API Specification for United Kingdom Cards and Alternate Payment Methods

Version: 1.1

Description

This document introduces the **OpenAPI specification** for the REST APIs of the HSBC Collection of digital payments - HSBC Omni Collect in the UK.

The target audience of this document is the Developer, Business Analyst and other related Project Team Members.

Update Log

- [Jan 3, 2022] **v1.1** Revised several content sections
- [Nov 23, 2021] **v1.0** Initial Version

How to Read this Document

Get to know what we offer in the Features Overview and take away the key ideas of our API operations. Before you connect to our APIs, remember to collect all prerequisites and go through all requirements in the How to Connect section.

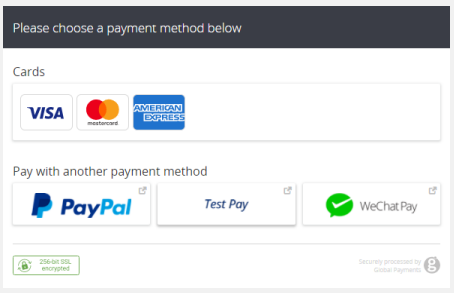
Features Overview

Checkout Solutions for Payment

You accept payments from your customers by integrating your website or app with the Hosted Payment Page or Direct Payment serving different payment methods.

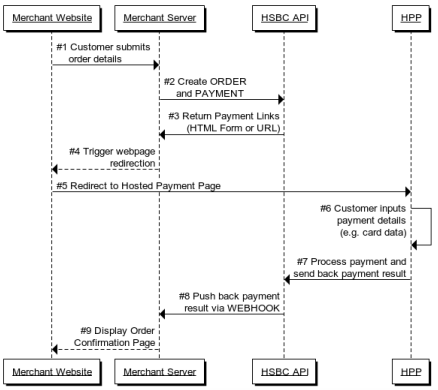
> **IMPORTANT NOTE:**
> Conditions may apply to the availability of each checkout solution and its associated payment options, please check with our support team for details.

Hosted Payment | Direct Payment

Hosted Payment Page	
Overview	The Hosted Payment Page (HPP) is a PCI DSS v3.2 compliant redirect solution, allowing you capture card data without having to worry about the PCI overhead associated with a traditional API integration.
Supported Payment Options	• **Debit / Credit Cards** - Card Data is captured securely inside HPP • **Paypal** - Redirect to Paypal Checkout page • **WeChat Pay** - Payment QR Code will be displayed in a redirected page • **Test Pay** - A payment simulator available in Sandbox environment for testing purpose. 
Integration Methods	• **Full-Page Redirection** A full-page redirect will be performed. • **iFrame/WebView Optimization** You can render the payment form using your own embedded iFrame or WebView, or as a Lightbox. • **Pay by Link** A Payment URL link you can embed directly in your website, an SMS or an email.

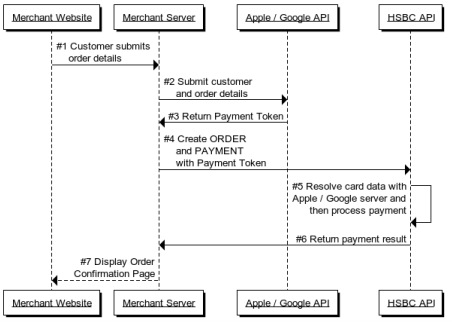
Direct Payment	
Overview	Payment will be processed simultaneously in an API call.
Supported Payment Options	• **Apple Pay** - Card Data Encryption is handled by Apple. • **Google Pay** - Card Data Encryption is handled by Google.
Integration Method	**Step 1. Obtain a Payment Token** - Payment Token is a token provided by Apple or Google that the merchant can use to bind a particular customer's card detail stored securely in Apple's or Google's server. Please visit Apple Pay and Google Pay developer site to learn how to obtain a Payment Token by calling their APIs. **Step 2. Process Payment** - Put the Payment Token in the corresponding API request message and HSBC Omni Collect processes the payment simultaneously as a typical card payment. A successful or a failed payment is returned in the synchronous API response message.

API Use Case of Hosted Payment Page



Please see API operations Create Order, Create Payment for an Order and Webhooks for more details.

API Use Case of Direct Payment



Please see API operations Create Order and Create Payment for an Order for more details.

Void or Refund a Payment

A payment can be voided (cancelled) before settlement. Please refer to API operation Update a Payment for more details.

You can also perform a full or partial refund if a payment transaction has been settled or batched. Please refer to API operation Create Refund for a Payment for more details.

Webhooks for Payment

An asynchronous callback will be pushed back to the Merchant for different payment events, such as a successful payment or a failed payment. Please refer to Webhooks for details.

How to Connect

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

	Definition	Components
API Authentication	HTTP BASIC Authentication	• Username • Password
	Locate API Gateway Policy of the corresponding user	• Profile ID
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

To make API calls, you need to include this before each API endpoint.

Production
https://ws-api-platform.business.hsbc.co.uk/glcm-mobilecoll-mcuk-ea-merchantservices-proxy/v1

Sandbox
https://ws-api-platform-pprd.business.hsbc.co.uk/glcm-mobilecoll-mcuk-ea-merchantservices-proxy/v1

API Authentication

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

Profile ID	
Purpose	API Gateway locates the corresponding policy of the specific API consumer
Components	• Profile ID
Where to get it?	Delivered by HSBC via secure email during onboarding procedure

Profile ID	
Implementation	In HTTP header: `x-hsbc-profileid: [Profile ID]`

User Identification

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

Connection Security

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

Message Security - Data Encryption and Signing

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

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

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

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

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

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

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

How to Sign and Encrypt Outgoing Message

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

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

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

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

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

#2  JWSHeader header = new JWSHeader
                .Builder(JWSAlgorithm.RS256)
                .keyID("0001")
```

```
                .customParam("iat", Instant.now().getEpochSecond()).build();
#3  JWSObject jwsObject = new JWSObject(header, payload);

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

    return jwsObject;
}
```

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

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

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

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

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

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

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

    return jweObject;
}
```

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

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

How to Decrypt Message and Verify Signature of an Incoming Message

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

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

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

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

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

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

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

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

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

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

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

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

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

Summary

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

How to Make an API Request

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

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

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

Make Your API Request with Plain Messages

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

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

Step 1. Run this command on your platform:

POST GET

```
#1 curl -X POST "https://ws-api-platform-pprd.business.hsbc.co.uk/glcm-mobilecoll-mcuk-ea-merchar
#2  -H "message_encrypt: false"
#3  -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#4  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#5  -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
#6  -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
#7  -H "Content-Type: application/json"
#8  -d "{ \"txnRef\": \"PAY-QJZV956664\", \"merId\": \"42298549900001\"}"
```

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

```
#1 curl -X GET "https://ws-api-platform-pprd.business.hsbc.co.uk/glcm-mobilecoll-mcuk-ea-merchant
#2  -H "message_encrypt: false"
#3  -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#4  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#5  -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
#6  -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
#7  -H "Content-Type: application/json"
```

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

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

Making API Request with Message Encryption

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

POST GET

```
#1 curl -X POST "https://ws-api-platform-pprd.business.hsbc.co.uk/glcm-mobilecoll-mcuk-ea-merchar
#2  -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#3  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#4  -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
#5  -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
#6  -H "Content-Type: application/json"
#7  -d "eyJraWQiOiIwMDAxIiwiZW5jIjoiQTEyOEdDTSIsImFsZyI6IlJTQS1PQUVQLTI1NiJ9.W4nobHoVXUMOXGM5I-W
```

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

```
#1 curl -X GET "https://ws-api-platform-pprd.business.hsbc.co.uk/glcm-mobilecoll-mcuk-ea-merchant
#2  -H "Authorization: Basic eW91cl91c2VybmFtZTp5b3VyX3Bhc3N3b3Jk"
#3  -H "x-HSBC-client-id: 8b915a4f5b5047f091f210e2232b5ced"
#4  -H "x-HSBC-client-secret: 1bb456a541dc416dB6016B5F9583C606"
#5  -H "x-HSBC-msg-encrypt-id: 42298549900001+0001+0002"
#6  -H "Content-Type: application/json"
```

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

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

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

Data Type Overview

Data Type Control:

Data Type	Allowed Characters	Definition & Important Notice
String *(For general field)*	Alphanumeric and Symbols	General field means field which is **NOT** a critical field. HSBC system will execute characters checking upon all string fields we received in order to tackle security vulnerability, such as Cross-site Scripting. Yet, we recommend you to try use Alphanumeric only for most cases.
String *(For critical field)*	`0-9` `a-z` `A-Z` `-` `_` `.`	Critical field is used to be either a key or search criteria in HSBC backend system and hence tight restriction is applied to the allowed characters. Moreover, the starting and ending space of the string value will be trimmed before stored in HSBC system. For example, string `" example 12 34 "` will be trimmed to `"example 12 34"`. **List of Critical Fields:** All `id`(s)
Integer	`0-9`	Instead of having Max Length check for String, integer range will be checked, e.g. `0 ≤ x ≤ 9999`

Field Mandatory Control:

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

Time Zone Control:

Aspect	Format	Definition & Important Notice
In Request Message	`yyyy-MM-dd'T'HH:mm:ssZ`	Time zone is expected to be `GMT+8` (Malaysia standard time). Merchant is required to perform any necessary time zone conversion before submit request if needed.
In Response Message	`yyyy-MM-dd'T'HH:mm:sszhh: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://ws-api-platform-pprd.business.hsbc.co.uk/**

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

Message Encryption Questions

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

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

Javascript Object Signing and Encryption (JOSE) Framework Questions

Where can I get more information about JOSE Framework?

