySimResponseModel_response
  statusReturnRequestModel
  statusReturnResponseModel
  refundNotificationRequestModel
  refundNotificationResponseModel

REFERENCE

Lifecycle of Cryptographic Keys
  Key Generation & Exchange
  Key Maintenance
  Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

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

Implementation	This is a Callback API. HSBC will trigger this API call and defines the interface with OpenAPI standard. Merchant is required to provide implementation.
Retry Mechanism	If no success response is received, up to 4 retries will be triggered in every 2 minutes. Maximum 5 calls including the 1st attempt.
Endpoint Definition	Field `notifyUrl` from Payment QR Code Creation API will be used as URL endpoint of the corresponding transaction.
Exception Handling	Only success case will be returned. Merchant can submit a Payment Status Enquiry API request if found no acknowledge message returned after a certain period of time.

> ! **IMPORTANT NOTE:**
> Field values returned from the Notification must be verified with the original transaction to avoid unauthorized QR Code alternation.

REQUEST PARAMETERS

Content-Type required in header	text/plain

REQUEST BODY

statusReturnRequestModel *Click link to navigate Schema Definition	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: text/plain

Request Example

```
{
  "merId": "0002900F0645774",
  "txnRef": "0002900F06457710500000001",
  "currency": "HKD",
  "amount": 1050,
  "proCode": "000000",
  "proMsg": "Transaction Successful",
  "bankTxnId": "HC80502097323467",
  "bankTxnTime": "2018-06-11T14:10:25+08:00",
  "suppInfo": "Supplementary Information"
}
```

RESPONSES

200 OK statusReturnResponseModel	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)

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

Refund

Contains resource collections for Refund and Refund Notification.

Refund

Refund Request API

POST `/refund/request`

DESCRIPTION

Merchant can request to refund a settled transaction by calling this API. It is a common practice that the refund amount should not exceed the original payment amount and we therefore ensure the same in our API logic. Once the refund request is submitted, HSBC payment platform will proceed the refund asynchronously, and a Refund Status Notification will be pushed to Merchant once the refund transaction is completed. This version of API support making only one successful Refund request for the same original transaction.

> ! **IMPORTANT NOTE:**
> The downstream systems invoked in the refund request scenario is not available for testing purpose, and hence the Refund Simulation API is offered in the Sandbox environment. Please see inside for more details.

REQUEST PARAMETERS

x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
Content-Type required in header	application/json

REQUEST BODY

refundRqRequestModel *Click link to navigate Schema Definition	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774",
  "bankTxnId": "HC80502097323467",
  "txnAmt": 5000,
  "refundAmt": 4000,
  "notifyUrl": "https://merchant.com/returnRefundStatus"
}
```

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "bankTxnId": "HC80502097323467",
    "refundTxnId": "HC1196080000319R",
    "proCode": "000000",
    "proMsg": "Refund Request Submitted"
  }
}
```

Response Example (400 Bad Request)

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

Refund

Refund Notification API

×

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

POST `/<Callback URL predefined by Merchant>`

DESCRIPTION

Refund status will be returned to Merchant by asynchronous callback once the refund request has been processed. HSBC will push the result back to Merchant by calling this API.

Implementation	This is a Callback API. HSBC will trigger this API call and defines the interface with OpenAPI standard. Merchant is required to provide implementation.
Retry Mechanism	If no success response is received, up to 4 retries will be triggered in every 2 minutes. Maximum 5 calls including the 1st attempt.
Endpoint Definition	Field `notifyUrl` from Refund Request API will be used as URL endpoint of the corresponding refund transaction.
Exception Handling	Merchant can submit a Payment Enquiry API request if found no acknowledge message returned after a certain period of time.

REQUEST PARAMETERS

Content-Type required in header	text/plain

REQUEST BODY

refundNotificationRequestModel *Click link to navigate Schema Definition	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

Request Content-Types: text/plain

Request Example

```
{
  "merId": "0002900F0645774",
  "txnRef": "0002900F06457710500000001",
  "bankTxnId": "HC80502097323467",
  "refundTxnId": "HC1196080000319R",
  "refundTxnTime": "2018-06-11T14:10:25+08:00",
  "refundAmt": 4000,
  "proCode": "000000",
  "proMsg": "SUCCESS"
}
```

RESPONSES

200 OK refundNotificationResponseModel	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)

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

Simulation

Contains resource collections for Simulating a Payment and Refund Request.

Simulation

Payment Simulation API

POST `/hk/payment/simulation`

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

DESCRIPTION

To simulate the process which the buyer confirms the payment with or without scanning the QR Code.

This is only available in Sandbox Environment. Please see details in here.

