DBS HOSTED PAYMENTS PLATFORM

API SPECIFICATION GUIDE

Version 2.2

Neither this document nor any part of it may be copied or reproduced in any form or by any means, or translated into another language, without the prior consent of DBS.

DBS makes no warranties or representations with respect to the content hereof and specifically disclaim any implied warranties of merchantability or fitness for any particular purpose. DBS assumes no responsibility for any errors that may appear in this document. The information contained in this document is subject to change without notice. DBS reserves the right to make any such changes without obligation to notify any person of such revision or changes. DBS makes no commitment to keep the information contained herein up to date.

Document Sign-Off

a. Revision History

Revision Date	Version	Summary of Changes
09/09/2020	1.0	Initial Draft included in versioning.
11/05/2021	1.0.10	Support the PayLah! Journey Web and app checkout for HPP users, Updates to the error codes in line with Merchant requirements.
25/08/2022	1.0.15	PayLah! Hybrid Journey Implementation for HPP users.
27/05/2023	2.0	• PayLah!, Creditcard and Paynow payments • Refund and status enquiry features
10/01/2023	2.2	Changes to the API specs along with the comprehensive documentation.

b. API Version

Version	Summary of Changes
1.0	Supports PayLah! express setup and payment
1.1	Supports PayLah! express setup only, express setup and payment, API based payment.
2.1	Version highlights: • Supports express setup only, onetime payment, express setup and, API based payment. • Supports credit cards (Visa, Mastercard) tokenization, onetime payment, express setup and payment. • Supports Paynow payments using DBS/POSB login, QR scan & Pay • Supports Status enquiry. • Supports online refund for all HPP Payment channels.

TABLE OF CONTENTS

1 DBS HOSTED PAYMENTS PLATFORM 2.0

1.1 INTRODUCTION

The DBS Hosted Payment Platform (HPP) is a payment solution tailored for online and digital merchants. It enables these merchants to integrate the platform into their internet-powered applications, allowing customers to make seamless transactions using PayLah!, credit cards, and PayNow.

The HPP Connect API is the primary integration point for merchants to connect with DBS. By integrating with this API, merchants can access a variety of payment methods offered by the Hosted Payment Platform, providing a straightforward and ready-to-use solution.

DBS Hosted Payment Platform (HPP) presents a convenient avenue for merchants to accept payments via PayLah! (Mobile wallet), credit cards, and PayNow. Through the DBS hosted payments platform, merchants can effortlessly incorporate these payment methods into their applications, enhancing the payment experience for their customers.

2 APIs MESSAGE HPP SPECIFICATION

2.1 OVERVIEW

This API message specification guide aims to detail the integration process for DBS's Hosted Payment Platform (HPP), which facilitates a smooth payment experience for buyers. Merchants are equipped to incorporate the HPP Connect API into various platforms, including websites, mobile web, and mobile apps, enabling them to accept payments via PayLah! (Mobile wallet), credit cards, and PayNow.

The document encompasses detailed API specifications, outlines mandatory parameters essential for processing payments, and describes optional fields that enhance customer interaction. Additionally, it includes information on status codes and error messages that require handling, along with code snippets to streamline the implementation process.

2.2 MESSAGE TRANSFER & STRUCTURE

All requests and responses for the API are formatted in JSON and encoded in UTF-8. It's essential to note that access to the API is restricted to secure HTTPS connections only.

Attempts to access the API via an unsecured HTTP connection will result in an error. Consequently, it's critical to ensure that all communications with the API are conducted over HTTPS to uphold the security and integrity of the data exchanged between the client and the API server.

2.3 MIME TYPES

To facilitate proper communication with the API, it's necessary to include an 'Accept' header in all requests. This header indicates the media type that the client can handle in the response.

For example, when the header is set to 'Accept: text/plain', it signals that the client expects a response in plain text format. The API server, in turn, provides the response in the specified format.

Similarly, when data is sent to the API using a POST request, it's crucial to include a 'Content-Type' header. This header defines the media type of the data in the request body.

An example would be 'Content-Type: text/plain', which ensures that the API server interprets and processes the request body as plain text. The inclusion of this header is essential for the API server to handle the content correctly.

2.4 CHARACTER SET

The message can contain a set of characters that are considered valid within the specified fields. These include:

Lowercase letters: a b c d e f g h i j k l m n o p q r s t u v w x y z

Uppercase letters: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Digits: 0 1 2 3 4 5 6 7 8 9

Special characters: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ ] ^ _ ` { | } ~

These characters can be considered valid and used within the specified fields. Additionally, spaces can also be valid characters within the fields, except as the first or last character within any field.

2.5 FIELD TYPES

Type	Description
Numeric (N)	0-9; two decimal point scaling
Alphabetic (A)	a-z, A-Z
Alphanumeric (AN)	a-z, A-Z, 0-9
Alphanumeric Dash (AND)	a-z, A-Z, 0-9, dash/hyphen character ("-")
DateTime	YYYY-MM-DDTHH:MM:SS.sss YYYY-MM-DDTHH:MM:SS YYYY-MM-DDTHH:MM:SS.sss+ZZZZ
SWIFT character set (S)	a-z, A-Z, 0-9 / - ? : ( ) . , ‘ + space character
String character set (G)	a-z, A-Z. 0-9 ! “ # $ % & ‘ ( ) * + , - . / : ; < ‘= > ? @ [ ] ^ _ ` { | } ~ space character
Email character set (EMAIL)	a-z, A-Z, 0-9 ! “ # $ % ‘ ( ) * + , - . / : ; < ‘= > ? @ [ ] ^ _ ` { } ~
Boolean	true, false

Note: \, <, > and " are not supported.

2.6 FIELD REQUIREMENT

Requirement	M/O/C	Description
Mandatory	M	Validation Rule: 1. The field value must exist in the payload. 2. It should not be 'null'.
Optional	O	Validation Rule: 1. The field value may or may not be present in the payload. 2. It allows null values or empty strings (" ").
Conditional	C	Validation Rule: 1. The field value can be mandatory or optional based on the value of other fields in the payload. 2. If it is mandatory, it follows the mandatory validation rule. 3. If it is optional, it follows the optional validation rule.

2.7 MERCHANT ONBOARDING

The following are the client onboarding parameters that need to be established one time with DBS:

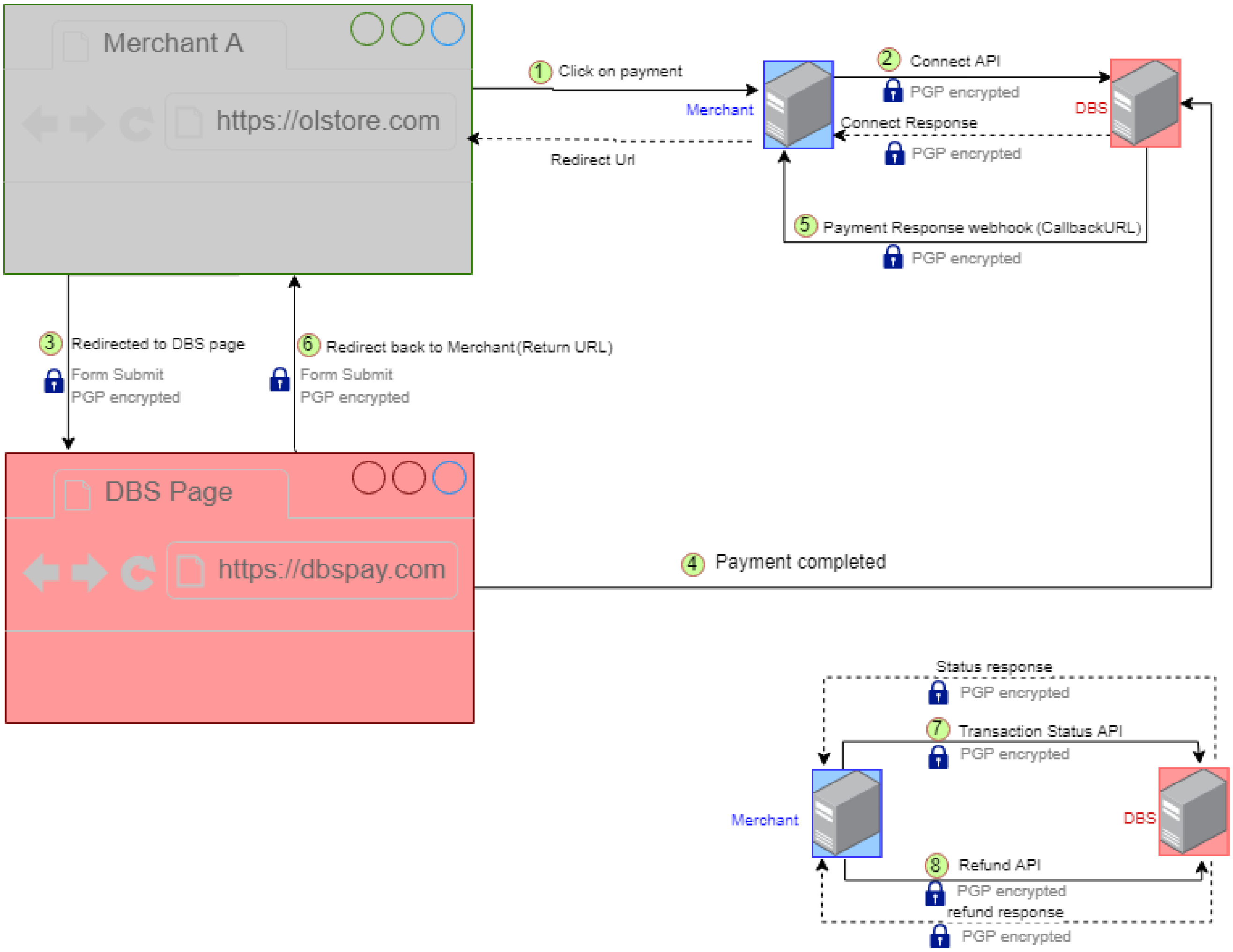
S/N	Type	Description
1.	ORG_ID	This is a unique Organization ID assigned to the merchant and provided by DBS. It serves as an identifier for the organization.
2.	PGP keys	The merchant's PGP (Pretty Good Privacy) public key and the Bank's PGP public key need to be exchanged. These keys should be RSA 2048-bit ASCII armored format.
3.	Merchant Logo	The Merchant's logo that needs to be displayed on HPP Pages must be presented. Logo Specifications: Dimension: nXn Size < 2 MB Format: PNG or JPEG
4.	IP Address(UAT)	The static IPs from which UAT will be conducted by client. These needs to be whitelisted
5	Url domain	Url domains for return, cancel and callback URLs must be shared. these are expected to whitelist for UAT and production.

As part of the account onboarding process, the DBS Merchant Onboarding team will provide the data mentioned above.

3 PROCESS FLOW

3.1 PROCESS FLOW DIAGRAM: MERCHANT ACCESSING HPP APIS

The diagram below illustrates the process flow for HPP payment integration, transaction status, and the refund API of HPP.



4 HPP CONNECT API

To initiate a payment request to DBS using the HPP (Hosted Payment Page) platform, you would typically follow these steps:

Set up the payment information: Collect the necessary payment details from the customer, such as the payment amount, currency, and any additional information required by DBS or the merchant.

Determine the preferences: Define the preferences for the payment request, considering both the merchant's requirements and the customer's preferences. These preferences may include things like the payment method, preferences, appearance, and other customizations.

Construct the payment request: Create a request to the DBS HPP platform, including the payment information and preferences. The exact parameters and format of the request may depend on the API specifications provided by DBS. Common parameters may include:

orgId: Unique identifier for the merchant initiating the payment request.
merchantReference: Unique identifier for the merchant transaction throughout the payment lifecycle.
typeOfPayment: The Payment experience that the merchant preferred to initiate with DBS for customers.
amount: The amount to be paid by the customer.
currency: The currency in which the payment should be processed.
preferredPaymentMethods: The preferred payment method, such as credit card, payLah! or paynow.
channelType: The channel through which customer initiate the payment.
returnUrl: The URL to which the customer should be redirected after completing the payment. This Return Url must be whitelisted with DBS and included in the payload.
cancelUrl: The URL to which the customer should be redirected after canceling the payment is required. This Cancel URL must be whitelisted with DBS and included in the payload.
callbackUrl: The callback url is required to be whitelisted with DBS and to be shared in payload to receive the webhook responses.
version: The version is mandatory to route to handle the HPP capabilities. The default value must be “2.1”.
Additionally, you may need to include other parameters specific to the HPP platform or the merchant's integration requirements.