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

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

Where can I download JOSE libraries for development?

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

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

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

Orders

Create Orders and link them to Payments. Order creation is the first and an important step as it helps you associate every payment with an order. Orders and payments can be created in one-go or seperately.

Orders

Create an Order

`POST` `/orders`

DESCRIPTION

This endpoint creates an Order.

To facilitate the checkout process of an e-commerce sale, this endpoint can offer a faster way to create `Order` and `Payment` in one-go by expanding the API operation to multiple entities. Please see the details as follows.

Nevertheless, merchant can still choose to create `Order` and `Payment` separately which may fit more on some other use cases such as bill payment or bulk processing.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-profileid required in header	[Profile ID]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
$expand: string[] in query	The `$expand` system query option specifies the related resources to be included in line with the original resource. Available Value: `payment`

Request Content-Types: application/json

Request Example

Submit without Payment | Submit with Hosted Payment | Submit with Direct Payment


```
{
  "txn_reference": "ORDER-1234QWER",
  "account_name": "internet",
  "amount": 1000,
  "currency": "GBP",
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 900,
      "unit": 1,
      "vat": 100,
      "subAmt": 1000
    }
  ],
  "metadata": {
    "note_1": "Customer is a VIP",
    "note_2": "In-store credit is used"
  }
}
```

POST /orders?$expand=payment

Use Case: Create objects `Order` and `Payment` in one go. This is recommended for easing the no. of API calls.

How-to: Include `Payment` object in the request body

POST /orders

Use Case: Only `Order` will be created. This gives the flexibility to link to a `Payment` object at a later time

How-to: Exclude `Payment` object in the request body. After an `Order` is created, create `Payment` and link `Order` by **POST /orders/{id}/payment**

enable_payment_url: string
in query

Available Value: `Y`

POST /orders?enable_payment_url=Y

Use Case: Enable to return Payment URL Link in response message: `PATH: $.response.order.payments[0].payment_method.hosted_payment.payment_link.url`

NOTE: Default value is `N`. Suggest to enable only when you need it since generating a Payment URL link needs extra computing resource.

REQUEST BODY

OrderInput

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


```
{
  "txn_reference": "ORDER-1234QWER",
  "account_name": "internet",
  "amount": 1000,
  "currency": "GBP",
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 900,
      "unit": 1,
      "vat": 100,
      "subAmt": 1000
    }
  ],
  "metadata": {
    "note_1": "Customer is a VIP",
    "note_2": "In-store credit is used"
  },
  "payment": {
    "payment_method": {
      "hosted_payment": {
        "url": {
          "page_redirect": "https://merchant.com/returnPage",
          "notification": "https://merchant.com/returnStatus"
        },
        "payment_option": [
          "cards",
          "paypal",
          "wechatpay"
        ],
        "billing": {
          "first_name": "james",
          "last_name": "mason",
          "email": "james.mason@example.com",
          "street1": "Flat 123",
          "street2": "House 456",
          "street3": "Unit 4",
          "city": "Halifax",
          "postal_code": "WS 9HR",
          "country": "826"
        }
      }
    }
  }
}
```



```
{
  "txn_reference": "ORDER-1234QWER",
  "account_name": "internet",
  "amount": 1000,
  "currency": "GBP",
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "A",
      "unitAmt": 900,
      "unit": 1,
      "vat": 100,
      "subAmt": 1000
    }
  ],
  "metadata": {
    "note_1": "Customer is a VIP",
    "note_2": "In-store credit is used"
  },
  "payment": {
    "payment_method": {
      "direct_payment": {
        "payment_option": "applepay",
        "token": "<payment token>"
      }
    }
  }
}
```


RESPONSES

200 OK
OrderOutput

Successful operation.

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

400 Bad Request
Exception

Missing or invalid Parameters.

403 Forbidden

Authorization credentials are missing or invalid.

404 Not Found

Empty resource/resource not found.

500 Internal Server Error

The request failed due to an internal error.

Response Content-Types: application/json

Response Example (200 OK)

Submit without Payment | Submit with Hosted Payment | Submit with Direct Payment


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "order": {
      "id": "ORDER-1234QWER",
      "txn_reference": "ORDER-1234QWER",
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "account_name": "internet",
      "amount": 1000,
      "currency": "GBP",
      "items": [
        {
          "product_name": "Product Item 1",
          "product_id": "A",
          "unitAmt": 900,
          "unit": 1,
          "vat": 100,
          "subAmt": 1000
        }
      ],
      "metadata": {
        "note_1": "Customer is a VIP",
        "note_2": "In-store credit is used"
      },
      "links": null
    }
  }
}
```



```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "order": {
      "id": "ORDER-1234QWER",
      "txn_reference": "ORDER-1234QWER",
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "account_name": "internet",
      "amount": 1000,
      "currency": "GBP",
      "items": [
        {
          "product_name": "Product Item 1",
          "product_id": "A",
          "unitAmt": 900,
          "unit": 1,
          "vat": 100,
          "subAmt": 1000
        }
      ],
      "metadata": {
        "note_1": "Customer is a VIP",
        "note_2": "In-store credit is used"
      },
      "payments": [
        {
          "id": "PAYMENT-5678TYUI",
          "pasref": null,
          "created_at": "2021-06-11T12:10:25Z",
          "last_modified": null,
          "amount": null,
          "currency": null,
          "status": "initiated",
          "payment_method": {
            "hosted_payment": {
              "payment_link": {
                "form": "<Encoded_Redirect_Submit_Form>",
```

```
                "iframe_form": "<Encoded_Redirect_Submit_Form>",
                "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-
4d25-8c4b-7d48b3a863ec"
              },
              "url": {
                "page_redirect": "https://merchant.com/returnPage",
                "notification": "https://merchant.com/returnStatus"
              },
              "payment_option": null,
              "billing": {
                "first_name": "james",
                "last_name": "mason",
                "email": "james.mason@example.com",
                "street1": "Flat 123",
                "street2": "House 456",
                "street3": "Unit 4",
                "city": "Halifax",
                "postal_code": "W5 9HR",
                "country": "826"
              }
            }
          },
          "metadata": null,
          "links": null
        }
      ],
      "links": null
    }
  }
}
```

```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "order": {
      "id": "ORDER-1234QWER",
      "txn_reference": "ORDER-1234QWER",
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "account_name": "internet",
      "amount": 1000,
      "currency": "GBP",
      "items": [
        {
          "product_name": "Product Item 1",
          "product_id": "A",
          "unitAmt": 900,
          "unit": 1,
          "vat": 100,
          "subAmt": 1000
        }
      ],
      "metadata": {
        "note_1": "Customer is a VIP",
        "note_2": "In-store credit is used"
      },
      "payments": [
        {
          "id": "14627849160897986",
          "pasref": "14627849160897986",
          "created_at": "2021-06-11T14:10:25Z",
          "last_modified": null,
          "amount": 1000,
          "currency": "GBP",
          "status": "pending",
          "payment_method": {
            "direct_payment": {
              "payment_option": "applepay",
              "token": "<payment token>",
              "card": {
                "brand": "VISA",
                "authcode": "12345",
                "mcn": "401200******1112",
                "cvv_result": "MATCHED",
                "dcc": {
                  "amount": 1324,
                  "currency": "USD",
                  "ccp": "FEXCO",
                  "fx_rate": 1.3244,
                  "margin_percentage": 3.75
                }
              }
            }
          },
          "metadata": null,
          "links": [
            {
              "href": "/payments/@payment_id/refund",
              "id": {
                "payment_id": "14627849160897986"
              },
              "rel": "refund",
              "method": "POST"
            },
            {
              "href": "/payments/@payment_id",
              "id": {
                "payment_id": "14627849160897986"
              },
              "rel": "self",
              "method": "PATCH"
            }
          ]
        }
      ],
      "links": null
    }
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```


Orders

Retrieve a particular Order by ID

GET /orders/{id}

DESCRIPTION

This endpoint retrieves the details of a particular Order.