REQUEST PARAMETERS

x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
Content-Type required in header	application/json
message_encrypt required in header	false

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774",
  "is_notification_encrypted": "Y",
  "for_bill_payment": {
    "currency": "HKD",
    "amount": 123450,
    "suppInfo": "Supplementary Information"
  },
  "for_nonbill_payment": {
    "currency": "HKD",
    "amount": 123450,
    "suppInfo": "Supplementary Information"
  }
}
```

REQUEST BODY

paySimRequestModel *Click link to navigate Schema Definition	*Data Encryption is not required.*

RESPONSES

200 OK paySimResponseModel	Successful operation. *Data Encryption is not required.*
400 Bad Request exceptionModel	Bad Request.
403 Forbidden	Authorization credentials are missing or invalid.
404 Not Found	Empty resource/resource not found.
500 Internal Server Error	The request failed due to an internal error.

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "proCode": "000000",
    "txnRef": "0002900F06457710500000001",
    "proMsg": "Payment Success"
  }
}
```

Response Example (400 Bad Request)

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

Simulation

Refund Simulation API

POST `/hk/refund/simulation`

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

DESCRIPTION

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
- Make Payment
- Refund
- Testing
- Payment Models

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

How to make an API request
- with Plain Message
- with Data Encryption

Data Type Overview
FAQ
- SSL Connection
- Message Encryption
- JOSE Framework
- QR Code Image

API OPERATIONS
Payments
- Payment QR Code Creation API
- Payment Status Enquiry API
- Payment Status Notification API

Refund
- Refund Request API
- Refund Notification API

Simulation
- Payment Simulation API
- Refund Simulation API

API SCHEMA
Schema Definitions
- commonRespObj
- exceptionModel
- qrCodeRequestModel
- qrCodeResponseModel
- qrCodeResponseModel_response
- txnEnqRequestModel
- txnEnqResponseModel
- txnEnqResponseModel_response
- subAmountObj
- refundAmountObj
- refundRqRequestModel
- refundRqResponseModel
- refundRqResponseModel_response
- paySimRequestModel
- simBillPaymentObj
- paySimResponseModel
- paySimResponseModel_response
- statusReturnRequestModel
- statusReturnResponseModel
- refundNotificationRequestModel
- refundNotificationResponseModel

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

Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

This API aims to simulate the Refund Request API in Sandbox environment while the testing data will not processed in the refund downstream system.

> **IMPORTANT NOTE:**
> Except the API endpoint, the API usage flow, business checking logic and message structure are identical to Refund Request API.
>
> Merchant is suggested to call this API during development phase and switch the API endpoint back to Refund Request API until the testing phase is completed.

REQUEST PARAMETERS

x-hsbc-client-id required in header	[Client ID]
x-hsbc-client-secret required in header	[Client Secret]
Content-Type required in header	application/json

Request Content-Types: application/json

Request Example

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774",
  "bankTxnId": "HC80502097323467",
  "txnAmt": 5000,
  "refundAmt": 4000,
  "notifyUrl": "https://merchant.com/returnRefundStatus"
}
```

REQUEST BODY

refundRqRequestModel *Click link to navigate Schema Definition	*Data Encryption is enforced. API Schema intends to demonstrate the skeleton of the message payload only.*

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "bankTxnId": "HC80502097323467",
    "refundTxnId": "HC1196080000319R",
    "proCode": "000000",
    "proMsg": "Refund Request Submitted"
  }
}
```

Response Example (400 Bad Request)

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

Schema Definitions

commonRespObj: object

PROPERTIES

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

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

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

Example

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

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

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

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

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

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models
GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image
API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API
API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel
REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger
DISCLAIMER
Disclaimer

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

exceptionModel: object

PROPERTIES

api_gw: commonRespObj required

Example

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

qrCodeRequestModel: object

PROPERTIES

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

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

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

- Required Merchant to generate a unique ID for each transaction.
- A uniqueness checking will be taken place based on each merId, duplicate ID will be rejected.
- This reference will be seen in HSBC's Daily Collection Detail Report and Global Information Reporting (GIR) Bank Statement.

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

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

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

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

merTimeout: integer range: 0 ≤ x ≤ 9999 optional
Merchant Time-out Period

- For dynamic QR code, a merchant may specify the latest time which the merchant system will wait for the payment.

> ! **NOTE:** If QR Code is scanned after the specified time-out time, it will be rejected by Mobile Banking App.

- Specify in minutes (i.e. 2 for 2 mins, and 1440 for 1 day, and 9999 for around 7 days, i.e. 1 week)
- The time period will be incremented at the time when HSBC system receives this API request
- If this field is not provided, or value = 0, it means no time-out will be specified in the generated QR code

payMethod: string[] required
Payment Method

Possible Value	Definition
HKFPS	HK Faster Payments System

- Multiple Payment Method will be supported in later version.