Encrypt the payment request: DBS recommends all the API payloads to be sent using PGP encryption technique.
To encrypt the payment request using PGP encryption, you'll need to follow these steps:

Obtain the DBS public key: Contact DBS IM to obtain our public key or a method to securely obtain it. The public key is required to encrypt the payment request.

Install PGP software: Install a PGP software or library that provides encryption capabilities. There are various PGP implementations available, such as GnuPG (GPG), PGP Command Line, or PGP libraries for your programming language.

Generate your own PGP key pair: Generate your own PGP key pair consisting of a private key (kept secret) and a public key (shared with DBS). You can use the PGP software to generate your key pair.

Import the DBS public key: Import DBS's public key into your PGP software. This allows your software to use our public key for encryption.

Encrypt the payment request: Use your PGP software to encrypt the payment request payload. The exact method and command may vary depending on the software you are using. Typically, you would use the DBS public key to encrypt the request and sign using your own private key.

Prepare the encrypted request: Format the encrypted payment request according to the specifications provided by DBS. This could involve embedding the encrypted data into a specific field or message format required by the API.

Send the encrypted payment request: Send the encrypted payment request to DBS using the designated API endpoint and typically using a POST request. The endpoint URL and authentication method may vary depending on the DBS API specifications. Ensure that the request is transmitted securely to protect the encrypted data from unauthorized access.

It's important to follow the specific instructions and recommendations provided by DBS regarding the encryption process, encryption algorithms, key management, and secure transmission of the encrypted request. Consult our documentation or contact DBS gateway support for any specific guidelines or requirements for encrypting the payment request using PGP encryption.

DBS will then use our private key to decrypt the encrypted payment request and process it accordingly.

Handle the payment response: Once the payment request is processed by the DBS HPP platform, you will receive a transaction response in PGP encrypted format.

You can further decrypt the encrypted response using the Key Pair (Merchant Private key) and un sign using the DBS public key. The response typically includes information about the status of the payment, transaction ID, and any additional details provided by DBS.

Process the payment response: Depending on the outcome of the payment response, you can take appropriate actions in your system. For example, if the payment is successful, you can update your order status and proceed with order fulfillment. If the payment fails or is declined, you can handle the error and notify the customer accordingly.

It's important to consult the documentation provided by DBS for our specific API and HPP platform to understand the exact details of constructing the payment request, handling the response, and any additional requirements or considerations for integrating with our system.

4.1 URL

Method	Banking Service URL
POST	api/sg/hpp/v4/payment/transaction

4.2 CONNECT API REQUEST MESSAGE STRUCTURE

The table displays the structural formatting and field validations in each building block.

4.2.1 MESSAGE HEADER

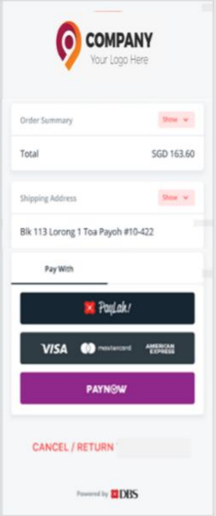
The message begins with a header that includes the following information:
x-api-key: <keyId>
X-DBS-ORG_ID: <orgId>
Content-Type: text/plain

In the above information:
<keyId> represents the value of the key to be exchanged with DBS. This key will be provided by DBS.
<orgId> refers to the Company ID assigned to the Merchant by DBS. All alphabetical characters should be in capital letters.

4.2.2 MESSAGE BODY

The transaction details are mandatory and should be encrypted using PGP (Pretty Good Privacy) with the bank's public key.

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN (20)	M	A unique reference number that is generated by the merchant for each service call. The expected format is : [YYYYMMDD][sequence number]. Example: 20230425scbdf32456"
1.0.2	orgId	AN (12)	M	Unique Organization Identifier for the merchant, provided by DBS. All alphabetical characters should be in capital letters. Example: TESTXXXXX01
1.0.3	timeStamp	DateTime	M	Date and time of the request message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains the array of request objects.**

2.0.1	txnInfo	Class	M	This block contains Connect API request details
2.0.1.1	typeOfPayment e.g., Below  COMPANY Your Logo Here Order Summary Show Total SGD 163.60 Shipping Address Show Blk 113 Lorong 1 Toa Payoh #10-422 Pay With PayLah! VISA mastercard AMERICAN EXPRESS PAYNOW CANCEL / RETURN Powered by DBS	S(25)	M	The merchant allows the following payment methods for buyers on HPP. The value must be a member of the following list, and the values are case sensitive: **w01**: Indicates onetime payment. Applicable for both registered and guest users of the ecommerce store, for PayLah!, Credit Card, and PayNow payment methods. Recommended for merchants with unregistered/guest users. **w02**: Indicates optional tokenization (express checkout) setup and payment. Buyers can choose between onetime payment or tokenization setup and payment. Applicable for PayLah!, Credit Card, and PayNow payment methods. Note: Express setup doesn't support PayNow, but users can still make PayNow payments. Recommended for merchants with registered users, allowing them to save their payment details for future hpp payment transactions. **w03**: Indicates mandatory tokenization (express checkout) and payment. Onetime payment is not allowed for customers. Buyers are enforced by the merchant to proceed with tokenization setup and payment. Applicable for PayLah! and Credit Card only. Tokenization setup and payment doesn't support PayNow. **a01**: Indicates API-based express payment for PayLah! only. Buyers are enforced by the merchant to proceed. Applicable for PayLah! only. **Note:** API-based payments don't support Credit Card and PayNow. Example: w01
2.0.1.2	**merchantInfo**	**Class**	**M**	**This class consists of merchant details**
2.0.1.2.1	brandName	S (100)	O	The brand name of the merchant can be sent for the buyer to view on the HPP (Hosted Payment Page) page. This feature facilitates merchants who prefer to display their brand name instead of their company name on the HPP page. Example: FOODO
2.0.1.2.2	geoLanguage	S (10)	O	The Geo language code represents the locale for web-based applications. The default value is en_US. Example: en_US, en_SG

<table>
<tr><td>2.0.1.2.3</td><td>returnUrl</td><td>S (256)</td><td>C</td><td>The returnUrl is the redirect URL where DBS redirects the transaction status after transaction processing from the front end.
<table><tr><th>Web</th><th>App</th></tr><tr><td>Mandatory</td><td>Not required</td></tr></table>
Note: Not applicable for API based express payments (typeOfPayment = ‘a01’)

Example:
https://xxx-xx.com.sg/merchant/return</td></tr>
<tr><td>2.0.1.2.4</td><td>cancelUrl</td><td>S (256)</td><td>C</td><td>The cancelUrl indicates the redirect URL to pass the termination status of the transaction from the front end.
<table><tr><th>Web</th><th>App</th></tr><tr><td>Mandatory</td><td>Not required</td></tr></table>
Note: Not applicable for API based express payments (typeOfPayment = ‘a01’)

Example: https://xxx-xx.com.sg/merchant/cancel</td></tr>
<tr><td>2.0.1.2.5</td><td>callbackUrl</td><td>S (256)</td><td>C</td><td>The callbackUrl indicates the webhook URL used to call back to the merchant with transaction updates.

The callbackUrl is mandatory. DBS recommends that merchants to rely on the callback response(asynchronous) as the primary source of transaction updates, rather than relying solely on the front-end response(synchronous).<table><tr><th>Web</th><th>App</th></tr><tr><td>Mandatory</td><td>Mandatory</td></tr></table>
Note: Not applicable for API based express payments (typeOfPayment = ‘a01’)

Example: https://xxx-xx.com.sg/merchant/callback</td></tr>
<tr><td>2.0.1.2.6</td><td>appDeeplinkUrl</td><td>S (256)</td><td>C</td><td>The appDeeplinkUrl is the scheme URL to deeplink from PayLah!.
The appDeeplinkUrl is required when the channel type is HC01 or HC02 or HC01H or HC02H.
It serves as a deeplink URL to launch the merchant's mobile app when the payment is completed or cancelled.</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td>Note: Merchant must send the appropriate deeplink url i.e for iOS App – iOS specific appDeeplinkUrl and for android app android specific appDeeplinkUrl
<table><tr><th>Web(PayLah!)</th><th>App(PayLah!)</th></tr><tr><td>Not required</td><td>Mandatory</td></tr></table>
App deeplink url is applicable for PayLah!.
For Creditcard and Paynow app deeplink is not required.
Note: Not applicable for API based express payments (typeOfPayment = ‘a01’)
i<i>.e, refer to type of payment parameter for more details.</i></td></tr>
<tr><td><b>2.0.1.3</b></td><td><b>customerInfo</b></td><td><b>Class</b></td><td><b>M</b></td><td><b>This class consists of customer information</b></td></tr>
<tr><td>2.0.1.3.1</td><td>customerId</td><td>S (256)</td><td>O</td><td>The customerId is the Merchant's user identifier of registered customer of the Merchant Store.
Example: MERCUST456702</td></tr>
<tr><td>2.0.1.3.2</td><td>hppExpressId</td><td>S (256)</td><td>C</td><td>The hppExpressId is the customer identifier of a registered hpp customer for express payment. The hppExpressId should be created by HPP and shared with the merchant upon successful express setup/tokenization.
This field is required for returning customers (buyers) who wish to make express payments. However, the HPP express Id will be optional for guest user payments.
Note: HPP expressId is optional for paynow based merchants.
Example: HEP345CGHF3124</td></tr>
<tr><td><b>2.0.1.4</b></td><td><b>transactionInfo</b></td><td><b>Class</b></td><td><b>M</b></td><td><b>This class consists of transaction information</b></td></tr>
<tr><td>2.0.1.4.1</td><td>merchantReference</td><td>AN(20)</td><td>M</td><td>The customerId is the Merchant's user identifier of registered customer of the Merchant Store.
Example: MERCUST456702</td></tr>
<tr><td><b>2.0.1.4.2</b></td><td><b>amount</b></td><td><b>Class</b></td><td><b>M</b></td><td><b>Transaction amount details</b></td></tr>
<tr><td>2.0.1.4.2.1</td><td>amount</td><td>N(8) or N(6).N(2)</td><td>M</td><td>The amount indicates the total amount for the order, including the net amount and any additional taxes and surcharges.
The data for the amount is a string that consists of the characters 0-9 and '.' and represents a valid decimal number with two decimal digits.</td></tr>
</table>

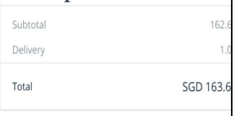
				Important: The numbering format should not include commas or special characters. Example: 100.55
2.0.1.4.2.2	currency	AN (3)	M	The currency code of the order is expressed as an ISO 4217 alpha code, for example, SGD. The currency code should be "SGD" by default. Please note that the system currently accepts SGD only. Example: SGD

2.0.1.4.3	channelType	S (4)	M	The channelType indicates the channel through which the payment is made.
				Possible values are:
				HC01 = iOS Mobile Application
				HC02 = Android Mobile Application
				HC01H = iOS Mobile Application-Hybrid Journey (Applicable for Paylah)
				HC02H = Android Mobile Application-Hybrid Journey (Applicable for paylah)
				HC03 = Web Channel
				HC04 = Mobile Web Channel
				HC09 = Others
				Example: HC03

2.0.1.4.4	txnType	S (4)	O	The transactionType indicates the type of transaction through which the payment is initiated.
				Possible values are:
				TX01 = Online
				TX02 = Offline
				TX03 = Instore
				TX04 = Point of Sale
				TX05 = Subscription
				TX06 = Invoices
				TX07 = offline2Online
				TX08 = B2B
				TX09 = Cash on Demand
				TX10 = QR Pay
				Example: TX01

2.0.1.4.5	preferredPaymentMethods	Array[10]	C	The preferredPaymentMethods parameter is a comma-separated list of the allowed payment methods for the buyer. All other methods will be ignored. It's important to note that this parameter will override the account settings.