Retrieval of other related-objects such as Payment and Refund in one single action is possible by expanding the API operation to multiple entities. This can benefit merchant by reducing the number of API calls. However, it may hit performance issue if one particular Order associates a long list of Payments or Refunds, so please choose to use this feature wisely.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]	

x-hsbc-profileid required in header
[Profile ID]

x-hsbc-msg-encrypt-id required in header
[Merchant ID]+[JWS ID]+[JWE ID]

Content-Type required in header
application/json

id: string required in path
Unique `id` of `order`

Data Encryption is enforced.

$expand: string[] in query
The `$expand` system query option specifies the related resources to be included in line with the original resource.

Available Values: `payment` `refund`

GET /orders/{id}

Only `Order` will be returned in response, all other related-objects will be associated in `links`

GET /orders/{id}?$expand=payment

`Order` and `Payment` will be returned in response, all other related-objects will be associated in `links`

GET /orders/{id}?$expand=payment/refund

`Order`, `Payment` and `Refund` will be returned in response

RESPONSES

200 OK
OrderOutput
Successful operation.

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

400 Bad Request
Exception
Missing or invalid Parameters.

403 Forbidden
Authorization credentials are missing or invalid.

404 Not Found
Empty resource/resource not found.

500 Internal Server Error
The request failed due to an internal error.

Response Content-Types: application/json

Response Example (200 OK)

Order only | Order + Payment | Order + Payment + Refund


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "order": {
      "id": "ORDER-1234QWER",
      "txn_reference": "ORDER-1234QWER",
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "account_name": "internet",
      "amount": 1000,
      "currency": "GBP",
      "items": [
        {
          "product_name": "Product Item 1",
          "product_id": "A",
          "unitAmt": 900,
          "unit": 1,
          "vat": 100,
          "subAmt": 1000
        }
      ],
      "metadata": {
        "note_1": "Customer is a VIP",
        "note_2": "In-store credit is used"
      },
      "links": [
        {
          "href": "/payments/@payment_id",
          "id": {
            "payment_id": "14627849160897986"
          },
          "rel": "payment",
          "method": "GET"
        }
      ]
    }
  }
}
```



```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "order": {
      "id": "ORDER-1234QWER",
      "txn_reference": "ORDER-1234QWER",
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "account_name": "internet",
      "amount": 1000,
      "currency": "GBP",
      "items": [
        {
          "product_name": "Product Item 1",
          "product_id": "prod-9ijn8uhb",
          "unitAmt": 900,
          "unit": 1,
          "vat": 100,
          "subAmt": 1000
        }
      ],
      "metadata": null,
      "payments": [
        {
          "id": "PAYMENT-5678TYUI",
          "pasref": "14627849160897986",
          "created_at": "2021-06-11T14:10:25Z",
          "last_modified": "2021-06-12T14:10:25Z",
          "amount": 1000,
          "currency": "GBP",
          "status": "batched",
          "payment_method": {
            "hosted_payment": {
              "payment_link": {
                "form": "<Encoded_Redirect_Submit_Form>",
                "iframe_form": "<Encoded_Redirect_Submit_Form>",
                "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-
4d25-8c4b-7d48b3a863ec"
              },
              "url": {
                "page_redirect": "https://merchant.com/returnPage",
                "notification": "https://merchant.com/returnStatus"
              },
              "payment_option": "cards",
              "billing": {
                "first_name": "james",
                "last_name": "mason",
                "email": "james.mason@example.com",
                "street1": "Flat 123",
                "street2": "House 456",
                "street3": "Unit 4",
                "city": "Halifax",
                "postal_code": "W5 9HR",
                "country": "826"
              },
              "card": {
                "brand": "VISA",
                "authcode": "12345",
                "mcn": "401200******1112",
                "cvv_result": "MATCHED",
                "dcc": {
                  "amount": 1324,
                  "currency": "USD",
                  "ccp": "FEXCO",
                  "fx_rate": 1.3244,
                  "margin_percentage": 3.75
                }
              }
            }
          },
```

```
          "metadata": null,
          "links": [
            {
              "href": "/refunds/@refund_id",
              "id": {
                "refund_id": "16219383951512048"
              },
              "rel": "refund",
              "method": "GET"
            },
            {
              "href": "/payments/@payment_id/refund",
              "id": {
                "payment_id": "PAYMENT-5678TYUI"
              },
              "rel": "refund",
              "method": "POST"
            },
            {
              "href": "/payments/@payment_id",
              "id": {
                "payment_id": "PAYMENT-5678TYUI"
              },
              "rel": "self",
              "method": "PATCH"
            }
          ]
        }
      ],
      "links": null
    }
  }
}
```

```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "order": {
      "id": "ORDER-1234QWER",
      "txn_reference": "ORDER-1234QWER",
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "account_name": "internet",
      "amount": 1000,
      "currency": "GBP",
      "items": [
        {
          "product_name": "Product Item 1",
          "product_id": "prod-91jn8uhb",
          "unitAmt": 900,
          "unit": 1,
          "vat": 100,
          "subAmt": 1000
        }
      ],
      "metadata": null,
      "payments": [
        {
          "id": "PAYMENT-5678TYUI",
          "pasref": "14627849160897986",
          "created_at": "2021-06-11T14:10:25Z",
          "last_modified": "2021-06-12T14:10:25Z",
          "amount": 1000,
          "currency": "GBP",
          "status": "batched",
          "payment_method": {
            "hosted_payment": {
              "payment_link": {
                "form": "<Encoded_Redirect_Submit_Form>",
                "iframe_form": "<Encoded_Redirect_Submit_Form>",
                "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-
4d25-8c4b-7d48b3a863ec"
              },
              "url": {
                "page_redirect": "https://merchant.com/returnPage",
                "notification": "https://merchant.com/returnStatus"
              },
              "payment_option": "cards",
              "billing": {
                "first_name": "james",
                "last_name": "mason",
                "email": "james.mason@example.com",
                "street1": "Flat 123",
                "street2": "House 456",
                "street3": "Unit 4",
                "city": "Halifax",
                "postal_code": "W5 9HR",
                "country": "826"
              },
              "card": {
                "brand": "VISA",
                "authcode": "12345",
                "mcn": "401200******1112",
                "cvv_result": "MATCHED",
                "dcc": {
                  "amount": 1324,
                  "currency": "USD",
                  "ccp": "FEXCO",
                  "fx_rate": 1.3244,
                  "margin_percentage": 3.75
                }
              }
            }
          },
          "metadata": null,
          "refunds": [
            {
              "id": "162193839515120 48",
              "pasref": "16219383951512048",
              "created_at": "2021-06-11T14:10:25Z",
              "last_modified": "2021-06-12T14:10:25Z",
              "amount": 1000,
              "currency": "GBP",
              "status": "pending",
              "metadata": null
            }
          ],
          "links": [
            {
              "href": "/payments/@payment_id/refund",
              "id": {
                "payment_id": "PAYMENT-5678TYUI"
              },
              "rel": "refund",
              "method": "POST"
            },
            {
              "href": "/payments/@payment_id",
              "id": {
                "payment_id": "PAYMENT-5678TYUI"
              },
              "rel": "self",
              "method": "PATCH"
            }
          ]
        }
      ],
      "links": null
    }
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```

Payments

You can accept payments from your customers by integrating your website or app with Hosted Payment Page or Direct Payment which serves different payment methods.

Please see the following checkout solutions:

> **IMPORTANT NOTE:**
> Condition may apply to the availability of each checkout solution and its associated payment options, please check with our support team for details.

Hosted Payment | Direct Payment

Hosted Payment Page

Overview	The Hosted Payment Page (HPP) is a PCI DSS v3.2 compliant redirect solution, allowing you capture card data without having to worry about the PCI overhead associated with a traditional API integration.
Supported Payment Options	• **Debit / Credit Cards** - Card Data is captured securely inside HPP • **Paypal** - Redirect to Paypal Checkout page • **WeChat Pay** - Payment QR Code will be displayed in a redirected page • **Test Pay** - A payment simulator available in Sandbox environment for testing purpose.

Integration Methods	• **Full-Page Redirection**
	A full-page redirect will be performed. A pre-rendered auto-executable HTML submit form will be returned in response field
	`PATH: $.response.payment.payment_method.hosted_payment.payment_link.form`
	This is a sample: (escape character is removed)
	`<script language="javascript">window.onload=funct`
	`<form id="pay_form" name="pay_form" action="https`
	`<input name="TIMESTAMP" type="hidden" id="TIMESTA`
	`<input name="MERCHANT_ID" type="hidden" id="MERCH`
	`<input name="ACCOUNT" type="hidden" id="ACCOUNT"`
	`/* More Input Fields Here... */`
	`</form>`
	• **iFrame/WebView Optimization**
	You can render the payment form using your own embedded iFrame or WebView, or as a Lightbox. The HTML submit form is similar but with some additional input fields. It will be returned in response field
	`PATH: $.response.payment.payment_method.hosted_payment.payment_link.iframe_form`
	• **Pay by Link**
	A Payment URL link you can embed directly in your website, an SMS or an email. The link will be returned in response field
	`PATH: $.response.payment.payment_method.hosted_payment.payment_link.url`
	> **INFORMATION:** Payment result of all of the above methods will be returned in Payment Webhook.