ITEMS
string

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

Possible Value	Definition
HKD	Hong Kong dollar
CNY	Renminbi

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

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

> ! **NOTE:** This is the total amount summed by both Payment Amount and **Tip**. For example, if tip is `$2.00` and payment amount is `$8.50`, total is `$10.50` and the value of this field should be `1050`

tip: integer range: 1 ≤ x ≤ 999999999999 optional
Tip for a payment

- Tip amount must not exceed Payment Amount
- Format: Eliminate punctuation and sign, support 2 decimal places according to ISO 4217, e.g. $10.50 = 1050

> ! **NOTE:** If there is a tip, all `amount` fields used in all APIs will include this tip

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

- To indicate if the payment amount can be editable in the midst of one payment

> ! **NOTE:** Payment amount is editable **only** when both this indicator is `Y` **and** merchant has indicated in their application form that QR amount amendment is allowed

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

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

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

- Required when `"txnChannel" = "01"`

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

- Required when `"txnChannel" = "01"`

Example

```
{
  "merId": "0002900F0645774",
  "txnRef": "0002900F06457710500000001",
  "txnChannel": "01",
  "txnTime": "2018-06-11T14:10:25Z",
  "merTimeout": 1440,
  "payMethod": [
    "HKFPS"
  ],
  "currency": "HKD",
  "amount": 1050,
  "tip": 200,
  "amtEditInd": "Y",
  "notifyUrl": "https://merchant.com/returnStatus",
  "goodsDes": "Description of goods.",
  "posMachineId": "00112233-4455-6677-8899-aabbccddeeff",
  "employeeId": "00112233-4455-6677-8899-xxyyzzxxyyzz"
}
```

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
- Make Payment
- Refund
- Testing
- Payment Models

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

How to make an API request
- with Plain Message
- with Data Encryption

Data Type Overview
FAQ
- SSL Connection
- Message Encryption
- JOSE Framework
- QR Code Image

API OPERATIONS
Payments
- Payment QR Code Creation API
- Payment Status Enquiry API
- Payment Status Notification API

Refund
- Refund Request API
- Refund Notification API

Simulation
- Payment Simulation API
- Refund Simulation API

API SCHEMA
Schema Definitions
- commonRespObj
- exceptionModel
- qrCodeRequestModel
- qrCodeResponseModel
- qrCodeResponseModel_response
- txnEnqRequestModel
- txnEnqResponseModel
- txnEnqResponseModel_response
- subAmountObj
- refundAmountObj
- refundRqRequestModel
- refundRqResponseModel
- refundRqResponseModel_response
- paySimRequestModel
- simBillPaymentObj
- paySimResponseModel
- paySimResponseModel_response
- statusReturnRequestModel
- statusReturnResponseModel
- refundNotificationRequestModel
- refundNotificationResponseModel

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

Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

qrCodeResponseModel: object

PROPERTIES

api_gw: commonRespObj required

response: qrCodeResponseModel_response required

Example

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "txnRef": "0002900F06457710500000001",
    "currency": "HKD",
    "amount": 1050,
    "proCode": "000000",
    "proMsg": "Transaction Successful",
    "qrCode":
"00020101021226460012hk.com.hkicl021020180711010512180731112742520400005303344540510.505802HK5902NA60

  }
}
```

qrCodeResponseModel_response:

PROPERTIES

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

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

- Format: ISO 4217 Alpha (e.g. HKD = Hong Kong Dollar)

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

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

Possible Value	Definition
000000	Transaction Successful
800050	Tip amount exceeds Payment amount
900030	Duplicate Transaction Reference
900040	Payment Currency and Settlement Currency is Not Matched
900050	Transaction Channel Not Available with the corresponding Merchant

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

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

qrCode: string range: (up to 512 chars) conditional
QR Code Token

- QR Code Token will only be returned if it is a successful transaction.

Example

```
{
  "txnRef": "0002900F06457710500000001",
  "currency": "HKD",
  "amount": 1050,
  "proCode": "000000",
  "proMsg": "Transaction Successful",
  "qrCode":
"00020101021226460012hk.com.hkicl021020180711010512180731112742520400005303344540510.505802HK5902NA60

}
```

txnEnqRequestModel: object

PROPERTIES

txnRef: string (Critical Field) range: (up to 25 chars) required
Merchant to provide transaction ID that referred to a specific transaction

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

Example

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774"
}
```