				However, the preferredPaymentMethods must be a subset of the merchant payment entitlements. **Note:** For merchant who opt for Payment as a separate rail, we expect this parameter to send in the payload, i.e for PayLah! “preferredPaymentMethods” : “[PT01]” PT01 – Indicates PayLah! payment PT02 – Indicates Credit Card PT03 – Indiactes PayNow null - no preference in payment method Example: preferredPaymentMethods: [PT01, PT03]
2.0.1.5	**paymentInfo**	**Class**	**O**	**This block consists of payment details. PaymentInfo is not required if tokenization/express setup scenarios.**

2.0.1.5.1	paymentItemSubtotal
	Example:
	
	Subtotal 162.6
	Delivery 1.0
	Total SGD 163.6
		N(8)
		N(6).N(2)
			O	The subtotal represents the payment amount excluding tax and surcharges.
				Data is a string that consists of the characters 0-9 and '.' with a valid decimal number with two decimal digits.
				No commas included in the numbering format.
				Example: 75.75

2.0.1.5.2	paymentShippingCharge	N(8)
		N(6).N(2)
			O	The shipping charge refers to the cost associated with shipping the purchased items.
				Data is a string that consists of the characters 0-9 and '.' with a valid decimal number with two decimal digits.
				No commas included in the numbering format.
				Example: 20.25

2.0.1.5.3	paymentTax	N(8)
		N(6).N(2)
			O	The tax component of the payment transaction amount, such as GST or VAT, should be provided. The data should be formatted with two decimal digits.
				Data is a string that consists of the characters 0-9 and '.' with a valid decimal number with two decimal digits.
				No commas included in the numbering format.
				Example: 4.55

2.0.1.5.4	**loyaltyDiscount**	**Class**	**O**	**Indicate the loyalty discount information provided by the merchant**
2.0.1.5.4.1	loyaltyDiscountDescr	S(100)	O	The loyalty discount name details for the transaction can be indicated. Example: NEWLAUNCH

2.0.1.5.4.2	**campaign**	**Array(3)**	O	**The campaign name for the included loyalty discount can be indicated.** It would be allowed to display a max of 3 items for the campaign. JSON type Array

2.0.1.5.4.2.1	loyaltyDiscountAmt	N(8)
		N(6).N(2)
			O	The discount amount provided for the campaign should be indicated.
				The data should be a string consisting of the characters 0-9 and '.' and represent a valid decimal number with two decimal digits. The numbering format should not include commas.
				
				Example: 5.50

2.0.1.5.4.2.2	campaignName	S(100)	O	The campaign name provided for the display of the loyalty discount can be indicated. Example: NEWLAUNCHDISCOUNT
2.0.1.5.5	itemCount	N(3)	O	The itemCount represents the count of the number of items in the transaction. It should be a whole number ranging from 0 to 999. The itemCount should not exceed 999. Example: 10
2.0.1.5.6	paymentSource	S (256)	O	The paymentSource Name indicates the name of the payment source for marketplace payments. It can be used when there is a marketplace with multiple merchants, and the payment source specific to a merchant needs to be shared. Example: NIKOUTLETSOUTH36
2.0.1.5.7	partnerId	S (100)	O	The partner ID for a Marketplace Merchant can be indicated. This field is optional and can be used to identify the specific merchant associated with the transaction. Example: VIVNIKE01
2.0.1.6	**itemInfo** ***This is an array List*. Max 10 items can be sent.**	**Array, Class**	**O**	**Item list that the buyer purchased in this transaction.** **Delivers better conversion. Your buyer can see the transaction items/amt when he or she completes checkout on DBS Platform**
2.0.1.6.1	itemName	S (256)	O	Name of the item purchased. Example: Baby Toy Car.
2.0.1.6.2	itemDescription	S (256)	O	Short description of the Item. Example: Joystick operated toy car
2.0.1.6.3	itemQuantity	N (3)	O	The item quantity should be between 1 and 999. Example: 2

2.0.1.6.4	itemCost	N (8)	O	The cost of each item should not exceed the total amount. The data should be in two decimal digits. Data is a string that consists of the characters 0-9 and '.' **IMP -** No commas or special characters should be included in the numbering format. For example: 55.99
2.0.1.7	**deliveryInfo**	**Class**	**O**	**This block consists of the delivery information for the products purchased. For tokenization only (typeOfPayment=w03 and amount=0) deliveryInfo is not required. For others, this field is optional.**
2.0.1.7.1	addressLine1	S (256)	O	The address line 1 should contain the house number or block number of the address. Example: #03-295 XXXXXX COMPLEX
2.0.1.7.2	addressLine2	S (256)	O	The address line 2 can be used to provide additional details such as the street name of the address. This field is optional. Example: 257 XXXXXX Road
2.0.1.7.3	city	S (256)	O	The city name can be provided as part of the address information. This field is optional. Example: Singapore
2.0.1.7.4	zipCode	S(6)	O	The post code or zip code can be provided as part of the address information. This field is optional. Example: 188350
2.0.1.7.5	country	S (100)	O	The country name can be provided as part of the address information. This field is optional. Example: Singapore
2.0.1.7.6	shopperEmail	S (256)	O	The email address can be provided for digital delivery purposes. This field is optional. Example: customer.XXXXX@gmail.com
2.0.1.7.7	deliveryMode	S (100)	O	The delivery mode indicates the mode of delivery for the order. The possible values for the delivery mode are 'Physical' and 'Digital'. This field is optional. Example: Physical
2.0.1.8	**riskInfo**	**Class**	**O**	**Information for transaction risk evaluation,**
2.0.1.8.1	ipAddress	S (256)	O	The IP address of the device used by the customer for the transaction can be provided. This field is optional. Example: 192.xxx.xx.90

2.0.1.8.2	deviceId	S (256)	O	The device id that the customer used for the transaction. Example: I-213t1236xxxe34
2.0.1.8.3	operatingSystem	S (100)	O	Operating system used in the device, Example: Mac
2.0.1.8.4	browserType	S (100)	O	Browser information for the transaction, Values: IE, Firefox, Edge, Safari, Chrome, Others. Example: Safari
2.0.1.9	version	S(5)	M	Indicates API version. Example: 2.1

4.3 CONNECT API RESPONSE STRUCTURE : ONETIME - w01/ ECSETUP&PAY- w02/w03

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**Header**	**Class**	**M**	**This block contains the response header details.**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number. Example: 20230425scbdf32456
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains the array of response objects.**
2.0.1	**txnResponse**	**Class**	**M**	**This block contains the transaction response**
2.0.1.1	txnStatus	S (10)	M	The transaction status is always provided and represented as a character string, which is case-sensitive. The possible values are as follows: ACTC: Success RJCT: Failed PDNG: Pending Please note that this information pertains to the connect API txnstatus at the gateway level. Example: ACTC
2.0.1.2	txnStatusDescription	S (256)	M	The description of the txnStatusCode is always provided and is represented as a JSON string. It should have a minimum length of 1 character and a maximum length of 256 characters. Example: Transaction created successfully.

2.0.1.3	transactionId	S (256)	M	An unique transaction Id generated by DBS(HPP) for merchant request. Example: ff477d68-f300-4dbd-a9d4-08b1a55568f9
2.0.1.4	requestId	S (256)	M	The request identifier, which should be used during redirection, is always provided. It is represented as a JSON string and must have a minimum length of 1 character and a maximum length of 256 characters. Example: 03087727-1960-4569-b1d9-349dcc6163d2
2.0.1.5	submitUrl	S (256)	M	The URL to be used during redirection is always provided. It is represented as a JSON string of URL type and should have a maximum length of 256 characters. Example: https://xxx-xx.com.sg/hpp/payment/pages
2.0.1.6	version	S (5)	M	API version used for this request. Version will not be returned for error responses. Example: The value must be 2.1.

4.4 CONNECT API RESPONSE STRUCTURE (SEAMLESS PAYMENT(a01)))

The table shows the structural formatting and field validations in each building block.

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**Header**	**Class**	**M**	**This block contains the response header details.**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number Example: 20230425scbdf32456
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**Data**	**Array, Class**	**M**	**This block contains the array of response objects.**
2.0.1	**txnResponse**	**Class**	**M**	**This block contains transaction details**
2.0.1.1	txnStatus	S (10)	M	The transaction status is always provided and represented as a character string, which is case-sensitive. The possible values are as follows: ACTC: Success RJCT: Failed

				PDNG: Pending Please note that this information pertains to the connect API txnstatus at the gateway level. Example: ACTC
2.0.1.2	txnStatusCode	AN(4)	M	The transaction's txnStatusCode is returned and is represented as a JSON string. Example: H502
2.0.1.3	txnStatusDescription	S (256)	M	The description of the txnStatusCode is always provided and is represented as a JSON string. It should have a minimum length of 1 character and a maximum length of 256 characters. Example: Transaction created successfully.
2.0.1.4	transactionId	S (256)	M	An unique transaction reference generated by DBS for merchant request. Example: ff477d68-f300-4dbd-a9d4-08b1a55568f9
2.0.1.5	merchantReference	AN(20)	M	The merchantReference indicates the unique reference number shared by the merchant. It is a significant parameter used to inquire about the transaction status in case DBS fails to communicate the definite status of transaction. Example: PDBA61CKJWARYBBE65C8
2.0.1.6	**customer**	**Class**	**M**	**This block returns the customer specific Information in response.**
2.0.1.6.1	hppExpressId	String	M	The response includes the hppExpressId of the transaction echoed back from the API request. Example: 32ruewgfer43tryuewgf
2.0.1.6.2	accountNumber	S(20)	M	Masked PayLah! phone number for the paylah transaction. PayLah! format: xxxx4597 Seamless response not applicable for creditcard and paynow. Example: xxxx4597
2.0.1.7	**references**	**Array, Class**	**C**	**Transaction reference pertaining to DBS payment sources**
2.0.1.7.1	type	S (128)	C	Indicates the transaction type. Existence: Conditional JSON type: String PT01 – Indicates PayLah! payment PT02 – Indicates Credit Card PT03 – Indicates PayNow

				Returns PT01 for seamless payments(supports PayLah! only)
2.0.1.7.2	transactionReference	S (256)	C	Reference generated by corresponding payment method. Existence : Conditional JSON type: String Applicable for PayLah! transactions only.
2.0.1.7.3	amount	N(8) or N(6).N(2)	M	The amount indicates the total amount for the transaction. The data for the amount is a string that consists of the characters 0-9 and '.' and represents a valid decimal number with two decimal digits. Important: The numbering format should not include commas or special characters. Example: 100.55
2.0.1.8	version	S (5)	M	API version used for this request. Version will not be returned for error responses. Example: The value must be 2.1.

4.5 CONNECT API ERROR STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN(20)	M	This will be the unique reference number generated by merchant for each API call.
1.0.2	timeStamp	DateTime	M	Date and time of the response message. Format: YYYY-MM-DDTHH:MM:SS.sss. E.g., 2021-03-04T15:07:26.123
2.0	**data**	**Array, Class**	**M**	**This block contains the response data**
2.0.1	**error**	**Class**	**M**	**This block contains Connect API error details**
2.0.1.1	status	S(10)	M	Status of the transaction. Existence : Mandatory JSON type - String Possible Values: ACTC - Successful RJCT – Failed PDNG – Pending
2.0.1.2	code	AN(4)	M	Response code of the transaction. Existence : Mandatory JSON type – String Min length – 4 Max length - 4

2.0.1.3	description	S(256)	M	Returns description of txnStatusCode Existence : Mandatory JSON type – String Min length – 0 Max length - 256
2.0.1.4	version	S(5)	M	Indicates API version. Example: “2.1”

4.6 GATEWAY ERROR RESPONSE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN(20)	M	This will be the unique reference number that is generated by the merchant for each API call.
1.0.2	timeStamp	DateTime	M	Date and time of the response message. Format: YYYY-MM-DDTHH:MM:SS.sss. E.g., 2021-03-04T15:07:26.123
2.0	**error**	**Class**	**M**	**This block contains Connect API error details**
2.0.1	status	S(10)	M	Status of the transaction. Existence : Mandatory JSON type - String Possible Values: ACTC - Successful RJCT – Failed PDNG – Pending
2.0.2	code	AN(4)	M	Response code of the transaction. Existence : Mandatory JSON type – String Min length – 4 Max length - 4
2.0.3	description	S(256)	M	Returns description of txnStatusCode Existence : Mandatory JSON type – String Min length – 0 Max length - 256
2.0.4	version	S(5)	M	Indicates API version. Example: “2.1”

4.7 CONNECT API REQUEST AND RESPONSE MESSAGE SAMPLES

Refer to the combinations of sample request and response for the Connect API below.