Direct Payment

Overview	Payment will be processed simultaneously in an API call.
Supported Payment Options	• **Apple Pay** - Card Data Encryption is handled by Apple. • **Google Pay** - Card Data Encryption is handled by Google.

Integration Method	**Step 1. Obtain a Payment Token** - Payment Token is a token provided by Apple or Google that merchant can use it to bind a particular customer's card detail stored securely in Apple's or Google's server. Please visit Apple Pay and Google Pay developer site to learn how to obtain a Payment Token by calling their APIs.
	`/* A Payment Token example from Apple Pay */`
	`{"data":"Sk1pYr1MYT4SMEKLibAPF342LZy3GEZJz0OZ0LdM`
	`/* A Payment Token example from Google Pay */`
	`{"signature":"MEUCIHI37nu9JakubEtif26PtEvw9lUC4kB`
	Step 2. Process Payment - Put the Payment Token in the corresponding API request message and HSBC Omni Collect will process the payment simultaneously as a typical card payment. A successful or a failed payment will be returned in the synchronous API response message.

Payments

Create a Payment for an Order

`POST` `/orders/{id}/payment`

DESCRIPTION

This endpoint creates a Payment which links to a specific Order.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-profileid required in header	[Profile ID]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
id: string required in path	Unique `id` of `order` *Data Encryption is enforced.*
enable_payment_url: string in query	Available Value: `Y` **POST /orders/{id}/payment?enable_payment_url=Y** **Use Case:** Enable to return Payment URL Link in response message: `PATH: $.response.order.payments[0].payment_method.hosted_payment.payment_link.url` **NOTE:** Default value is `N`. Suggest to enable only when you need it since generating a Payment URL link needs extra computing resource.

Request Content-Types: application/json

Request Example

Submit with Hosted Payment | Submit with Direct Payment

```
{
  "payment_method": {
    "hosted_payment": {
      "url": {
        "page_redirect": "https://merchant.com/returnPage",
        "notification": "https://merchant.com/returnStatus"
      },
      "payment_option": [
        "cards",
        "paypal",
        "wechatpay"
      ],
      "billing": {
        "first_name": "james",
        "last_name": "mason",
        "email": "james.mason@example.com",
        "street1": "Flat 123",
        "street2": "House 456",
        "street3": "Unit 4",
        "city": "Halifax",
        "postal_code": "W5 9HR",
        "country": "826"
      }
    }
  }
}
```

```
{
  "payment_method": {
    "direct_payment": {
      "payment_option": "applepay",
```

REQUEST BODY

PaymentInput

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

RESPONSES

200 OK Successful operation.
PaymentOutput

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

400 Bad Request Missing or invalid Parameters.
Exception

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.


```
      "token": "<payment token>"
    }
  }
}
```


Response Content-Types: application/json

Response Example (200 OK)

Submit with Hosted Payment | Submit with Direct Payment


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "payment": {
      "id": "PAYMENT-567BTYUI",
      "pasref": null,
      "created_at": "2021-06-11T12:10:25Z",
      "last_modified": null,
      "amount": null,
      "currency": null,
      "status": "initiated",
      "payment_method": {
        "hosted_payment": {
          "payment_link": {
            "form": "<Encoded_Redirect_Submit_Form>",
            "iframe_form": "<Encoded_Redirect_Submit_Form>",
            "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-4d25-
8c4b-7d48b3a863ec"
          },
          "url": {
            "page_redirect": "https://merchant.com/returnPage",
            "notification": "https://merchant.com/returnStatus"
          },
          "payment_option": null,
          "billing": {
            "first_name": "james",
            "last_name": "mason",
            "email": "james.mason@example.com",
            "street1": "Flat 123",
            "street2": "House 456",
            "street3": "Unit 4",
            "city": "Halifax",
            "postal_code": "W5 9HR",
            "country": "826"
          }
        }
      },
      "metadata": null,
      "links": null
    },
    "links": [
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```



```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "payment": {
      "id": "14627849160897986",
      "pasref": "14627849160897986",
      "created_at": "2021-06-11T14:10:25Z",
      "last_modified": null,
      "amount": 1000,
      "currency": "GBP",
      "status": "pending",
      "payment_method": {
        "hosted_payment": {
          "direct_payment": {
            "payment_option": "applepay",
            "token": "<payment token>",
            "card": {
              "brand": "VISA",
              "authcode": "12345",
              "mcn": "401200******1112",
              "cvv_result": "MATCHED",
              "dcc": {
                "amount": 1324,
                "currency": "USD",
                "ccp": "FEXCO",
                "fx_rate": 1.3244,
                "margin_percentage": 3.75
              }
            }
          }
        }
      },
      "metadata": null,
      "links": [
        {
          "href": "/payments/@payment_id/refund",
          "id": {
            "payment_id": "14627849160897986"
          },
          "rel": "refund",
          "method": "POST"
        },
        {
          "href": "/payments/@payment_id",
          "id": {
            "payment_id": "14627849160897986"
          },
          "rel": "self",
          "method": "PATCH"
        }
      ]
    },
    "links": [
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```

Payments

Retrieve a particular Payment by ID

`GET` `/payments/{id}`

DESCRIPTION

This endpoint retrieves the details of a particular Payment.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-profileid required in header	[Profile ID]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
id: string required in path	Unique `id` of `payment` *Data Encryption is enforced.*

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "payment": {
      "id": "14627849160897986",
      "pasref": "14627849160897986",
      "created_at": "2021-06-11T14:10:25Z",
      "last_modified": "2021-06-12T14:10:25Z",
      "amount": 1000,
      "currency": "GBP",
      "status": "batched",
      "payment_method": {
        "hosted_payment": {
          "payment_link": {
            "form": "<Encoded_Redirect_Submit_Form>",
            "iframe_form": "<Encoded_Redirect_Submit_Form>",
            "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-4d25-
8c4b-7d48b3a863ec"
          },
          "url": {
            "page_redirect": "https://merchant.com/returnPage",
            "notification": "https://merchant.com/returnStatus"
          },
          "payment_option": "cards",
          "billing": {
            "first_name": "james",
            "last_name": "mason",
            "email": "james.mason@example.com",
            "street1": "Flat 123",
            "street2": "House 456",
            "street3": "Unit 4",
            "city": "Halifax",
            "postal_code": "W5 9HR",
            "country": "826"
          },
          "card": {
            "brand": "VISA",
            "authcode": "12345",
            "mcn": "401200******1112",
            "cvv_result": "MATCHED",
            "dcc": {
              "amount": 1324,
              "currency": "USD",
              "ccp": "FEXCO",
              "fx_rate": 1.3244,
              "margin_percentage": 3.75
            }
          }
        }
      },
      "metadata": null,
      "links": [
        {
          "href": "/refunds/@refund_id",
          "id": {
            "refund_id": "1002345678999"
          },
          "rel": "refund",
          "method": "GET"
        },
        {
          "href": "/refunds/@refund_id",
          "id": {
            "refund_id": "7778889996665"
          },
          "rel": "refund",
          "method": "GET"
        },
        {
          "href": "/payments/@payment_id/refund",
          "id": {
            "payment_id": "14627849160897986"
          },
          "rel": "refund",
          "method": "POST"
        },
        {
          "href": "/payments/@payment_id",
          "id": {
            "payment_id": "14627849160897986"
          },
          "rel": "self",
          "method": "PATCH"
        }
      ]
    },
    "links": [
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```

Payments

Update a Payment such as Void a Payment or Add Notes

`PATCH` `/payments/{id}`

DESCRIPTION

A payment can be voided by updating its `status` or modified by adding/replacing its `metadata`.

> **NOTE:**
> Only an unsettled payment can be voided.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-profileid required in header	[Profile ID]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
id: string required in path	Unique `id` of `payment` *Data Encryption is enforced.*

REQUEST BODY

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

RESPONSES

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

Request Content-Types: application/json

Request Example