txnEnqResponseModel: object

PROPERTIES

api_gw: commonRespObj required

response: txnEnqResponseModel_response required

Example

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "txnRef": "0002900F06457710500000001",
    "proCode": "000000",
    "proMsg": "Payment Success",
    "currency": "HKD",
    "totalAmtPaid": 5000,
    "totalAmtRefunded": 2500,
    "totalAmtPendingRefund": 2000,
    "arrayOfSubAmt": [
      {
        "bankTxnId": "HC80502097323467",
        "bankTxnTime": "2018-06-11T14:10:25+08:00",
        "subAmtPaid": 2500,
        "suppInfo": "Supplementary Information",
        "arrayOfRefundAmt": [
          {
            "refundTxnId": "HC1196080000319R",
            "refundTxnTime": "2018-06-11T18:12:44+08:00",
            "subAmtRefund": 2500,
            "refundStatus": "SUCCESS"
          }
        ]
      },
      {
        "bankTxnId": "HC80502057680987",
        "bankTxnTime": "2018-06-11T15:11:12+08:00",
        "subAmtPaid": 2500,
        "suppInfo": "Supplementary Information",
        "arrayOfRefundAmt": [
          {
            "refundTxnId": "HC1196080002256R",
            "refundTxnTime": "2018-06-11T21:20:11+08:00",
            "subAmtRefund": 2000,
            "refundStatus": "PENDING"
          }
        ]
      }
    ]
  }
}
```

txnEnqResponseModel_response:

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
  Make Payment
  Refund
  Testing
  Payment Models
GETTING STARTED
How to Connect
  API Gateway URL
  API Authentication
  User Identification
  Connection Security
  Message Security
    Sign & Encrypt
    Decrypt & Verify
    Summary
How to make an API request
  with Plain Message
  with Data Encryption
Data Type Overview
FAQ
  SSL Connection
  Message Encryption
  JOSE Framework
  QR Code Image
API OPERATIONS
Payments
  Payment QR Code Creation API
  Payment Status Enquiry API
  Payment Status Notification API
Refund
  Refund Request API
  Refund Notification API
Simulation
  Payment Simulation API
  Refund Simulation API
API SCHEMA
Schema Definitions
  commonRespObj
  exceptionModel
  qrCodeRequestModel
  qrCodeResponseModel
  qrCodeResponseModel_response
  txnEnqRequestModel
  txnEnqResponseModel
  txnEnqResponseModel_response
  subAmountObj
  refundAmountObj
  refundRqRequestModel
  refundRqResponseModel
  refundRqResponseModel_response
  paySimRequestModel
  simBillPaymentObj
  paySimResponseModel
  paySimResponseModel_response
  statusReturnRequestModel
  statusReturnResponseModel
  refundNotificationRequestModel
  refundNotificationResponseModel
REFERENCE
Lifecycle of Cryptographic Keys
  Key Generation & Exchange
  Key Maintenance
  Key Renewal
Refund Transaction Reference
Download Swagger
DISCLAIMER
Disclaimer

PROPERTIES

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

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

proCode	proMsg	Definition
000000	Payment Success	Only Single Payment in the said transaction and it is completed
100010	Pending	Only Single Payment in the said transaction and it is pending
200010	Multiple Transactions or Refund Request Found	For any case when multi-transactions or refund request is found. Required Merchant to look into the sub-array for details
900010	Transaction Record Not Found	Self-explanatory

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

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

- Format: ISO 4217 Alpha

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

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

> ! **NOTE:** This amount is to indicate if one customer accidentally submit a **full** payment more than once but **not** a partial payment which will be supported in later version.

totalAmtRefunded: integer range: 1 ≤ x ≤ 999999999999 optional
Total Amount of all completed refund

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

> ! **NOTE:** Pending refund is excluded.

totalAmtPendingRefund: integer range: 1 ≤ x ≤ 999999999999 optional
Total Amount of pending refund

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

arrayOfSubAmt: Array< subAmountObj > required

ITEMS

subAmountObj

Example

```
{
  "txnRef": "0002900F06457710500000001",
  "proCode": "000000",
  "proMsg": "Payment Success",
  "currency": "HKD",
  "totalAmtPaid": 5000,
  "totalAmtRefunded": 2500,
  "totalAmtPendingRefund": 2000,
  "arrayOfSubAmt": [
    {
      "bankTxnId": "HC80502097323467",
      "bankTxnTime": "2018-06-11T14:10:25+08:00",
      "subAmtPaid": 2500,
      "suppInfo": "Supplementary Information",
      "arrayOfRefundAmt": [
        {
          "refundTxnId": "HC1196080000319R",
          "refundTxnTime": "2018-06-11T18:12:44+08:00",
          "subAmtRefund": 2500,
          "refundStatus": "SUCCESS"
        }
      ]
    },
    {
      "bankTxnId": "HC80502057680987",
      "bankTxnTime": "2018-06-11T15:11:12+08:00",
      "subAmtPaid": 2500,
      "suppInfo": "Supplementary Information",
      "arrayOfRefundAmt": [
        {
          "refundTxnId": "HC1196080002256R",
          "refundTxnTime": "2018-06-11T21:20:11+08:00",
          "subAmtRefund": 2000,
          "refundStatus": "PENDING"
        }
      ]
    }
  ]
}
```

subAmountObj: object

PROPERTIES

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

- This transaction reference id will be seen in HSBC's Daily Collection Detail Report and Global Information Reporting (GIR) Bank Statement. It contains 16 characters with prefix HC such as HCxxxxxxxxxxxxxx where x refers to alphanumeric characters.

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

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

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

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

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

- If this field does not contain any value, it will not be returned in the response message.

arrayOfRefundAmt: Array< refundAmountObj > optional

ITEMS

refundAmountObj

Example