4.7.1 CONNECT API REQUEST MESSAGE SAMPLE (WEB CHECKOUT)

```
{
  "header": {
    "msgId": "be9ec28026a74a67aaaa",
    "orgId": "MERCORG01",
    "timeStamp": "2022-09-22T05:16:10.992Z"
  },
  "data": [
    {
      "txnInfo": {
        "typeOfPayment": "w02",
        "version": "2.1",
        "merchantInfo": {
          "brandName": "FOODO",
          "geoLanguage": "en_US",
          "returnUrl": "https://xxx-xx.com.sg/ecocon/merchant/store/return",
          "callbackUrl": "https://xxx-xx.com.sg/webhook",
          "cancelUrl": "https://xxx-xx.com.sg/ecocon/merchant/store/return",
          "appDeeplinkUrl": null
        },
        "customerInfo": {
          "customerId": "custidxxxxxxx",
          "hppExpressId": "1808e2da-b0f8-4258-897b-9c8ecd9e773a"
        },
        "paymentInfo": {
          "paymentItemSubtotal": 5.25,
          "paymentShippingCharge": 0.80,
          "paymentTax": 0.20,
          "itemCount": 1,
          "loyaltyDiscount": {
            "loyaltyDiscountDescr": "Special Discount",
            "campaign": [
              {
                "campaignName": "New year discount",
                "loyaltyDiscountAmt": 0.50
              }
            ]
          }
        },
        "transactionInfo": {
          "merchantReference": "perftest9124400608",
          "txnType": "TX01",
          "amount": {
            "amount": 5.75,
            "currency": "SGD"
          },
          "channelType": "HC03",
          "preferredPaymentMethods": [
            "PT01",
            "PT02",
            "PT03"
          ]
        },
        "itemInfo": [
          {
            "itemName": "HealthMania",
            "itemDescription": "great product for health alerts",
            "itemQuantity": 1,
            "itemCost": 5.25
          }
        ],
        "deliveryInfo": {
          "addressLine1": "1xx Taxxxxg xxx Rd #1x-x2",
          "addressLine2": "Sanxxxxy Green Condo",
          "city": "Singapore",
          "zipCode": "4xxx22",
          "country": "Singapore",
          "deliveryMode": "PHYSICAL",
          "shopperEmail": "xxxxxxx@gmail.com"
        },
        "riskInfo": {
          "ipAddress": "10.xx.xx.32",
          "deviceId": "I-383t87656xxxe34",
          "operatingSystem": "Mac",
          "browserVersion": "Safari"
        }
      }
    }
  ]
}
```

4.7.2 CONNECT API REQUEST MESSAGE SAMPLE (APP CHECKOUT)

```
{
  "header": {
    "msgId": "be9ec28026a74a67aaaa",
    "orgId": "MERCORG01",
    "timeStamp": "2022-09-22T05:16:10.992Z"
  },
  "data": [
    {
      "txnInfo": {
        "typeOfPayment": "w02",
        "version": "2.1",
        "merchantInfo": {
          "brandName": "BRANDO",
          "geoLanguage": "en_US",
          "returnUrl": "https://xxx-xx.com.sg/ecocon/merchant/store/return",
          "callbackUrl": "https://xxx-xx.com.sg/webhook",
          "cancelUrl": "https://xxx-xx.com.sg/ecocon/merchant/store/return",
          "appDeeplinkUrl": "https://xxx-xx.com.sg/deeplink"
        },
        "customerInfo": {
          "customerId": "custidxxxxxxx",
          "hppExpressId": "1808e2da-b0f8-4258-897b-9c8ecd9e773a"
        },
        "paymentInfo": {
          "paymentItemSubtotal": 5.25,
          "paymentShippingCharge": 0.80,
          "paymentTax": 0.20,
          "itemCount": 1,
          "loyaltyDiscount": {
            "loyaltyDiscountDescr": "Special Discount",
            "campaign": [
              {
                "campaignName": "New year discount",
                "loyaltyDiscountAmt": 0.50
              }
            ]
          }
        },
        "transactionInfo": {
          "merchantReference": "perftest9124400608",
          "txnType": "TX01",
          "amount": {
            "amount": 5.75,
            "currency": "SGD"
          },
          "channelType": "HC01",
          "preferredPaymentMethods": [
            "PT01",
            "PT02",
            "PT03"
          ]
        },
        "itemInfo": [
          {
            "itemName": "HealthMania",
            "itemDescription": "great product for health alerts",
            "itemQuantity": 1,
            "itemCost": 5.25
          }
        ],
        "deliveryInfo": {
          "addressLine1": "1xx Taxxxxg xxx Rd #1x-x2",
          "addressLine2": "Sanxxxxy Green Condo",
          "city": "Singapore",
          "zipCode": "4xxx22",
          "country": "Singapore",
          "deliveryMode": "PHYSICAL",
          "shopperEmail": "xxxxxxx@gmail.com"
        },
        "riskInfo": {
          "ipAddress": "10.xx.xx.32",
          "deviceId": "I-213t1236xxxe34",
          "operatingSystem": "Mac",
          "browserVersion": "Safari"
        }
      }
    }
  ]
}
```

4.7.3 CONNECT API RESPONSE MESSAGE SAMPLE (ONETIME(w01)/ECSETUP&PAY(w02/w03) USECASES)

```
{
          "header": {
            "msgId": "8251cd2196034a668307",
            "timeStamp": "2022-09-22T05:16:11.222Z"
          },
          "data": [
            {
              "txnResponse": {
                "txnStatus": "ACTC",
                "txnStatusDescription": "Transaction created successfully",
                "transactionId": "01983c39-873f-4aa7-87cd-97122feeebb2",
                "requestId": "34a0aa4b-da31-4993-b9bc-764cbfa0448e",
                "submitUrl": "https://xxx-xx.com.sg/hpp/payment/pages",
                "version": "2.1"
              }
            }
          ]
        }
```

4.7.4 CONNECT API RESPONSE MESSAGE SAMPLE (SEAMLESS PAYMENT(a01))

```
{
 "header": {
  "msgId": "1545357575227",
  "timeStamp": "2022-09-22T05:16:11.222Z"

 },
 "data": [
  {
   "txnResponse": {
    "txnStatus": "ACTC",
    "txnStatusCode": "H502",
    "txnStatusDescription": "Payment successful",
    "merchantReference": "1545357575227",
    "transactionId": "133b5c3e-9c9e-4b6f-900d-bd767ea9b949",
    "customer": null,
    "references": [
     {
      "type": "PT01",
      "transactionReference": "7rwerg8732y32y4gsfhg8",
      "amount": "10"
     }
    ],
    "version": "2.1"
   }
  }
 ]
}
```

4.7.5 CONNECT API ERROR SAMPLE

```
{
 "header": {
  "msgId": "SG70123987456",
  "timestamp": "2017-01-26T16:16:43.567"
 },
 "data": [
  {
   "error": {
    "status": "RJCT",
    "code": "H001",
    "description": "Incorrect Organisation ID.",
    "version": "2.1"
   }
  }
 ]
}
```

4.7.6 GATEWAY ERROR RESPONSE SAMPLE

```
{
 "header": {
  "msgId": "SG70123987456",
  "timestamp": "2017-01-26T16:16:43.567"
 },
 "error": {
  "status": "RJCT",
  "code": "A001",
  "description": "Incorrect Organisation ID."
 }
}
```

5 REDIRECT TO HPP HOME PAGE(FORM POST)

To redirect to the DBS HPP homepage using the requestId, transactionId, and submitURL returned in the Connect API response and encrypt the form post request with the HPP public key and sign it with the merchant's private key using PGP encryption, you'll need to follow these steps:

- Retrieve the requestId, transactionId, and submitURL from the Connect API response.
- Construct the form post request with the necessary parameters, including the requestId and transactionId. The form should be configured to submit to the submitURL.
- Obtain the HPP public key from DBS. This key is used to encrypt the form post request.
- Generate a PGP key pair for the merchant. This key pair consists of a private key and a corresponding public key. Keep the private key secure and accessible only to the merchant.
- Use the HPP public key to encrypt the form post request. There are various libraries and tools available for PGP encryption in different programming languages. Refer to the documentation of your chosen programming language or library for specific instructions on encrypting the request.
- Sign the encrypted form post request with the merchant's private key using PGP encryption. This step ensures the authenticity and integrity of the request. Again, consult the documentation or resources specific to your chosen programming language or library for signing the request.
- Once the form post request is encrypted and signed, construct a form with the encrypted data and any other required parameters. The form should have the submitURL as the action attribute and use the POST method.
- Include any additional hidden fields or parameters required by the DBS HPP homepage in the form and programmatically submit the form

When the form is submitted, the encrypted and signed data will be sent to the DBS HPP homepage, where DBS can decrypt the request using our private key and verify the signature using the merchant's public key.
By following these steps, you can redirect to the DBS HPP homepage securely, encrypting the form post request with the HPP public key and signing it with the merchant's private key using the PGP encryption method. Make sure to consult the documentation provided by DBS for any additional requirements or specifications related to our API.

JS Interface Handling for Mobile APP flow:
It's generally recommended for the merchant (web application) to handle the JS interface and ensure proper closing of the transaction and callback handling. The implementation will depend on your specific requirements and how you want to manage the integration with the native app.

5.1 URL

Method	Banking Service URL
FORM POST	submitURL (returned from connect api response)

5.2 FORM POST REQUEST MESSAGE STRUCTURE

The table displays the structural formatting and field validations in each building block.

5.2.1 MESSAGE BODY

The transaction details are compulsory and should be in form encoded format.

S/N	Field Name	M/O/C	Description
1.0	requestId	M	requestId generated in Connect API response. Example: 34a0aa4b-da31-4993-b9bc-764cbfa0448e
2.0	encryptedPayload	M	Payload which contains the transactionId and OrgId as JSON and encrypted with HPP Public Key Example: -----BEGIN PGP MESSAGE----- Version: OpenPGP.js v4.10.4 Comment: https://openpgpjs.org wcBMA5lCfYdWCT/VAQgAidaULAH6xW8jzQVfJZ4bzxAD jTRysG8QGSvvoQyk 65i2v4Eb5DqLKit59o+ElSQ4W

5.3 JSON STRUCTURE BEFORE PAYLOAD ENCRYPTION

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	orgId	G (NA)	M	Org Id of the merchant Example: TESTxxxxxxx9
2.0	transactionId	G (35)	M	transactionId generated in Connect API response Example: 01983c39-873f-4aa7-87cd-97122feeebb2

5.4 JSON MESSAGE SAMPLE BEFORE PAYLOAD ENCRYPTION

```
{
  "orgId": "TESTxxxxxxx9",
  "transactionId": "01983c39-873f-4aa7-87cd-97122feeebb2"
}
```

5.5 FORM POST REQUEST MESSAGE IN ENCRYPTED FORMAT (UGING PGP ENCRYPTION)

```
{
"requestId": "34a0aa4b-da31-4993-b9bc-764cbfa0448e",
"encryptedPayload": "-----BEGIN PGP MESSAGE-----
Version: OpenPGP.js v4.10.4
Comment: https://openpgpjs.org
        wcBMA5lCfYdWCT/VAQgAidaULAH6xW8jzQVfJZ4bzxADjTRysG8QGSvvoQyk
        65i2v4Eb5DqLKit59o+ElSQ4Wj++y62p50JJmaczmpzKIa8vC45i1KSFSPKX
        DnTUyHed3po+J8C0bc5XZljocUAzvTqo5xVYesbMh6fJf4hdTBuHfUxh9aYT
        oHDhPcz4Rjm2QeXHihQdFGbEC3Mr/21pNkuYe3iUc5J20dZWbZMB+id+HaKB
        My+Rei4kd0vaCUXP5Y86GRTl1lblHmFp6EA1Mkf86B5YCrItBKSSrFyRneYf
                -----END PGP MESSAGE-----"
}
```

6 TRANSACTION STATUS API

The Transaction Status API provided by DBS HPP allows merchants to inquire about the latest status of merchant-initiated transactions for their customers. This API specifically enables merchants to check the payment transaction status, Express checkout setup/tokenization status and refund transaction status of HPP payments.