```
{
  "status": "void",
  "metadata": {
    "customer_id": "12345",
    "customer_comment": "engrave customer's name on product"
  }
}
```

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "system": {
    "messageId": "89817674-da0O-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "payment": {
      "id": "PAYMENT-5678TYUI",
      "pasref": "14627849160897986",
      "created_at": "2021-06-11T14:10:25Z",
      "last_modified": "2021-06-12T14:10:25Z",
      "amount": 1000,
      "currency": "GBP",
      "status": "voided",
      "payment_method": {
        "hosted_payment": {
          "payment_link": {
            "form": "<Encoded_Redirect_Submit_Form>",
            "iframe_form": "<Encoded_Redirect_Submit_Form>",
            "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-4d25-
8c4b-7d48b3a863ec"
          },
          "url": {
            "page_redirect": "https://merchant.com/returnPage",
            "notification": "https://merchant.com/returnStatus"
          },
          "payment_option": "cards",
          "billing": {
            "first_name": "james",
            "last_name": "mason",
            "email": "james.mason@example.com",
            "street1": "Flat 123",
            "street2": "House 456",
            "street3": "Unit 4",
            "city": "Halifax",
            "postal_code": "W5 9HR",
            "country": "826"
          },
          "card": {
            "brand": "VISA",
            "authcode": "12345",
            "pcn": "401200******1112",
            "cvv_result": "MATCHED",
            "dcc": {
              "amount": 1324,
              "currency": "USD",
              "ccp": "FEXCO",
              "fx_rate": 1.3244,
              "margin_percentage": 3.75
            }
          }
        }
      },
      "metadata": {
        "customer_id": "12345",
        "customer_comment": "engrave customer's name on product"
      },
      "links": [
        {
          "href": "/refunds/@refund_id",
          "id": {
            "refund_id": "1002345678999"
          },
          "rel": "refund",
          "method": "GET"
        },
        {
          "href": "/refunds/@refund_id",
          "id": {
            "refund_id": "7778889996665"
          },
          "rel": "refund",
          "method": "GET"
        },
        {
          "href": "/payments/@payment_id/refund",
          "id": {
            "payment_id": "PAYMENT-5678TYUI"
          },
          "rel": "refund",
          "method": "POST"
        },
        {
          "href": "/payments/@payment_id",
          "id": {
            "payment_id": "PAYMENT-5678TYUI"
          },
          "rel": "self",
          "method": "PATCH"
        }
      ]
    },
    "links": [
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-da0O-4883",
```

```
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```


Refunds

You can make full or partial refunds to customers. Only a settled payment can be refunded.

Refunds

Create a Refund for a Payment

POST `/payments/{id}/refund`

DESCRIPTION

This endpoint creates a Refund which links to a specific Payment.

REQUEST PARAMETERS

Authorization required in header	BASIC [Base64-encoded Credential]
x-hsbc-profileid required in header	[Profile ID]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
id: string required in path	Unique `id` of `payment` *Data Encryption is enforced.*

REQUEST BODY

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

RESPONSES

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

Request Content-Types: application/json

Request Example


```
{
  "amount": 1000,
  "currency": "GBP"
}
```


Response Content-Types: application/json

Response Example (200 OK)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "refund": {
      "id": "162193839512048",
      "pasref": "162193839512048",
      "created_at": "2021-06-11T14:10:25Z",
      "last_modified": null,
      "amount": 1000,
      "currency": "GBP",
      "status": "pending",
      "metadata": {
        "refund_reason": "Product is damaged"
      }
    },
    "links": [
      {
        "href": "/payment/@payment_id",
        "id": {
          "order_id": "14627849160897986"
        },
        "rel": "payment",
        "method": "GET"
      },
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```


Refunds

Retrieve a particular Refund by ID

GET `/refunds/{id}`

DESCRIPTION

This endpoint retrieves the details of a particular Refund.

REQUEST PARAMETERS

Authorization	

required in header	BASIC [Base64-encoded Credential]
x-hsbc-profileid required in header	[Profile ID]
x-hsbc-msg-encrypt-id required in header	[Merchant ID]+[JWS ID]+[JWE ID]
Content-Type required in header	application/json
id: string required in path	Unique `id` of `refund` *Data Encryption is enforced.*

RESPONSES

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

Response Content-Types: application/json

Response Example (200 OK)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "200",
    "returnReason": "Successful operation",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "refund": {
      "id": "16219383951512048",
      "pasref": "16219383951512048",
      "created_at": "2021-06-11T14:10:25Z",
      "last_modified": null,
      "amount": 1000,
      "currency": "GBP",
      "status": "pending",
      "metadata": {
        "refund_reason": "Product is damaged"
      }
    },
    "links": [
      {
        "href": "/payment/@payment_id",
        "id": {
          "order_id": "14627849160897986"
        },
        "rel": "payment",
        "method": "GET"
      },
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```


Response Example (400 Bad Request)


```
{
  "system": {
    "messageId": "89817674-daOO-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
  }
}
```


Webhooks

What is a Webhook

Webhooks (Web Callback, HTTP Push API or Reverse API) is one way one web application can send information to another application in real-time when a specific event happens.

You can use HSBC Omni Collect Webhooks to receive notifications when a specific event occurs. When one of these events is triggered, we send an HTTP POST payload in encrypted JSON to the webhook's configured URL.

Set Up

Entity	Event	URL Set Up
Payments	• payment.captured • payment.failed	Define in `$.payment_method.hosted_payment .url.notification` when creating a `payment` resource

Exception Handling

Every event that receives a non-2xx response is considered as an event delivery failure and retry mechanism will be triggered. Up to 4 retries will be triggered in every 2 minutes. Maximum 5 calls including the 1st attempt.

Idempotency

There could be scenarios where your endpoint might receive the same webhook multiple times. This could happen as an expected behaviour such as the retry mechanism or any other exceptional behaviour such as network problem.

To handle duplicate webhooks, we offer a unique webhook ID `x-hsbc-webhook-id` where you can find it in the HTTP header on every webhook.

Webhooks

Webhooks for Payments

POST `/<Callback URL predefined by Merchant>`

DESCRIPTION

The table below lists the Webhook events available for payments.

Webhook Event	Definition
payment.captured	Triggered when a payment is successfully captured.
payment.failed	Triggered when a payment fails.

REQUEST PARAMETERS

x-hsbc-webhook-id required in header	UUID

Request Content-Types: text/plain

Request Example


```
{
  "webhook": {
```

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

REQUEST BODY

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


```
    "event": "payment.captured",
    "entities": [
      "payment"
    ]
  },
  "payload": {
    "payment": {
      "id": "14627849160897986",
      "pasref": "14627849160897986",
      "created_at": "2021-06-11T14:10:25Z",
      "last_modified": null,
      "amount": 1000,
      "currency": "GBP",
      "status": "pending",
      "payment_method": {
        "hosted_payment": {
          "payment_link": {
            "form": "<Encoded_Redirect_Submit_Form>",
            "iframe_form": "<Encoded_Redirect_Submit_Form>",
            "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-4d25-
8c4b-7d48b3a863ec"
          },
          "url": {
            "page_redirect": "https://merchant.com/returnPage",
            "notification": "https://merchant.com/returnStatus"
          },
          "payment_option": "paypal",
          "billing": {
            "first_name": "james",
            "last_name": "mason",
            "email": "james.mason@example.com",
            "street1": "Flat 123",
            "street2": "House 456",
            "street3": "Unit 4",
            "city": "Halifax",
            "postal_code": "W5 9HR",
            "country": "826"
          },
          "paypal": {
            "TransactionID": "0F513533U4356815L",
            "SecureMerchantAccountID": "XXXXXXXXXXXX",
            "PayPalAccountID": "testmerchant999@exmaple.com"
          }
        }
      },
      "metadata": {
        "customer_id": "12345",
        "customer_comment": "engrave customer's name on product"
      },
      "links": null
    }
  }
}
```


RESPONSES

200 OK Callback — 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"
}
```


Schema Definitions

OrderInput: object

PROPERTIES

txn_reference: string range: (up to 50 chars) required
Unique Transaction Reference defined by Merchant, it will be appeared to be the unique `id` of an `order`.

account_name: string range: (up to 30 chars) required
Merchant account name given during the merchant profile setup.

amount: integer range: 1 ≤ x ≤ 9999999999 required
Order Amount

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

currency: string enum: [GBP, EUR, USD] range: (up to 3 chars) required
Order Currency

items: Array< Item > range: (up to 20 objects) required
List of Product Descriptions in the basket

metadata: Metadata range: (up to 20 objects) optional
Key-value pair that can be used to store additional information about the entity.

payment: PaymentInput conditional
Required when query parameter `expand[]=payment` is opted in. Including this will create `order` and `payment` resources in one-go.