```
{
  "bankTxnId": "HC80502097323467",
  "bankTxnTime": "2018-06-11T14:10:25+08:00",
  "subAmtPaid": 2500,
  "suppInfo": "Supplementary Information",
  "arrayOfRefundAmt": [
    {
      "refundTxnId": "HC1196080000319R",
      "refundTxnTime": "2018-06-11T18:12:44+08:00",
      "subAmtRefund": 2500,
      "refundStatus": "SUCCESS"
    }
  ]
}
```

refundAmountObj: object

PROPERTIES

refundTxnId: string range: (up to 16 chars) optional
Refund transaction reference ID. Please see here for Format details.

refundTxnTime: string range: (up to 25 chars) required
Bank Transaction time of the corresponding refund process

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

subAmtRefund: integer range: 1 ≤ x ≤ 999999999999 required
Amount of requested Refund

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

refundStatus: string range: (up to 140 chars) required
Status of Corresponding requested Refund

Possible Value	Definition
SUCCESS	Refund Completed
REJECTED	Refund Rejected
PENDING	Refund process is pending

Example

```
{
  "refundTxnId": "HC1196080000319R",
  "refundTxnTime": "2018-06-11T18:12:44+08:00",
  "subAmtRefund": 2500,
  "refundStatus": "SUCCESS"
}
```

refundRqRequestModel: object

PROPERTIES

Example

×

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
- Make Payment
- Refund
- Testing
- Payment Models

GETTING STARTED

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

How to make an API request
- with Plain Message
- with Data Encryption

Data Type Overview
FAQ
- SSL Connection
- Message Encryption
- JOSE Framework
- QR Code Image

API OPERATIONS

Payments
- Payment QR Code Creation API
- Payment Status Enquiry API
- Payment Status Notification API

Refund
- Refund Request API
- Refund Notification API

Simulation
- Payment Simulation API
- Refund Simulation API

API SCHEMA

Schema Definitions
- commonRespObj
- exceptionModel
- qrCodeRequestModel
- qrCodeResponseModel
- qrCodeResponseModel_response
- txnEnqRequestModel
- txnEnqResponseModel
- txnEnqResponseModel_response
- subAmountObj
- refundAmountObj
- refundRqRequestModel
- refundRqResponseModel
- refundRqResponseModel_response
- paySimRequestModel
- simBillPaymentObj
- paySimResponseModel
- paySimResponseModel_response
- statusReturnRequestModel
- statusReturnResponseModel
- refundNotificationRequestModel
- refundNotificationResponseModel

REFERENCE

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

Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

txnRef: string (Critical Field) range: (up to 25 chars) required
Merchant provides the original transaction reference that refers to a specific transaction

merId: string (Critical Field) range: (up to 15 chars) required
Merchant provides Merchant ID for identification

bankTxnId: string range: (up to 16 chars) required
Merchant provides HSBC transaction ID that refers to a specific inward credit payment

- This transaction ID will be seen in HSBC's Daily Collection Detail Report and Global Information Reporting (GIR) Bank Statement. It contains 16 characters with prefix HC such as HCxxxxxxxxxxxxxx where x refers to alphanumeric characters.

txnAmt: integer range: 1 ≤ x ≤ 999999999999 required
Merchant provides original Payment Amount that refers to particular received amount

- The original received amount can be found in Payment Status Notification API or Business Collect Report
- Format: Eliminate punctuation and sign, support 2 decimal places according to `ISO 4217`, e.g. $10.50 = 1050

refundAmt: integer range: 1 ≤ x ≤ 999999999999 required
Merchant provides requested Refund Amount

- Refund Amount must not exceed original Payment Amount
- Format: Eliminate punctuation and sign, support 2 decimal places according to `ISO 4217`, e.g. $10.50 = 1050

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

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774",
  "bankTxnId": "HC80502097323467",
  "txnAmt": 5000,
  "refundAmt": 4000,
  "notifyUrl": "https://merchant.com/returnRefundStatus"
}
```

refundRqResponseModel: object

PROPERTIES

api_gw: commonRespObj required
response: refundRqResponseModel_response required

Example

```
{
  "api_gw": {
    "messageId": "89817674-da00-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "bankTxnId": "HC80502097323467",
    "refundTxnId": "HC1196080000319R",
    "proCode": "000000",
    "proMsg": "Refund Request Submitted"
  }
}
```

refundRqResponseModel_response:

PROPERTIES

bankTxnId: string range: (up to 16 chars) required
Returning back the original HSBC transaction ID provided by merchant

refundTxnId: string range: (up to 16 chars) optional
Returning back a refund transaction ID that refer to a specific refund transaction. Please see here for Format details.