DBS recommends merchants to trigger the Transaction Status Enquiry API in the following scenarios:

- When DBS HPP returns a webhook notification with a status of 'PDNG' (Pending), and the merchant fails to consume the webhook notification within the expected timeframe.
- If the merchant does not receive the webhook notification at all, either due to timing issues or unknown reasons.

In these cases, the merchant can use the Transaction Status Enquiry API to retrieve the current status of the transaction directly from DBS.

To utilize the Transaction Status Enquiry API, you would typically perform the following steps:

- Retrieve the necessary transaction information required for the enquiry, such as the merchantReference or transactionId.
- Make a request to the Transaction Status Enquiry API endpoint provided by DBS. This request should include the required parameters, such as the merchantReference, authentication credentials, and any additional information needed.
- Encrypt the status enquiry call using PGP encryption with DBS public key.(Same PGP keys used for connect API call).
- Send the request to the API endpoint using the appropriate HTTP method (POST) and include any necessary headers or authentication tokens.
- Handle the API response, which will contain the current status of the transaction or refund inquiry. The response may also include additional details such as timestamps, payment amounts, or any other relevant information.

Process the response in your merchant application accordingly. You can use the received status information to update your internal records or take appropriate actions based on the transaction status.

It's important to refer to the documentation or resources provided by DBS for specific details regarding the Transaction Status API, including the API endpoint URL, request/response formats, authentication methods, and any additional guidelines or best practices recommended by DBS.

6.1 URL

Method	Banking Service URL
POST	api/sg/hpp/v4/enquiry/transactionstatus

6.2 TRANSACTION STATUS REQUEST MESSAGE STRUCTURE

The table shows the structural formatting and field validations in each building block for HPP transaction status API.

6.2.1 MESSAGE HEADER

The message starts with a header that contains the following information:

X-DBS-ORG_ID: <orgId> (The unique identity given to the merchant by DBS, where all alphabet characters should be in capital letters)
x-api-key: <keyId> (The key value exchanged with DBS, provided by DBS)
Content-Type: text/plain"

Please note that "<orgId>" represents the unique identity given to the merchant by DBS, and "<keyId>" represents the key value exchanged with DBS, which will be provided by DBS.

6.2.2 MESSAGE BODY

The transaction details are mandatory and should be encrypted using PGP with the bank's public key.

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN (20)	M	A unique reference number that is generated by the merchant for each service call. The expected format is : [YYYYMMDD][sequence number]. Example: 20230425scbdf32456"
1.0.2	orgId	AN (12)	M	Unique Organization Identifier for the merchant, provided by DBS. All alphabetical characters should be in capital letters. Example: TESTXXXXXX
1.0.3	timeStamp	DateTime	M	Date and time of the request message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains request data**
2.0.1	**txnInfo**	**Class**	**M**	**This block contains the transaction status request details.**
2.0.1.1	transactionId	S (256)	C	HPP Transaction reference ID for the merchant request. Note: Required If merchant is already having transactionId. Example: ff477d68-f300-4dbd-a9d4-08b1a55568f9Note:
2.0.1.2	merchantReference	AN(20)	M	The merchantReference indicates the unique reference number shared by the merchant. It is a significant parameter used to inquire about the transaction status in case DBS fails to communicate the definite status of transaction.

				Example: PDBA61CKJWARYBCE65C8
2.0.1.3	version	S (5)	M	Indicates api version. Example: “2.1”

6.3 TRANSACTION STATUS RESPONSE MESSAGE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains header details**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number. Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains response data**
2.0.1	**txnResponse**	**Class**	**M**	**This block contains transaction status**
2.0.1.1	txnStatus	S(10)	M	Indicates the transaction status. ‘ACTC’ = accepted ‘RJCT’ = rejected ‘PDNG’ = pending Note: For refunds status enquiry, txnStatus could be either ACTC or RJCT only.
2.0.1.2	txnStatusDescription	S (256)	M	The description of the txnStatusCode is always provided and is represented as a JSON string. It should have a minimum length of 1 character and a maximum length of 256 characters Example: Transaction created successfully.
2.0.1.3	txnStatusCode	S (10)	M	The status code represents the code returned by HPP for the transaction. It indicates the current status of the transaction. Example: H502
2.0.1.4	transactionId	S (256)	M	HPP Transaction reference ID for the merchant payment request. Example: ff477d68-f300-4dbd-a9d4-08b1a55568f9
2.0.1.5	merchantReference	AN(20)	M	The merchantReference indicates the unique reference number shared by the merchant. It is a significant parameter used to inquire about the transaction

				status in case DBS fails to communicate the definite status of transaction. Example: PDBA61CKJWARYBBE65C8
2.0.1.6	**customer**	**Class**	**C**	**Customer specific Information**
2..0.1.6.1	hppExpressId	S(256)	C	HPP generated ID for a user/customer with saved / tokenized payment details. Example: MERCUST456702
2.0.1.6.2	accountNumber	S(20)	C	Masked PayLah! phone number, masked card number for credit card transaction. Card format: 5432 15xx xxxx 4567 PayLah! format: xxxx4597 PayNow: this will not be returned to customer. Example: xxxx4597
2.0.1.7	**References**	**Array, Class**	**M**	**Transaction reference pertaining to DBS payment sources**
2.0.1.7.1	type	S (128)	M	The payment method used in this transaction represents the method chosen by the customer to make the payment. Possible values are : PT01 – Indicates PayLah! payment PT02 – Indicates Credit Card PT03 – Indicates PayNow Example: PT03
2.0.1.7.2	transactionReference	S (256)	C	The transaction reference refers to the unique identifier associated with the transaction from the appropriate payment method. It is a reference number or code that uniquely identifies the specific transaction. Note: transactionReference will not be optional for tokenization/ECSetupStatus enquiry responses. Example: 7rwerg8732y32y4gsfhg8
2.0.1.7.3	amount	N(6).N(2)	M	The amount indicates the amount that the transaction made. The data for the amount is a string that consists of the characters 0-9 and '.' and represents a valid decimal number with two decimal digits. Important: The numbering format should not include commas or special characters. Example: 100.50

2.0.1.8	version	S(5)	M	Indicates api version. Example: “2.1”

6.4 TRANSACTION STATUS ERROR MESSAGE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains header details**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains response data**
2.0.1	**error**	**Class**	**M**	**This block contains transaction status error response details**
2.0.1.1	status	S(10)	M	The status of the transaction is mandatory and must be provided. It is represented as a JSON string and can have the following possible values: ‘ACTC’ indicating successful transaction, 'RJCT' indicating a failed transaction and 'PDNG' indicating a pending transaction. Example: RJCT
2.0.1.2	code	S(4)	M	The response code of the transaction is mandatory and must be provided. It is represented as a JSON string with a minimum length of 4 characters and a maximum length of 4 characters. Example: H999
2.0.1.3	Description	S(256)	M	The description of the transaction status is mandatory and must be provided. It is represented as a JSON string with a minimum length of 0 characters and a maximum length of 256 characters. Example: Validation error.

6.5 GATEWAY ERROR RESPONSE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains header details**

1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**error**	**Class**	**M**	**This block contains API gateway error details.**
2.0.1	status	S (10)	M	The status of the transaction is mandatory and must be provided. It is represented as a JSON string and can have the following possible values: 'RJCT' indicating a failed transaction. Example: RJCT
2.0.2	code	AN (4)	M	The response code of the transaction is mandatory and must be provided. It is represented as a JSON string with a minimum of 4 characters & maximum of 4 characters. Example: A001
2.0.3	description	S (256)	M	The description of the transaction status is mandatory and must be provided. It is represented as a JSON string with a minimum length of 0 characters and a maximum length of 256 characters. Example: Incorrect Organisation ID, please resend

6.6 TRANSACTION STATUS REQUEST AND RESPONSE MESSAGE SAMPLES

6.6.1 TRANSACTION STATUS REQUEST MESSAGE SAMPLE

```
{
  "header": {
    "msgId": "1545357575227", //will be generated by Merchant.
    "orgId": "M10321242141",
    "timeStamp": "2018-12-21 09:59:35.227"
  },
  "data": [
    {
      "txnInfo": {
        "transactionId": "133b5c3e-9c9e-4b6f-900d-bd767ea9b949", //Optional
        "merchantReference": "1545357575227", //Recommended.
        "version": "2.1"
      }
    }
  ]
}
```

6.6.2 TRANSACTION STATUS RESPONSE MESSAGE SAMPLE

```
{
  “header”: {
    “msgId”: “1545357575227”,
    “timeStamp”: “2018-12-21 09:59:35.227”
  },
  “data”: [
    {
      “txnResponse”: {
        “version”: “2.1”,
        “txnStatus”: “ACTC”,
        “txnStatusDescription”: “Payment Successful”,
        “txnStatusCode”: “H502”,
        “transactionId”: “133b5c3e-9c9e-4b6f-900d-bd767ea9b949”,
        “merchantReference”: “1545357575227”,
        “customer”: {
          “hppExpressId”: “32ruewgfer43tryuewgf”,
          “accountNumber”: “XXXX3451”
        },
        “references”: [
          {
            “type”: “PT01”,
            “transactionReference”: “7rwerg8732y32y4gsfhg8”,
            “amount”: “10.00”
          }
        ]
      }
    }
  ]
}
```

6.6.3 TRANSACTION STATUS ERROR MESSAGE SAMPLE

```
{
  "header": {
    "msgId": "1545357575227",
    "timeStamp": "2018-12-21 09:59:35.227"
  },
  "data": [
    {
      "error": {
        "status": "RJCT",
        "code": "H201",
        "description": "Invalid TransactionId, please resend"
      }
    }
  ]
}
```

6.6.4 GATEWAY ERROR RESPONSE SAMPLE

```
{
  "header": {
    "msgId": "1545357575227",
    "timeStamp": "2018-12-21 09:59:35.227"
  },
  "error": {
    "status": "RJCT",
    "code": "A002",
    "description": " Invalid Request "
  }
}
```

7 REFUND API

Merchants can initiate refund transactions on behalf of your customers for PayLah!, Creditcard and PayNow transactions using the Refund API provided by DBS. This allows you to process refunds based on the entitlements provided to you as a merchant.

To initiate a refund transaction using the DBS RAPID API gateway, you would typically follow these steps:

- Retrieve the necessary transaction information required for the refund, such as the original transaction ID or any other relevant identifiers.
- Prepare the refund request with the required parameters, including the original transaction ID, merchantReference, refund amount, authentication credentials, and any additional information specified by DBS.
- Encrypt the refund request using PGP keys (Same as connect API, Transaction enquiry API).
- Make a request to the Refund API endpoint provided by DBS via the RAPID API gateway. This request should include the required parameters and any necessary headers or authentication tokens.
- Send the request to the Refund API endpoint using the appropriate HTTP method (usually POST) along with the required parameters.
- Handle the API response(encrypted), which will contain the result of the refund transaction request. The response may include information such as the refund status, timestamps, or any other relevant details.

Process the refund transaction response in your merchant application accordingly. Update your internal records or take appropriate actions based on the refund status and any additional information provided.

It's important to refer to the documentation or resources provided by DBS for specific details regarding the Refund API, including the API endpoint URL, request/response formats, authentication methods, and any additional guidelines or best practices recommended by DBS.

Please note that the availability of the Refund API may vary depending on your agreement with DBS and the entitlements provided to you as a merchant. Therefore, it's recommended to consult the DBS Implementation manager for documentation or contact their support for accurate and up-to-date information on initiating refund transactions for PayNow transactions.