Example


```
{
  "txn_reference": "ORDER-1234QWER",
  "account_name": "internet",
  "amount": 1000,
  "currency": "GBP",
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "prod-9ijn8uhb",
      "unitAmt": 900,
      "unit": 1,
      "vat": 100,
      "subAmt": 1000
    }
  ],
  "metadata": null,
  "payment": {
    ...See schema PaymentInput for details...
  }
}
```


OrderOutput: object

PROPERTIES

system: System required
response: object optional
Return if it is a HTTP 200 response

PROPERTIES

order: Order required

Example


```
{
  "system": {
    ...See schema System for details...
  },
  "response": {
    "order": {
      ...See schema Order for details...
    }
  }
}
```


Order: object

PROPERTIES

id: string range: (up to 50 chars) required
Unique Entity ID of an Order, technically derived from `txn_reference`

txn_reference: string range: (up to 50 chars) required
Unique Transaction Reference defined by Merchant, it will be appeared to be the unique `id` of an `order`.

created_at: string range: (up to 20 chars) required
Created Time of the Order

last_modified: string range: (up to 20 chars) required
Last Modified Time of the Order

account_name: string range: (up to 30 chars) required
Merchant account name given during the merchant profile setup.

amount: integer range: 1 ≤ x ≤ 9999999999 required
Order Amount

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

currency: string enum: [GBP, EUR, USD] range: (up to 3 chars) required
Order Currency

items: Array< Item > required
List of Product Descriptions in the basket

metadata: Metadata range: (up to 20 objects) optional
Key-value pair that can be used to store additional information about the entity.

Example


```
{
  "id": "ORDER-1234QWER",
  "txn_reference": "ORDER-1234QWER",
  "created_at": "2021-06-11T12:10:25Z",
  "last_modified": "2021-06-12T15:00:25Z",
  "account_name": "internet",
  "amount": 1000,
  "currency": "GBP",
  "items": [
    {
      "product_name": "Product Item 1",
      "product_id": "prod-9ijn8uhb",
      "unitAmt": 900,
      "unit": 1,
      "vat": 100,
      "subAmt": 1000
    }
  ],
  "metadata": null,
  "payments": [
    {
      ...See schema Payment for details...
    }
  ],
  "links": [
    {
      "href": "/payments/@payment_id",
      "id": {
        "payment_id": "14627849160897986"
      },
      "rel": "payment",
      "method": "GET"
    }
```

payments: Array< Payment > conditional
List of all payments linked with this Order, appear if query parameter `expand[]=payment` is opted in.

links: Array< HAL > conditional
List of all related resources. If query parameter `expand[]=payment` is opted out, payments will be related here in HAL format.

PaymentInput: object

PROPERTIES

payment_method: object required
Can be selected either `hosted_payment` or `direct_payment` at one time.

PROPERTIES

hosted_payment: object conditional
Invoke this object if the payment is processed by Hosted Payment.

PROPERTIES

url: object required
Set up URLs used by HPP.

PROPERTIES

page_redirect: string range: (up to 2083 chars) required
URL defined by Merchant for redirecting back to Merchant's website after the payment process is completed in the HPP.

notification: string range: (up to 2083 chars) required
URL defined by Merchant for receiving Payment Webhooks.

payment_option: string[] optional
Opt in Payment Option(s) to be displayed in the HPP, can be multiple. If no value is provided, all possible options will be displayed.

Possible Value	Definition
cards	Credit / Debit Cards
paypal	Paypal
wechatpay	WeChat Pay
testpay	Test Pay (available in sandbox only)

ITEMS

string enum: [cards, paypal, wechatpay, testpay]

billing: object required

PROPERTIES

first_name: string range: (up to 60 chars) required
Customer's first name.The value should be the same as the value that appears on the card.

last_name: string range: (up to 60 chars) required
Customer's last name.The value should be the same as the value that appears on the card.

email: string range: (up to 254 chars) required
Customer's email address

street1: string range: (up to 50 chars) required
First line of the customer's billing address.

street2: string range: (up to 50 chars) required
Second line of the customer's billing address.

street3: string range: (up to 50 chars) required
Third line of the customer's billing address.

city: string range: (up to 40 chars) required
The city of the customer's billing address.

postal_code: string range: (up to 16 chars) required
ZIP or other postal code customer's billing address.

country: string range: (up to 3 chars) required
The country of the customer's billing address. ISO 3166-1 numeric three-digit country code. For example, US = 840.

direct_payment: object conditional
Invoke this object if the payment is processed by Direct Payment.

PROPERTIES

payment_option: string enum: [applepay, googlepay] range: (up to 10 chars) required
Payment Option

Possible Value	Definition
applepay	Apple Pay
googlepay	Google Pay

token: string range: (up to 1000 chars) required
Payment Token from Apple Pay or Google Pay

```
/* A Payment Token example from Apple Pay */
{"data":"Sk1pYr1MYT4SMEKLibAPF342LZy3GEZJzdOZ0LdMK55k3UN02GaAphr/w8YtU+Cr5TZzJgHdCUh/35aEz3ORJrhe

/* Escape character added when cast into JSON String value */
{\"data\":\"Sk1pYr1MYT4SMEKLibAPF342LZy3GEZJzdOZ0LdMK55k3UN02GaAphr\/w8YtU+Cr5TZzJgHdCUh\/35aEz3O
```

```
/* A Payment Token example from Google Pay */
{"signature":"MEUCIHI37nu9JakubEtif26PtEvw9lUC4kBZ+pfqZM0AIw7IAiEA9izE5MEBzZK8tjZ8fjjMzy/VAdVXfgP

/* Escape character added when cast into JSON String value */
{\"signature\":\"MEUCIHI37nu9JakubEtif26PtEvw9lUC4kBZ+pfqZM0AIw7IAiEA9izE5MEBzZK8tjZ8fjjMzy\/VAdV
```

Example

```
{
  "payment_method": {
    "hosted_payment": {
      "url": {
        "page_redirect": "https://merchant.com/returnPage",
        "notification": "https://merchant.com/returnStatus"
      },
      "payment_option": [
        "cards",
        "paypal",
        "wechatpay"
      ],
      "billing": {
        "first_name": "james",
        "last_name": "mason",
        "email": "james.mason@example.com",
        "street1": "Flat 123",
        "street2": "House 456",
        "street3": "Unit 4",
        "city": "Halifax",
        "postal_code": "W5 9HR",
        "country": "826"
      }
    },
    "direct_payment": {
      "payment_option": "applepay",
      "token": "<payment token>"
    }
  }
}
```

PaymentPatch: object

PROPERTIES

status: string enum: [void] range: (up to 50 chars) optional
Payment Status, available to void an unsettled payment by changing the status to `void`

metadata: Metadata range: (up to 20 objects) optional
Key-value pair that can be used to store additional information about the entity.

> **NOTE:**
> Updating metadata in here will replace existing record.

Example

```
{
  "status": "void",
  "metadata": null
}
```

PaymentOutput: object

PROPERTIES

Example

system: System required
response: object optional
Return if it is a HTTP 200 response

PROPERTIES

payment: Payment required
links: Array< HAL > required
List of all related resources.

> ! **NOTE:**
> HAL appeared outside a response entity object usually means they are the parent entities. In this case, `order` is the parent entity of `payment`.

```
{
  "system": {
    ...See schema System for details...
  },
  "response": {
    "payment": {
      ...See schema Payment for details...
    },
    "links": [
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

PaymentWebhook: object

PROPERTIES

webhook: object required

PROPERTIES

event: string enum: [payment.captured, payment.failed] range: (up to 100 chars) required
Event Type

entities: string[] required
The list of Entities contained in this Webhook

ITEMS

string enum: [payment]

payload: object required

PROPERTIES

payment: Payment required

Example

```
{
  "webhook": {
    "event": "payment.captured",
    "entities": [
      "payment"
    ]
  },
  "payload": {
    "payment": {
      ...See schema Payment for details...
    }
  }
}
```

Payment: object

PROPERTIES

id: string range: (up to 50 chars) required
Unique Entity ID of a Payment

pasref: string range: (up to 50 chars) required
Payment Reference. A unique reference generated by Payment Gateway

created_at: string range: (up to 20 chars) required
Created Time of the payment

last_modified: string range: (up to 20 chars) required
Last Modified Time of the payment

amount: integer range: 1 ≤ x ≤ 9999999999 required
Payment Amount

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

currency: string enum: [GBP, EUR, USD] range: (up to 3 chars) required
Payment Currency

status: string enum: [batched, pending, delayed, voided] range: (up to 50 chars) required
The payment/settlment status of the corresponding Payment, only available for Card Payment

Possible Value	Definition
batched	The transaction has been batched and sent for settlement, can no longer be voided. Applies to both refunds and sales.
pending	The transaction has been successfully processed and is awaiting batching (midnight cutoff). Can still be voided prior to batching. Applies to both refunds and sales.
delayed	The transaction has been successfully authorised but is pending a subsequent settle/capture request. Can be voided. Applies only to sales whereby the autosettle field has been set to "0".
voided	the pending/delayed transaction has been voided and will not be batched for settlement. Applies to both refunds and sales.

payment_method: object required
Either `hosted_payment` or `direct_payment` will be appeared at one time.

PROPERTIES

hosted_payment: object conditional
Appear if the payment is processed by Hosted Payment.

PROPERTIES

payment_link: object required
All possible methods that connect to the HPP

PROPERTIES

form: string range: (up to 5120 chars) required
Encoded Redirect Link with all form submit parameters

iframe_form: string range: (up to 5120 chars) required
Encoded Redirect Link with all form submit parameters

url: string range: (up to 1024 chars) required
Pay By Link. For merchants who wish to embed a payment link in an SMS or an email, you can send a request to HPP to create a transaction link to be paid within 24 hours. HPP will respond with the dedicated payment link which can then be sent on to your customer.

url: object required
Set up URLs used by HPP.

PROPERTIES

page_redirect: string range: (up to 2083 chars) required
URL defined by Merchant for redirecting back to Merchant's website after the payment process is completed in the HPP.

notification: string range: (up to 2083 chars) required
URL defined by Merchant for receiving Payment Webhooks.

payment_option: string range: (up to 20 chars) required
Payment Option selected in the HPP payment

Possible Value	Definition
cards	Credit / Debit Cards
paypal	Paypal
wechatpay	WeChat Pay
testpay	Test Pay (available in sandbox only)

billing: object required

PROPERTIES

first_name: string range: (up to 60 chars) required
Customer's first name.The value should be the same as the value that appears on the card.

last_name: string range: (up to 60 chars) required

Example