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

Possible Value	Return Message	Remark
000000	Refund Request Submitted	
800010	Refund Request Not Allowed: Prior refund found	When you make additional refund request for the same transaction which already has a processed refund request or a refund request under processing, the system will return this code.
800020	Refund Request Not Allowed: 31 calendar days acceptable refund period is over. Please note the day when the original successful transaction is made is taken as day 1.	E.g. if payment time is 2019-01-01 00:00:00, the merchant is allowed to make refund request by 2019-01-31 23:59:59 the latest.
800030	Refund Request Not Allowed: Not Supported in Current Channel	Offline and batch mode bill payment are not supported.
800040	Refund Amount exceed Original Payment Transaction	Please verify `txnAmt` and `refundAmt`
900060	Payment Transaction Not Found or is Not Matched	Please check `bankTxnId`
900070	Payment Transaction Amount is Not Matched	Please ensure `txnAmt` = the original received amount which can be found in Payment Status Notification API or Business Collect Report

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

Example

```
{
  "bankTxnId": "HC80502097323467",
  "refundTxnId": "HC1196080000319R",
  "proCode": "000000",
  "proMsg": "Refund Request Submitted"
}
```

paySimRequestModel: object

PROPERTIES

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

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

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

for_bill_payment: simBillPaymentObj conditional
for_nonbill_payment: simBillPaymentObj conditional

> ! **NOTE:**
> Only choose between either a Bill Payment or a Non-Bill Payment

Example

```
{
  "txnRef": "0002900F06457710500000001",
  "merId": "0002900F0645774",
  "is_notification_encrypted": "Y",
  "for_bill_payment": {
    "currency": "HKD",
    "amount": 123450,
    "suppInfo": "Supplementary Information"
  },
  "for_nonbill_payment": {
    "currency": "HKD",
    "amount": 123450,
    "suppInfo": "Supplementary Information"
  }
}
```

simBillPaymentObj: object

PROPERTIES

currency: string enum: [HKD, CNY] range: (up to 3 chars) required
Payment Currency for Bill Payment Simulation

Possible Value	Definition
HKD	Hong Kong dollar

Example

```
{
  "currency": "HKD",
  "amount": 123450,
  "suppInfo": "Supplementary Information"
}
```

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models
GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image
API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API
API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel
REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger
DISCLAIMER
Disclaimer

Possible Value	Definition
CNY	Renminbi

- Format: ISO 4217 Alpha

amount: integer range: 1 ≤ x ≤ 999999999999 required
Payment Amount for Bill Payment Simulation

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

suppInfo: string range: (up to 140 chars) required
Supplementary Info for Bill Payment Simulation

- This value will be shown in subsequent Push Notification.

paySimResponseModel: object

PROPERTIES

api_gw: commonRespObj required
response: paySimResponseModel_response required

Example

```
{
  "api_gw": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "proCode": "000000",
    "txnRef": "0002900F06457710500000001",
    "proMsg": "Payment Success"
  }
}
```

paySimResponseModel_response:

PROPERTIES

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

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

Possible Value	Definition
000000	Payment Success
900010	Transaction Record Not Found
900050	Transaction Channel Not Available with the corresponding Merchant

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

Example

```
{
  "proCode": "000000",
  "txnRef": "0002900F06457710500000001",
  "proMsg": "Payment Success"
}
```

statusReturnRequestModel: object

PROPERTIES

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

txnRef: string (Critical Field) range: (up to 25 chars) required
Returning back Transaction Reference No.

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

- Format: ISO 4217 Alpha (e.g. HKD = Hong Kong Dollar)

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

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

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

Possible Value	Definition
000000	Transaction Successful

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

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

- This transaction reference id will be seen in HSBC's Daily Collection Detail Report and Global Information Reporting (GIR) Bank Statement. It contains 16 characters with prefix HC such as HCxxxxxxxxxxxxxx where x refers to alphanumeric characters.

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

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

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

- If this field does not contain any value, it will not be passed in the resquest message.

Example

```
{
  "merId": "0002900F0645774",
  "txnRef": "0002900F06457710500000001",
  "currency": "HKD",
  "amount": 1050,
  "proCode": "000000",
  "proMsg": "Transaction Successful",
  "bankTxnId": "HC80502097323467",
  "bankTxnTime": "2018-06-11T14:10:25+08:00",
  "suppInfo": "Supplementary Information"
}
```

statusReturnResponseModel: object

PROPERTIES

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

Example

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

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

refundNotificationRequestModel: object

PROPERTIES

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

txnRef: string (Critical Field) range: (up to 25 chars) required
Returning back Original Transaction Reference No.

bankTxnId: string range: (up to 16 chars) required
Returning back HSBC transaction ID that refers to a specific inward credit payment.

refundTxnId: string range: (up to 16 chars) optional
Returning back refund transaction ID that refer to a specific refund transaction. Please see here for Format details.

refundTxnTime: string range: (up to 25 chars) required
Returning refund completion Transaction time

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

refundAmt: integer range: 1 ≤ x ≤ 999999999999 required
Returning back Refund Amount