7.1 URL

Method	Banking Service URL
POST	api/sg/hpp/v4/refund/transaction

7.2 REFUND REQUEST MESSAGE STRUCTURE

The table shows the structural formatting and field validations in each building block.

7.2.1 MESSAGE HEADER

The message starts with a header that contains the following information:

X-DBS-ORG_ID : <orgId>

x-api-key: <keyId>
Content-Type: text/plain

<keyId> is the key value that is exchanged with DBS. It is to be provided by DBS.

<orgId> is the Company ID given to Merchant by DBS. All alphabet characters should be in capital letters.

7.2.2 MESSAGE BODY

The transaction details are compulsory and should be encrypted using PGP with the bank's public key.

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains type of the message.**
1.0.1	msgId	AN (20)	M	A unique reference number that is generated by the merchant for each service call. The expected format is : [YYYYMMDD][sequence number]. Example: 20230425scbdf31456"
1.0.2	orgId	AN (12)	M	Unique Organization Identifier for the merchant, provided by DBS. All alphabetical characters should be in capital letters. Example: TESTXXXX01
1.0.3	timeStamp	DateTime	M	Date and time of the request message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains request data**
2.0.1	**txnInfo**	**Class**	**M**	**This block contains the refund request details.**
2.0.1.1	originalTransactionId	S (256)	M	The original transaction ID refers to the unique identifier generated by DBS at the time of the payment transaction. Example: d4520eb6-3fff-4c31-93a2-30ae374da79f
2.0.1.2	originalFundingMethod	S(25)	M	The original funding method describes the method used for the initial payment transaction. Possible values are PT01 – Indicates PayLah! payment PT02 – Indicates Credit Card PT03 – Indicates Paynow Example: PT03
2.0.1.3	merchantReference	AN(20)	M	The merchantReference indicates the unique reference number shared by the merchant. It is a significant parameter used to inquire about the transaction status in

				case DBS fails to communicate the definite status of transaction. Example: PDBA61CKJWARYBBE65C8
2.0.1.4	**amountDetails**	**Class**	**M**	**Payment amount details**
2.0.1.4.1	amount	N(6).N(2)	M	The refund amount can be either full or partial, but it should not exceed the original transaction amount. The data for the amount is a string that consists of the characters 0-9 and '.' and represents a valid decimal number with two decimal digits. Important: The numbering format should not include commas or special characters. Example: 35.75
2.0.1.4.2	currency	AN(3)	M	The currency code of the order is expressed as an ISO 4217 alpha code, for example, SGD. The currency code should be "SGD" by default. Please note that the system currently accepts SGD only. Example: SGD
2.0.1.5	refundDescription	S(256)	O	Description of the refund transaction. Example: Product retuned
2.0.1.6	version	S(5)	M	API version used for this request. Version will not be returned for error response. The value must be “2.1” by default. Example: 2.1

7.3 REFUND RESPONSE MESSAGE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**Header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222

2.0	**data**	**Array, Class**	**M**	**This block contains response data**
2.0.1	**txnResponse**	**Class**	**M**	**This block contains transaction status**
2.0.1.1	txnStatus	S(10)	M	The field indicates the status of the transaction, and it can have the following possible values: ACTC: Accepted RJCT: Rejected Please note that "ACTC" represents the status for an accepted transaction, and "RJCT" represents the status for a rejected transaction
2.0.1.2	txnStatusDescription	S (256)	M	The description of the txnStatusCode is always provided and is represented as a JSON string. It should have a minimum length of 1 character and a maximum length of 256 characters. Example: Transaction created successfully.
2.0.1.3	txnStatusCode	S(4)	M	The DBS status code represents the specific code assigned by DBS to indicate the status of the refund transaction. It provides information about the status of the refund transaction as determined by DBS. Example: R701
2.0.1.4	merchantReference	AN(20)	M	The merchantReference indicates the unique reference number shared by the merchant. It is a significant parameter used to inquire about the transaction status in case DBS fails to communicate the definite status of transaction. Example: PDBA61CKJWARYBBE65C8
2.0.1.5	transactionId	S (256)	M	HPP Transaction reference ID for the merchant refund request. Example: ff477d68-f300-4dbd-a9d4-08b1a55868f9
2.0.1.6	transactionReference	S(256)	M	Transaction reference provided by subsystem for refund transaction. Example: ff477d68-f300-4dbd-a9d4-08b1a55525321
2.0.1.7	version	S(5)	M	API version used for this request. Version will not be returned for error response. The value must be 2.1. Example: 2.1

7.4 REFUND ERROR MESSAGE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**Header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**data**	**Array, Class**	**M**	**This block contains response data**
2.0.1	transactionId	S (256)	M	HPP Transaction reference ID for the merchant refund request. transactionId is returned when available. Example: ff477d68-f300-4dbd-a9d4-08b1a55568f9
2.0.2	**error**	**Class**	**M**	**This block contains transaction status error response details**
2.0.2.1	status	S(10)	M	The status of the transaction is mandatory and must be provided. It is represented as a JSON string and can have the following possible values: 'RJCT' indicating a failed transaction. Example: RJCT
2.0.2.2	code	AN(4)	M	The response code of the transaction is mandatory and must be provided. It is represented as a JSON string with a minimum length of 4 characters and a maximum length of 4 characters. Example: R702
2.0.2.3	Description	S(256)	M	The description of the transaction status is mandatory and must be provided. It is represented as a JSON string with a minimum length of 0 characters and a maximum length of 256 characters. Example: Invalid TransactionId, please resend

7.5 GATEWAY ERROR RESPONSE STRUCTURE

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**Header**	**Class**	**M**	**This block contains type of the message**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**error**	**Class**	**M**	**This block contains API gateway error details.**
2.0.1	status	S (10)	M	The status of the transaction is mandatory and must be provided. It is represented as a JSON string and can have the following possible values: 'RJCT' indicating a failed transaction. Example: RJCT
2.0.2	code	AN (4)	M	The response code of the transaction is mandatory and must be provided. It is represented as a JSON string with a minimum length of 4 characters and a maximum length of 4 characters. Example: A001
2.0.3	description	S (256)	M	The description of the txnStatusCode is mandatory and must be provided. It is represented as a JSON string with a minimum length of 0 characters and a maximum length of 256 characters. Example: Incorrect Organisation ID.

7.6 REFUND REQUEST AND RESPONSE MESSAGE SAMPLES

7.6.1 REFUND REQUEST MESSAGE SAMPLE

```
{
        "header":{
                "msgId":"07683747313384197",
                "orgId":"HPP000999",
                "timeStamp":"2021-03-05T11:51:21.847Z"
        },
        "data":[
                {
                "txnInfo":{
                "originalTransactionId":"d4520eb6-3fff-4c31-93a2-30ae374da79f",
                "originalFundingMethod":"PT01",
                "merchantReference":"M9087655666666",
                "amountDetails":{
                "amount":50.50,
                "currency":"SGD"
                },
                “version”: “2.1”
            }
          }
        ]
}
```

7.6.2 REFUND RESPONSE MESSAGE SAMPLE

```
{
  "header": {
    "msgId": "0125349934912826687",
    "timeStamp": "2021-04-05 11:42:56.632"
  },
  "data": [
    {
      "txnResponse": {
        "txnStatus": "ACTC",
        "txnStatusDescription": "Refund Transaction is successful",
        "txnStatusCode": "R701",
        "merchantReference": "refund001test5d",
        "transactionId": "a969a7a7-91e9-4d85-904d-c37fdae1af1d",
        "transactionReference": "1617622976244st5d060",
        "version": "2.1"
      }
    }
  ]
}
```

7.6.3 REFUND ERROR MESSAGE SAMPLE

```
{
   "header": {
      "msgId": "1545357575227",
      "timeStamp": "2018-12-21 09:59:35.227"
   },
   “data”: [
      {
         "transactionId": "e92843b0-e248-419e-9c0d-da768cbd4e7a",
         "error": {
            "status": "RJCT",
            "code": "H022",
            "description": "Invalid originalFundingMethod "
         }
      }
   ]
}
```

7.6.4 GATEWAY ERROR RESPONSE SAMPLE

```
{
   "header": {
      "msgId": "1545357575227",
      "timeStamp": "2018-12-21 09:59:35.227"
   },
   "error": {
      "status": "RJCT",
      "code": "A001",
      "description": "Organization ID is incorrect"
   }
}
```

8 MERCHANT TRANSACTION NOTIFICATION (WEBHOOK) API

When using the DBS HPP Connect API, merchants can include a callback URL as part of their API request. This callback URL is used by DBS to trigger a webhook, providing transaction details to the merchant at the end of each transaction. Here's how the process typically works:

- **Merchant includes a callback URL:** When making a request to the DBS HPP Connect API, the merchant includes their desired callback URL as one of the parameters or fields in the API request. This URL specifies where DBS should send the webhook notification

- Make sure callback url is followed with below:

    - **HTTPS Secured:** Ensure that your callback URL uses HTTPS (Hypertext Transfer Protocol Secure) instead of HTTP. HTTPS encrypts the data transmitted between the client and the server, providing a secure connection. This is crucial for protecting sensitive information during the callback process.
    - **Categorized URL:** It's advisable to categorize your callback URL appropriately to ensure it meets any security or compliance requirements. This categorization helps in managing access and enforcing policies related to the callback URL. URL categorization has to be done as per the below procedure:
        - Go to https://sitereview.bluecoat.com
        - Enter the web domain (ex: merchant.com)
        - Select Filtering Service – Blue Coat ProxySG
        - Select the correct Category for your business and submit for review.
    - **Whitelisting for DBS Outgoing Calls:** Contact DBS to request the whitelisting of your callback URL for DBS outgoing calls. DBS may have a whitelist of approved URLs that we allow for callback notifications. By whitelisting your URL, you ensure that DBS can successfully initiate webhook notifications to your system.

- **Transaction completes:** Once the transaction is completed, DBS triggers a webhook in encrypted format (Encrypt using Merchant’s Public key and sign using the HPP private key) to the callback URL specified by the merchant in the Connect API request.

- **Merchant receives the webhook:** The merchant's server or application, which is responsible for handling incoming webhooks, receives the webhook from DBS at the specified callback URL.

- **Process the webhook payload:** The merchant's server or application processes (Decrypts using Merchant’s private key and unsign using HPP public key) the webhook payload received from DBS. The payload typically includes transaction details such as the transaction ID, status, payment amount, and any other relevant information.

- **Update merchant records or take appropriate actions:** Based on the webhook payload and the specific requirements of the merchant's application, the merchant can update their internal records, trigger further actions, or perform any other necessary processing related to the transaction.

  By utilizing the webhook functionality provided by DBS, the merchant can receive real-time transaction updates without having to continuously poll or query the DBS HPP API for the status of each transaction. The webhook allows for a more efficient and timely exchange of transaction information between DBS and the merchant's application.

  It's important to refer to the documentation or resources provided by DBS for specific details on how to configure and handle webhooks, including the required parameters in the Connect API request and the expected format of the webhook payload.

8.1 URL

Method	Merchant Callback URL
POST	Merchant shares the Callback URL in Connect API

All fields are compulsory and should be encrypted using PGP with the partner's public key and sign using DBS-HPP private key.

8.2 RESPONSE MESSAGE STRUCTURE

The table shows the structural formatting and field validations in each building block.