```
{
  "id": "14627849160897986",
  "pasref": "14627849160897986",
  "created_at": "2021-06-11T14:10:25Z",
  "last_modified": "2021-06-12T14:10:25Z",
  "amount": 1000,
  "currency": "GBP",
  "status": "batched",
  "payment_method": {
    "hosted_payment": {
      "payment_link": {
        "form": "<Encoded_Redirect_Submit_Form>",
        "iframe_form": "<Encoded_Redirect_Submit_Form>",
        "url": "https://pay.sandbox.realexpayments.com/card.html?guid=f82dc878-4752-4d25-8c4b-
7d48b3a863ec"
      },
      "url": {
        "page_redirect": "https://merchant.com/returnPage",
        "notification": "https://merchant.com/returnStatus"
      },
      "payment_option": "cards",
      "billing": {
        "first_name": "james",
        "last_name": "mason",
        "email": "james.mason@example.com",
        "street1": "Flat 123",
        "street2": "House 456",
        "street3": "Unit 4",
        "city": "Halifax",
        "postal_code": "W5 9HR",
        "country": "826"
      },
      "card": {
        ...See schema Card for details...
      },
      "paypal": {
        "TransactionID": "0F513533U4356815L",
        "SecureMerchantAccountID": "XXXXXXXXXXXX",
        "PayPalAccountID": "testmerchant9999@exmaple.com"
      }
    },
    "direct_payment": {
      "payment_option": "applepay",
      "token": "<payment token>",
      "card": {
        ...See schema Card for details...
      }
    }
  },
  "metadata": null,
  "refunds": [
    {
      ...See schema Refund for details...
    }
  ],
  "links": [
    {
      "href": "/refunds/@refund_id",
      "id": {
        "refund_id": "1002345678999"
      },
      "rel": "refund",
      "method": "GET"
    },
    {
      "href": "/payments/@payment_id/refund",
      "id": {
        "payment_id": "14627849160897986"
      },
      "rel": "refund",
      "method": "POST"
    },
    {
      "href": "/payments/@payment_id",
      "id": {
        "payment_id": "14627849160897986"
      },
      "rel": "self",
      "method": "PATCH"
    }
  ]
}
```

Customer's last name.The value should be the same as the value that appears on the card.

email: string range: (up to 254 chars) required
Customer's email address

street1: string range: (up to 50 chars) required
First line of the customer's billing address.

street2: string range: (up to 50 chars) required
Second line of the customer's billing address.

street3: string range: (up to 50 chars) required
Third line of the customer's billing address.

city: string range: (up to 40 chars) required
The city of the customer's billing address.

postal_code: string range: (up to 16 chars) required
ZIP or other postal code customer's billing address.

country: string range: (up to 3 chars) required
The country of the customer's billing address. ISO 3166-1 numeric three-digit country code. For example, US = 840.

card: Card conditional
Card Details. Appear if `"payment_option" = "cards"`

paypal: object conditional
Paypal Details. Appear if `"payment_option" = "paypal"`

PROPERTIES

TransactionID: string range: (up to 50 chars) required
Transaction ID defined by Paypal

SecureMerchantAccountID: string range: (up to 50 chars) required
Secure Merchant Account ID defined by Paypal

PayPalAccountID: string range: (up to 100 chars) required
PayPal Account ID defined by Paypal

direct_payment: object conditional
Appear if the payment is processed by Direct Payment.

PROPERTIES

payment_option: string enum: [applepay, googlepay] range: (up to 10 chars) required
Payment Option

Possible Value	Definition
applepay	Apple Pay
googlepay	Google Pay

token: string range: (up to 1000 chars) required
Payment Token from Apple Pay or Google Pay

card: Card required
Card Details of the corresponding Paypal payment

metadata: Metadata range: (up to 20 objects) optional
Key-value pair that can be used to store additional information about the entity.

refunds: Array< Refund > conditional
List of all Refunds linked with this Payment, appear if query parameter `expand[]=refund` is opted in.

links: Array< HAL > conditional
List of all related resources. If query parameter `expand[]=refund` is opted out, refunds will be related here in HAL format.

RefundInput: object

PROPERTIES

amount: integer range: 1 ≤ x ≤ 9999999999 required
Refund Amount

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

currency: string enum: [GBP, EUR, USD] range: (up to 3 chars) required
Refund Currency

Example

```
{
  "amount": 1000,
  "currency": "GBP"
}
```

RefundOutput: object

PROPERTIES

system: System required

response: object optional
Return if it is a HTTP 200 response

PROPERTIES

refund: Refund required

links: Array< HAL > required
List of all related resources.

> **NOTE:**
> HAL appeared outside a response entity object usually means they are the parent entities. In this case, `order` and `payment` are the parent entities of `refund`.

Example

```
{
  "system": {
    ...See schema System for details...
  },
  "response": {
    "refund": {
      ...See schema Refund for details...
    },
    "links": [
      {
        "href": "/payment/@payment_id",
        "id": {
          "order_id": "14627849160897986"
        },
        "rel": "payment",
        "method": "GET"
      },
      {
        "href": "/orders/@order_id",
        "id": {
          "order_id": "ORDER-1234QWER"
        },
        "rel": "order",
        "method": "GET"
      }
    ]
  }
}
```

Refund: object

PROPERTIES

id: string range: (up to 50 chars) required
Unique Entity ID of a Refund, identical to `pasref`

pasref: string range: (up to 50 chars) required
Refund Reference. A unique reference generated by Payment Gateway

created_at: string range: (up to 20 chars) required
Created Time of the refund

last_modified: string range: (up to 20 chars) required
Last Modified Time of the refund

amount: integer range: 1 ≤ x ≤ 9999999999 required
Refund Amount

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

currency: string enum: [GBP, EUR, USD] range: (up to 3 chars) required
Refund Currency

status: string enum: [batched, pending] range: (up to 50 chars) required

Example

```
{
  "id": "16219383951512048",
  "pasref": "16219383951512048",
  "created_at": "2021-06-11T14:10:25Z",
  "last_modified": "2021-06-12T14:10:25Z",
  "amount": 1000,
  "currency": "GBP",
  "status": "pending",
  "metadata": null
}
```

The payment/settlment status of the corresponding Payment, only available for Card Payment

Possible Value	Definition
batched	The transaction has been batched and sent for settlement, can no longer be voided. Applies to both refunds and sales.
pending	The transaction has been successfully processed and is awaiting batching (midnight cutoff). Can still be voided prior to batching. Applies to both refunds and sales.

metadata: Metadata range: (up to 20 objects) optional
Key-value pair that can be used to store additional information about the entity.

Item: object

PROPERTIES

product_name: string range: (up to 200 chars) required
Product Item Name / Description

product_id: string range: (up to 50 chars) required
Product Number / ID

unitAmt: integer range: 100 ≤ x ≤ 99999999999 required
Unit Amount of each item

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

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

vat: integer range: 0 ≤ x ≤ 999999999 required
Total VAT Tax Amount for all units

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

subAmt: integer range: 100 ≤ x ≤ 999999999 required
The Sum of one particular item with mulitple orders plus VAT.

> ! NOTE: For example, `unitAmt x unit + vat = subAmt` Do not use comma or dot. For example: Input `10000` instead of `100.00`

Example