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

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

Possible Value	Definition
000000	SUCCESS
100010	REJECTED
100020	PENDING

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

Example

```
{
  "merId": "0002900F0645774",
  "txnRef": "0002900F06457710500000001",
  "bankTxnId": "HC80502097323467",
  "refundTxnId": "HC1196080000319R",
  "refundTxnTime": "2018-06-11T14:10:25+08:00",
  "refundAmt": 4000,
  "proCode": "000000",
  "proMsg": "SUCCESS"
}
```

refundNotificationResponseModel: object

PROPERTIES

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

Example

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

Lifecycle of Cryptographic Keys

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

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

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

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

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

Key Generation and Certificate Exchange with HSBC

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

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

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

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

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

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

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

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

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models
GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image
API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API
API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel
REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger
DISCLAIMER
Disclaimer

> ! **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>

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

×

INTRODUCTION

Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED

How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

API OPERATIONS

Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA

Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE

Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER

Disclaimer

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.

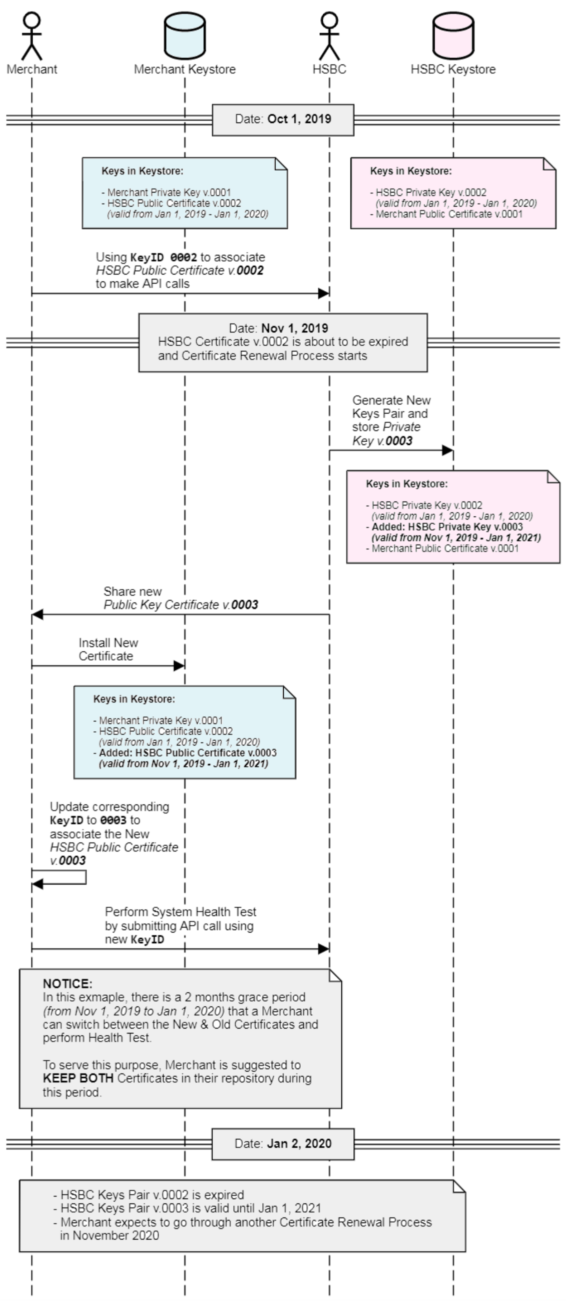
Certificates and Keys Renewal

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.

HSBC Public Key Certificate Renewal (Logical Flow)


Merchant
Merchant Keystore
HSBC
HSBC Keystore
Date: Oct 1, 2019
Keys in Keystore:
- Merchant Private Key v.0001
- HSBC Public Certificate v.0002 (valid from Jan 1, 2019 - Jan 1, 2020)
Keys in Keystore:
- HSBC Private Key v.0002 (valid from Jan 1, 2019 - Jan 1, 2020)
- Merchant Public Certificate v.0001
Using KeyID 0002 to associate HSBC Public Certificate v.0002 to make API calls
Date: Nov 1, 2019
HSBC Certificate v.0002 is about to be expired and Certificate Renewal Process starts
Generate New Keys Pair and store Private Key v.0003
Keys in Keystore:
- HSBC Private Key v.0002 (valid from Jan 1, 2019 - Jan 1, 2020)
- Added: HSBC Private Key v.0003 (valid from Nov 1, 2019 - Jan 1, 2021)
- Merchant Public Certificate v.0001
Share new Public Key Certificate v.0003
Install New Certificate
Keys in Keystore:
- Merchant Private Key v.0001
- HSBC Public Certificate v.0002 (valid from Jan 1, 2019 - Jan 1, 2020)
- Added: HSBC Public Certificate v.0003 (valid from Nov 1, 2019 - Jan 1, 2021)
Update corresponding KeyID to 0003 to associate the New HSBC Public Certificate v.0003
Perform System Health Test by submitting API call using new KeyID
NOTICE:
In this exmaple, there is a 2 months grace period (from Nov 1, 2019 to Jan 1, 2020) that a Merchant can switch between the New & Old Certificates and perform Health Test.
To serve this purpose, Merchant is suggested to KEEP BOTH Certificates in their repository during this period.
Date: Jan 2, 2020
- HSBC Keys Pair v.0002 is expired
- HSBC Keys Pair v.0003 is valid until Jan 1, 2021
- Merchant expects to go through another Certificate Renewal Process in November 2020