S/N	Field Name	Type (Length)	M/O/C	Description
1.0	**header**	**Class**	**M**	**This block contains header details**
1.0.1	msgId	AN (20)	M	Each API call is assigned a unique message reference number. Example: 20230425scbdf32457
1.0.2	timeStamp	DateTime	M	Date and time of the response message. The expected format is : YYYY-MM-DDTHH:MM:SS.mmm Example: 2021-02-14T15:07:26.12.222
2.0	**txnResponse**	**Class**	**M**	**This block contains transaction details**
2.0.1	txnStatus	S(10)	M	Indicates the transaction status. ‘ACTC’ = accepted ‘RJCT’ = rejected ‘PDNG’ = pending Note: For refunds status enquiry, txnStatus could be either ACTC or RJCT only.
2.0.2	txnStatusDescription	S (256)	M	The description of the txnStatusCode is always provided and is represented as a JSON string. It should have a minimum length of 1 character and a maximum length of 256 characters. Example: Transaction created successfully.
2.0.3	txnStatusCode	S (10)	M	The status code represents the code returned by HPP for the transaction. It indicates the current status of the transaction. Example: H502
2.0.4	transactionId	S (256)	M	HPP Transaction reference ID for the merchant payment request. Example: ff477d68-f300-4dbd-a9d4-08b1a55568f9

2.0.5	merchantReference	AN(20)	M	The merchantReference indicates the unique reference number shared by the merchant. It is a significant parameter used to inquire about the transaction status in case DBS fails to communicate the definite status of transaction. Example: PDBA61CKJWARYBBE65C8
2.0.6	**customer**	**Class**	**C**	**Customer specific Information**
2.0.6.1	hppExpressId	S(256)	C	HPP generated ID for a user/customer with saved / tokenized payment details. Example: MERCUST456702
2.0.6.2	accountNumber	S(20)	C	Masked PayLah! phone number, masked card number for credit card transaction. Card format: 5432 15xx xxxx 4567 PayLah! format: xxxx4597 PayNow: this will not be returned to customer. Example: xxxx4597
2.0.7	**References**	**Array, Class**	**M**	**Transaction reference pertaining to DBS payment sources**
2.0.7.1	type	S (128)	M	The payment method used in this transaction represents the method chosen by the customer to make the payment. Possible values are : PT01 – Indicates PayLah! payment PT02 – Indicates Credit Card PT03 – Indicates PayNow Example: PT03
2.0.7.2	transactionReference	S (256)	C	The transaction reference refers to the unique identifier associated with the transaction from the appropriate payment method. It is a reference number or code that uniquely identifies the specific transaction. Note: transactionReference will be optional for tokenization or ECSetupStatus responses. Example: 7rwerg8732y32y4gsfhg8
2.0.7.3	amount	N(6).N(2)	M	The amount indicates the amount that the transaction made. The data for the amount is a string that consists of the characters 0-9 and '.' and represents a valid decimal number with two decimal digits. Important: The numbering format should not include commas or special characters.

				Example: 100.50
2.0.8	version	S(5)	M	Indicates api version Example: “2.1”

8.3 WEBHOOK RECEIPT CONFIRMATION

Merchant should provide the acknowledgement after the receipt of the server notification API call.
Response Code - 200 - If the Server Notification is success.
Response Code - 4xx/5xx - If the server Notification failed.

8.4 WEBHOOK RESPONSE MESSAGE SAMPLES

Refer to the following for various callback/webhook responses after a successful or unsuccessful payment.

8.4.1 SUCCESSFUL WEBHOOK RESPONSE – PAYLAH!

```
{
  "header": {
    "msgId": "1545357575227", //will be generated by HPP
    "timeStamp": "2018-12-21 09:59:35.227"
  },
  "txnResponse": {
    "txnStatus": "ACTC",
    "txnStatusCode": "H200",
    "txnStatusDescription": "Request Processed successfully.",
    "transactionId": "133b5c3e-9c9e-4b6f-900d-bd767ea9b949",
    "merchantReference": "1545357575227",
    "customer": {
      "hppExpressId": "32ruewgfer43tryuewgf",
      "accountNumber": “XXXX3451"
    },
    "references": [
      {
        "type": "PT01",
        "transactionReference": "23020618492899231960 ",
        "amount": "10.00"
      }
    ],
    "version": “2.1"
  }
}
```

8.4.2 SUCCESSFUL WEBHOOK RESPONSE – CREDIT CARD

```
{
  "header": {
    "msgId": "1545357575227", //will be generated by HPP
    "timeStamp": "2018-12-21 09:59:35.227"
  },
  "txnResponse": {
    "txnStatus": "ACTC",
    "txnStatusCode": "H200",
    "txnStatusDescription": "Request Processed successfully.",
    "transactionId": "133b5c3e-9c9e-4b6f-900d-bd767ea9b949",
    "merchantReference": "1545357575227",
    "customer": {
      "hppExpressId": "32ruewgfer43tryuewgf",
      "accountNumber": “5432 13XX XXXX 4567"
    },
    "references": [
      {
        "type": "PT02",
        "transactionReference": "23020618492899231960 ",
        "amount": "10.00"
```

```
            }
          ],
          "version": “2.1"
        }
      }
```

8.4.3 SUCCESSFUL WEBHOOK RESPONSE – PAYNOW

```
      {
        "header": {
          "msgId": "1545357575227", //will be generated by HPP
          "timeStamp": "2018-12-21 09:59:35.227"
        },
        "txnResponse": {
          "txnStatus": "ACTC",
          "txnStatusCode": "H200",
          "txnStatusDescription": "Request Processed successfully.",
          "transactionId": "133b5c3e-9c9e-4b6f-900d-bd767ea9b949",
          "merchantReference": "1545357575227",
          "references": [
            {
              "type": "PT03",
              "transactionReference": "23020618492899231960 ",
              "amount": "10.00"
            }
          ],
          "version": “2.1"
        }
      }
```

8.4.4 UNSUCCESSFUL WEBHOOK RESPONSE – PAYLAH!, CREDITCARD, PAYNOW

```
      {
        "header": {
          "msgId": "1545357575227",
          "timeStamp": "2018-12-21 09:59:35.227"
        },
        "txnResponse": {
          "txnStatus": "RJCT",
          "txnStatusCode": "H201",
          "txnStatusDescription": "Invalid TransactionId, please resend",
          "transactionId": "133b5c3e-9c9e-4b6f-900d-bd767ea9b949",
          "merchantReference": "1545357575227"
        }
      }
```

8.5 RETRY MECHANISM

The retry mechanism for Server to Server (webhook) notifications in the DBS Hosted Payment Platform (HPP) operates as follows:

Retry Attempts: DBS HPP will automatically retry sending a webhook notification up to three times if the initial attempt fails.

Increasing Intervals: The retry intervals progressively increase with each attempt. The first retry occurs 1 second after the initial failure, the second after 2 seconds, and the third after 4 seconds.

Failure Logging: Should all three retry attempts fail, DBS HPP records the failure and ceases to send further notifications to the merchant's webhook endpoint for that specific transaction. This means that the merchant will not receive any further webhook notifications for that particular transaction.

To stay informed of the latest transaction status when webhook notification retries are unsuccessful, merchants are advised to use the Transaction Status API. This API allows merchants to query the current status of the transaction directly from DBS HPP, ensuring they receive the most recent information.

9 TRANSACTION REJECTION REASON CODES

9.1 APPENDIX A: API ATEWAY REJECT CODES WITH REASON

The table describes the rejection reason for each reason code. Error code 1-14 is from the DBS API Gateway. The response for error codes below will be a plain JSON.

Module	S/N	HTTP Code	Code	Description
API Gateway Rejection Code	1	401	A001	Organization ID is incorrect
	2	429	A002	Maximum transaction transmission is exceeded
	3	400	A003	Invalid Request
	4	401	A004	Security credential is incorrect
	5	504	A005	Transaction has timed out
	6	500	A006	Gateway System Error
	7	500	A009	Internal Server Error
	8	403	A010	Security check failed
	9	401	A011	Invalid API Key
	10	401	A012	User is not authorized to access this API
	11	401	A014	Invalid Authorization code
	12	401	A015	Invalid Authorization Header
	13	401	A016	Use is Inactive
	14	401	A017	You are not subscripted to the financial statement

9.2 HPP CONNECT API REJECT CODES WITH REASON

The table describes the rejection reason for each reason code.

API Name	S/N	HTTP Code	Code	Description
HPP Connect /Status /Refund request error codes	1	200	H001	Incorrect Organisation ID.
	2	200	H003	Merchant not entitled. Please reach out to your relationship manager.
	3	200	H004	Transaction amount cannot be zero or less Note: If typeOfPayment=w03 and PreferredPaymentMethod=PT02 then transaction amount can be allowed as Zero but not negative values.
	4	200	H005	Maximum allowed transaction amount is exceeded.

	5	200	H006	Request timestamp missing in payload. Please add and resend.
	6	200	H007	returnUrl missing in payload. Please add and resend.
	7	200	H008	cancelUrl missing. Please add and resend.
	8	200	H009	App deeplinkUrl is missing. Please add and resend.
	9	200	H011	CallbackUrl is missing. Please add and resend.
	10	200	H013	merchantReference is missing. Please add and resend.
	11	200	H014	amount is missing. Please add and resend.
	12	200	H015	currency is missing. Please add and resend.
	13	200	H016	channelType is missing. Please add and resend.
	14	200	H017	Invalid currency code.
	15	200	H018	Invalid channelType.
	16	200	H019	Sorry, we are unable to process your payment. Please return to merchant and try using a different payment method.
	17	200	H020	originalTransactionId is missing, please resend.
	18	200	H021	originalFundingMethod is missing, please resend.
	19	200	H022	Invalid originalFundingMethod
	20	200	H023	Unable to identify valid hppExpressId Note: This message will be applicable for typeOfPayment= w03 and a01
	21	200	H024	TypeOfPayment is missing or Invalid
	22	200	H025	preferredPaymentMethod is Invalid
	23	200	H026	Version mismatch, send valid request
	24	200	H201	Invalid TransactionId, please resend.
	25	200	H202	Transaction cancelled by User.
	26	200	H203	Your Express Setup Rejected/Expired
	27	200	H804	Transaction Limit exceeded, Unable to proceed
	28	200	H998	Login failed
	29	200	H999	Validation error
	30	200	H901	Payment transaction initiated
	31	200	H906	Payment transaction is in process
	32	200	H919	Payment transaction successful

	33	200	H920	Payment transaction pending
	34	200	H922	Payment transaction is failed
	35	200	H923	Invalid Payment transaction
	36	200	H000	Request Processed successfully.
	37	200	H081	Express payment setup is disabled
HPP Transaction Status API	38	200	H201	Invalid TransactionId, please resend.
	39	200	H204	Invalid merchant reference, please resend
HPP Refund API	40	200	H020	originalTransactionId is missing, please resend.
	41	200	H021	originalFundingMethod is missing, please resend
	42	200	H022	Invalid originalFundingMethod
	43	200	H014	amount is missing. Please add and resend.
	44	200	H015	currency is missing. Please add and resend.
	45	200	H017	Invalid currency code.
	46	200	H201	Invalid TransactionId, please resend.

9.3 APPENDIX B: HPP RESPONSE CODES FOR THE HPP REQUESTS

The table describes the HPP response codes for the PayLah! payment method.

Sl.No	HTTP Code	HPP Code	PayLah!	Credit Card	Paynow	Description
1	200	H500	Y	N	N	Invalid PayLah! wallet
2	200	H503	Y	N	N	PayLah! wallet is blocked
3	200	H502	Y	Y	Y	Payment successful
4	200	H600	Y	N	N	Express checkout setup successful
5	200	H601	Y	N	N	Express checkout setup failed
6	200	H501	Y	Y	Y	Pending
7	200	H504	Y	Y	Y	Your transaction has expired. Please return to the merchant's checkout page.
8	200	H505	Y	Y	Y	Payment is expired or rejected
9	200	H506	Y	Y	Y	payment is unsuccessful. Please reinitiate the payment
10	200	H507	Y	N	Y	Unable to decrypt the payment request
11	200	H508	Y	N	Y	Invalid transaction amount
12	200	H509	Y	N	Y	Payment amount limit exceed
13	200	H511	Y	N	Y	Field size is not in expected size.