```
{
  "product_name": "Product Item 1",
  "product_id": "prod-9ijn8uhb",
  "unitAmt": 900,
  "unit": 1,
  "vat": 100,
  "subAmt": 1000
}
```

Card: object

PROPERTIES

brand: string enum: [VISA, MASTERCARD, AMEX, DINERS, DISCOVER, JCB, CUP] range: (up to 50 chars) required
Indicates the card brand that issued the card

authcode: string range: (up to 50 chars) required
The authorization code generated when the card is successfully authorized

mcn: string range: (up to 16 chars) required
Masked card number

cvv_result: string enum: [MATCHED, NOT_MATCHED, NOT_CHECKED] range: (up to 50 chars) required
The result of the CVV check

dcc: object required
Dynamic currency conversion (DCC). Return `null` if the card payment is not a DCC payment.

PROPERTIES

amount: integer range: 1 ≤ x ≤ 9999999999 required
Card Holder Amount

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

currency: string enum: [GBP, EUR, USD] range: (up to 3 chars) required
Card Holder Currency

ccp: string enum: [FEXCO, EUROCONEX] range: (up to 50 chars) required
The name of the Currency Conversion Processor (CCP) the request is to be sent to

fx_rate: number *(double)* required
Exchange rate. In this example, 1 GBP = 1.3244 USD

margin_percentage: number *(double)* required
Margin Percentage. The foreign exchange markup in relation to the ECB Daily Rate

Example

```
{
  "brand": "VISA",
  "authcode": "12345",
  "mcn": "401200******1112",
  "cvv_result": "MATCHED",
  "dcc": {
    "amount": 1324,
    "currency": "USD",
    "ccp": "FEXCO",
    "fx_rate": 1.3244,
    "margin_percentage": 3.75
  }
}
```

HAL: object

DESCRIPTION

Hypertext Application Language (HAL) is an Open API standard convention for defining hypermedia such as links to related resources within JSON or XML code

PROPERTIES

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

id: object required

PROPERTIES

entity_id: string range: (up to 100 chars)
Entity ID used in the URL

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

method: string enum: [GET, POST, PATCH, DELETE] range: (up to 100 chars) required
HTTP Method

Example

```
{
  "href": "/entity/@entity_id",
  "id": {
    "entity_id": "entity-111222333"
  },
  "rel": "EntityName",
  "method": "GET"
}
```

Exception: object

PROPERTIES

system: System required

response: object optional
Return if the exception is taken place in any downstream system

PROPERTIES

request_result: object required

PROPERTIES

api_gateway: object required
Result returned by API Gateway

PROPERTIES

code: string range: (up to 50 chars) required
Result code

Example

```
{
  "system": {
    "messageId": "89817674-da00-4883",
    "returnCode": "400",
    "returnReason": "<Corresponding Error Message>",
    "sentTime": "2016-11-15T10:00:00.000Z",
    "responseTime": "2016-11-15T10:00:00.000Z"
  },
  "response": {
    "request_result": {
      "api_gateway": {
        "code": "999999",
        "message": "System Error"
      },
      "payment_gateway": {
        "code": "101",
        "message": "Declined by Bank"
      }
    }
```

message: string range: (up to 100 chars) required
Result message

payment_gateway: object required
Result returned by Payment Gateway

PROPERTIES

code: string range: (up to 50 chars) required
Result code

message: string range: (up to 100 chars) required
Result message

System: object

PROPERTIES

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

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

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

Possible Value	Definition
200	Successful operation
400	Bad Request (With detail message in field `returnReason`)
500	Internal Error. Notices: Faster Payment System is a multiple-tiers system, system down or unavailable of any single dependent system (or tier) across the entire FPS pipeline can return HTTP Return Code `500` with different `returnReason`. Furthermore, if one tier which comes before the API Cloud Foundry is unavailable, such as the API Gateway, there will be even no json message returned. Developer is suggested to catch the native HTTP Return Code before trying to look into the `returnCode` in json message.

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	Profile ID - Merchant ID mapping is not correct/updated!	The binding of Profile 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 corresponding dependent system which returns this message. Yet, all reasons can be concluded into System Unavailable.

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

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

Callback: object

PROPERTIES

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

Metadata: object

PROPERTIES

A JSON object delimited by `,` both `key` and `value` support free-typed string up to 512 chars.

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™**.

```
  }
}
```

Example

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

Example

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

Example

```
"metadata": {
  "key_1": "value_1",
  "key_2": "value_2",
  "key_3": "value_3",
  ...More pairs can be defined below...
}
```

Key Generation and Certificate Exchange with HSBC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 3 entries

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

<Other Information>

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

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

<Other Information>

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

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

<Other Information>

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

Certificates and Keys Maintenance

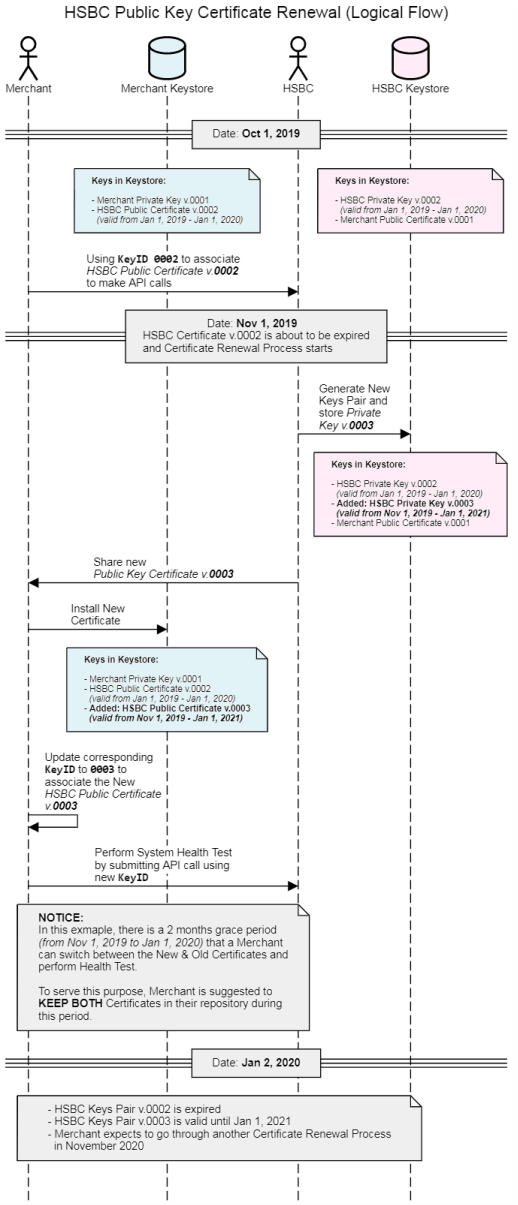
Here are some recommendations to Merchant of how to properly maintain certificates and keys:

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

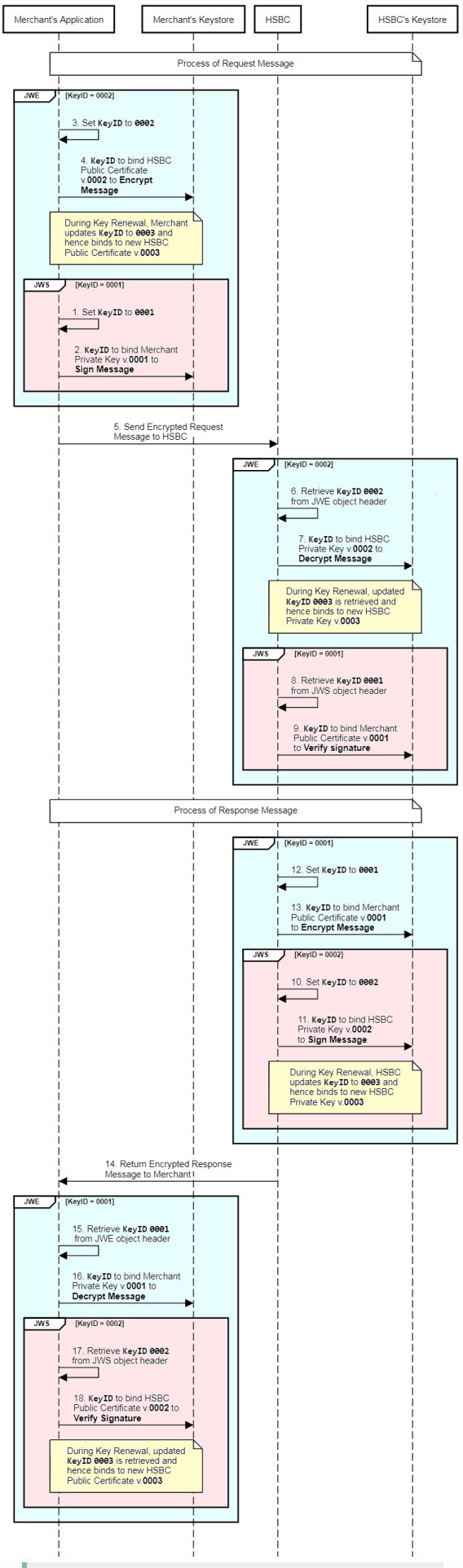
Certificates and Keys Renewal

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

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



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



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

Disclaimer