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models

GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image

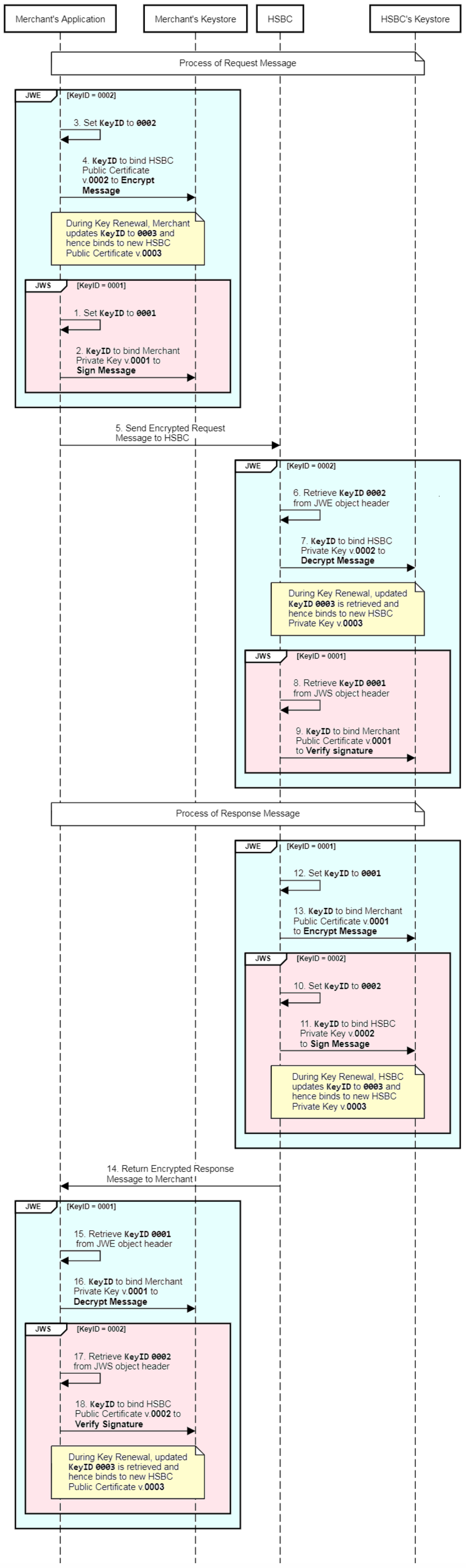
API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API

API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel

REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger

DISCLAIMER
Disclaimer

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


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

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

INTRODUCTION
Description
Update Log
How to Read this Document
Use Cases for this API
Make Payment
Refund
Testing
Payment Models
GETTING STARTED
How to Connect
API Gateway URL
API Authentication
User Identification
Connection Security
Message Security
Sign & Encrypt
Decrypt & Verify
Summary
How to make an API request
with Plain Message
with Data Encryption
Data Type Overview
FAQ
SSL Connection
Message Encryption
JOSE Framework
QR Code Image
API OPERATIONS
Payments
Payment QR Code Creation API
Payment Status Enquiry API
Payment Status Notification API
Refund
Refund Request API
Refund Notification API
Simulation
Payment Simulation API
Refund Simulation API
API SCHEMA
Schema Definitions
commonRespObj
exceptionModel
qrCodeRequestModel
qrCodeResponseModel
qrCodeResponseModel_response
txnEnqRequestModel
txnEnqResponseModel
txnEnqResponseModel_response
subAmountObj
refundAmountObj
refundRqRequestModel
refundRqResponseModel
refundRqResponseModel_response
paySimRequestModel
simBillPaymentObj
paySimResponseModel
paySimResponseModel_response
statusReturnRequestModel
statusReturnResponseModel
refundNotificationRequestModel
refundNotificationResponseModel
REFERENCE
Lifecycle of Cryptographic Keys
Key Generation & Exchange
Key Maintenance
Key Renewal
Refund Transaction Reference
Download Swagger
DISCLAIMER
Disclaimer

Refund Transaction Reference

Format : `HC1[2-digit year][1-digit month][2-digit day][7-digit number]R`

1-digit month	Definition
1	January
2	February
3	March
4	April
5	May
6	June
7	July
8	August
9	September
A	October
B	November
C	December

Download Swagger

Click here to download Swagger 2.0 file in YAML format.

Disclaimer

IMPORTANT NOTICE