14	200	H512	Y	N	Y	Mandatory field is missing.
15	200	H513	Y	N	Y	Invalid merchant transaction ID
16	200	H514	Y	N	Y	Invalid currency code.
17	200	H515	Y	N	Y	Invalid transaction reference
18	200	H516	Y	N	Y	Invalid number format.
19	200	H517	Y	N	N	Express payment transaction pending
20	200	H518	Y	N	Y	Unable to proceed, Eligible account not found
21	200	H521	Y	N	Y	Merchant Keys are empty.
22	200	H522	Y	N	Y	Merchant Secret Key is mismatching.
23	200	H523	Y	N	Y	Duplicate Merchant Txn reference number
24	200	H525	Y	N	N	Invalid PayLah! transaction.
25	200	H526	Y	N	N	Duplicate Express Checkout Setup request
26	200	H527	Y	N	Y	Error in processing the transaction
27	200	H528	Y	N	Y	Invalid Merchant Code.
28	200	H529	Y	N	Y	Unable to Process transaction.
29	200	H530	Y	N	Y	Merchant setup id is not available.
30	200	H531	Y	N	Y	Invalid transaction
31	200	H541	Y	N	Y	encryption failed for merchant’s request
32	200	H551	Y	N	Y	Merchant Txn Record is not found.
33	200	H552	Y	N	N	No Record found for Txn Ref Number.
34	200	H553	Y	N	N	Wallet not found
35	200	H526	Y	N	N	Duplicate Express Checkout Setup request
36	200	H527	Y	N	Y	Error in processing the transaction
37	200	H555	Y	N	Y	Decryption failed for merchant request
38	200	H018	Y	N	Y	Txn Amount cannot be zero or less
39	200	H524	Y	N	Y	Duplicate merchant user Id
40	200	H556	Y	N	Y	Daily transfer limit exceeded
41	200	H571	Y	N	N	PayLah! wallet temporarily disabled
42	200	H572	Y	N	N	Wallet Blocked.
43	200	H573	Y	N	N	Wallet closed.
44	200	H581	Y	N	N	UUID Inactive
45	200	H582	Y	N	N	Internal Server Error
46	200	H583	Y	N	N	Transaction is captured already

47	200	H584	Y	N	N	Transaction is voided
48	200	H585	Y	N	N	Capture can't be performed on this action
49	200	H586	Y	N	N	Void can't be performed on this action
50	200	H590	Y	N	Y	Invalid parameter, Unable to proceed
51	200	H999	Y	N	Y	Validation error
52	200	R701	Y	Y	Y	Refund txn is successful
53	200	R702	Y	Y	Y	Data not found for your Transaction
54	200	R703	Y	Y	Y	Refund amount more than eligible amount.
55	200	R704	Y	Y	Y	Refund Greater than Transaction Amount.
56	200	R705	Y	Y	Y	Original Txn is expired, hence cannot process refund. Transaction is unsuccessful.
57	200	R799	Y	Y	Y	Transaction is unsuccessful.
58	200	R706	Y	Y	Y	Refund transaction is pending
59	200	R707	Y	Y	Y	Refund transaction timed out
60	200	R708	Y	Y	Y	Refund transaction is Initiated

9.4 APPENDIX C: ENDPOINT URLS

End point URLs on DBS hosted payments platform application

SL NO	ENVIRONMENT	BASE URL
1	UAT	https://testcld-enterprise-api.dbs.com/
2	Production	https://enterprise-api.dbs.com/

10 FREQUENTLY ASKED QUESTIONS

10.1 GENERAL FAQs

1. What are the payment methods currently functional?

   Ans. DBS hosted payment platform currently functional with PayLah!, Creditcard and Paynow

2. What is PGP Encryption?

   Ans. PGP (Pretty Good Privacy) encryption is a data encryption method used for securing the confidentiality of digital communication. It's widely used for encrypting and decrypting **texts**, emails, files and directories to increase the security of data communications.

   Technique:

   Public-Key Cryptography: PGP uses a public key for encryption and a private key for decryption. The public key is shared openly, while the private key is kept secret. When a sender wants to send a secure message, they use the recipient's public key to encrypt the message. The recipient then uses their private key to decrypt it.

3. How does PGP encryption and signing helps?

   Ans: PGP (Pretty Good Privacy) encryption and signing are processes used to secure and verify digital communications, such as emails or files. Here's an overview of how each process works:

   PGP Encryption

   Generating a Key Pair: First, a user generates a key pair, which includes a public key and a private key. The public key is shared with others to encrypt messages to you, while the private key is kept secret and is used to decrypt messages.

   Encrypting a Message: To send an encrypted message, the sender uses the recipient's public key to encrypt the message. This can be done using PGP software or email clients that support PGP. The encrypted message can only be decrypted by the recipient's private key.

   Decrypting a Message: Upon receiving the encrypted message, the recipient uses their private key to decrypt it. Since only the intended recipient has access to the corresponding private key, the message remains secure during transit.

   PGP Signing

   Creating a Digital Signature: When sending a message, a sender can also sign it using their private key. This is done by creating a hash (a unique digital fingerprint) of the message and then encrypting the hash with the sender's private key.

   Verifying a Signature: The recipient, upon receiving the message, can use the sender's public key to decrypt the hash and compare it to the hash they generate from the received message. If the hashes match, it confirms that the message has not been altered since it was signed and that it indeed comes from the holder of the private key.

   Combined Use for Maximum Security

   Encrypt and Sign: For maximum security, a sender might both sign and encrypt a message. First, the sender signs the message with their private key, then encrypts the entire message (including the signature) with the recipient's public key.

Decrypt and Verify: The recipient first decrypts the message with their private key and then verifies the sender's signature with the sender's public key.

Using PGP for both encrypting and signing is a robust way to ensure the confidentiality, integrity, and authenticity of digital communications. It protects the message from being read by unauthorized parties and also verifies that the message hasn't been tampered with and that it comes from a legitimate source.

10.2 HPP – PAYLAH! FAQs

1. What is PayLah!?
   Ans: PayLah! is the mobile wallet used by DBS consumers to make retail payment.

2. What is the max amount currently supported by PayLah! per transaction?
   Ans: Max amount that supported by PayLah! per customer per transaction is SGD 2000

3. What is the max amount currently supported per customer per day by PayLah!?
   Ans: Max amount that supported by PayLah! per customer per day is SGD 2000.

4. Does hpp supports website and mobile app-based payments?
   Ans: DBS hpp supports merchant with both web based and app-based payments.
   This includes, desktop web, mobile web, iOS app and android app

5. Does hpp supports PayLah! payment for guest checkouts?
   Ans. Yes, Users, who are not registered with merchant and are doing instant checkouts will still be allowed to make payments.
   In this case merchants must send the transaction registration request(connect api call) with type of payment as w01(guest/onetime users).

6. Does hpp supports PayLah! payment for registered merchant user checkouts?
   Ans. Yes, Users, who are registered with merchant and are doing instant checkouts will still be allowed to make payments.
   In this case merchants must send the transaction registration request(connect api call) with type of payment as w02(guest/onetime users).

7. Does hpp supports PayLah! support express payments?
   Ans. Yes, Users must opt for the express setup(tokenization) to access the express payment capability for all the future transactions.
   DBS will send the hppExpressId in response the express setup. So that, for the subsequent transaction merchants must send the hppExpressId to identify the user and to route to express payment capability.

8. Does user allowed to register multiple PayLah! wallets for express payment?
   Ans. No, User can use max 1 PayLah! wallet per merchant registration.

9. Does user allowed to change PayLah! account that already registered for express payments?
   Ans. Yes, to do this user must unregister the existing account from PayLah! wallet. And then user must do the fresh express setup from the merchant account.

10. What are the possible of payment methods does support by PayLah!?

    Ans. hpp supports guest and the registered users

    For guest users hpp support onetime payments only. Tokenization will not be supported.

    For registered customers HPP supports,

    a) Onetime payment.
    b) Express setup
    c) Express setup and payment
    d) Express payments
    e) API based payments

10.3 HPP - CREDIT CARD FAQS

1. What creditcard?

   Ans: A credit card is a type of credit facility, provided by banks that allow customers to borrow funds within a pre-approved credit limit. It enables customers to make purchase transactions on goods and services.

2. What are the card schemes that are supported by DBS hosted payments?

   Ans: At this moment only Visa and Mastercard are allowed to make the payments via DBS hosted payment platform.

   Card schemes such as Amex, JCB, Diners club are not supported currently.

3. What is the max amount currently supported per customer per day by creditcard?

   Ans: There is no such hard and fast rule at HPP end. Standard limits set by issues and card schemes will be applied.

4. What is the max amount currently supported per customer per day by PayLah!?

   Ans: there are such restrictions at hpp end. Standard limits set by issues and card schemes will be applied.

5. Do the users allowed to use foreign cards?

   Ans: Yes, however hpp supports all the transactions for our merchants in SGD only.

6. Does hpp supports website and mobile app-based payments?

   Ans: DBS hpp supports merchant with both web based and app-based payments.

   This includes, desktop web, mobile web, iOS app and android app.

7. Does hpp supports card payment for guest checkouts?

   Ans. Yes, Users, who are not registered with merchant and are doing instant checkouts will still be allowed to make card payments.

   In this case merchants must send the transaction registration request(connect api call) with type of payment as w01(guest/onetime users).

8. Does hpp supports card payment for registered merchant user checkouts?

   Ans. Yes, Users, who are registered with merchant and are doing instant checkouts will still be allowed to make payments.

   In this case merchants must send the transaction registration request(connect api call) with type of payment as w02(guest/onetime users).

9. Does hpp supports the tokenization (save card) to enable the express payments?

   Ans. Yes, Users must opt for the express setup(tokenization) to access the express payment capability for all the future transactions.

   DBS will send the hppExpressId in response the express setup. So that, for the subsequent transaction merchants must send the hppExpressId to identify the user and to route to express card payment capability.

10. Does user allowed to save multiple cards for express payment?

    Ans. Yes, User can use max 2 (product level configuration) 2 cards per merchant registration.

11. What are the possible of payment methods does support by creditcard?

    Ans. hpp supports guest and the registered users

    For guest users hpp support onetime payments only. Tokenization will not be supported.

    For registered customers HPP supports,

    a) Onetime payment.
    b) Tokenization (Save card)
    c) Tokenization and payment
    d) Express payments.

10.4 HPP – PAYNOW FAQs

1. What are the types of journeys supported for Paynow payment rails?

   Ans: For PayNow payment rails, the following types of journeys are supported:

   Seamless Journey:

   - Users with DBS/POSB savings accounts can authenticate and make payments seamlessly to the merchant's PayNow account.
   - This journey leverages the existing banking relationship, allowing users to make payments directly from their savings accounts without the need for additional authentication steps.

   Scan & Pay Journey:

   - The HPP (Hosted Payment Page) generates a PayNow QR code for users to scan and initiate payment.
   - Buyers are given a specified timeframe to scan the QR code and make payment to the merchant's Unique Entity Number (UEN).
   - The buyer performs this payment from their local bank/payment app that support paynow QR scan/upload and pay.
   - If the buyer fails to scan/upload the QR code within the specified timeframe, the QR code will expire.
   - The specified timeframe for scanning the QR code and initiating payment can be configured during the merchant onboarding process or through interactions with the Relationship Manager (RM).
   - The maximum timeframe for scanning and initiating payment is typically set to 13 minutes.
   - These journeys provide different options for users to make payments through PayNow, enabling seamless payments for account holders and a scan-and-pay experience for other users using QR codes.

   Please note that the specific implementation and configuration details of these journeys may vary based on the merchant's onboarding process, PayNow configuration, and any specific requirements or agreements with DBS. It's recommended to consult the documentation or resources provided by DBS or engage with our support or relationship manager for more accurate and up-to-date information on the supported journeys for PayNow payment rails.

2. Are there any restrictions on transaction amount ?

Ans: Yes, there are restrictions on transaction amounts for PayNow payments, depending on the payment method. PayNow users opting to pay through.

DBS/POSB accounts:

- There is a maximum transaction amount limit of SGD 200,000 per day for payments made through DBS/POSB accounts.
- Users cannot exceed this limit when making PayNow payments using their DBS/POSB savings accounts.

Scan & Pay payments:

- No specific restrictions are applied to transaction amounts for Scan & Pay payments.
- Users can initiate payments by scanning the PayNow QR code and there is no predefined limit on the transaction amount for this payment method.

It's important to note that these restrictions may be subject to change and could vary based on the specific agreements, policies, and regulations of DBS and the PayNow system. It's recommended to refer to the official documentation or reach out to DBS or PayNow support for the most up-to-date and accurate information on transaction amount restrictions for PayNow payments.

****The-End*****