



Dragonpay Online Payment

Merchant Payment Switch API

Version 2.17 – July 15, 2024

Table of Contents

| | |
|---|----|
| Table of Contents..... | 2 |
| 1. About this Document..... | 4 |
| 2. Intended Audience..... | 4 |
| 3. Change Log..... | 5 |
| 4. Introduction..... | 6 |
| 4.1 What is online bank debit payment?..... | 6 |
| 4.2 How does online bank debit payment work?..... | 8 |
| 5. Payment Switch API..... | 10 |
| 5.1 System Requirements..... | 10 |
| 5.2 Requesting a Payment..... | 11 |
| 5.2.1 Request Parameters..... | 12 |
| 5.2.2 Response Parameters..... | 16 |
| 5.2.3 Payment Completion Response Parameters..... | 17 |
| 5.3 Additional Support Functions..... | 19 |
| 5.3.1 Transaction Status Inquiry..... | 19 |
| 5.3.1.1 Request Parameters..... | 19 |
| 5.3.1.2 Response Parameters..... | 20 |
| 5.3.2 Cancellation of Transaction..... | 21 |
| 5.3.2.1 Request Parameters..... | 21 |
| 5.3.2.2 Response Parameters..... | 21 |
| 5.3.3 Multiple-Use Virtual Accounts and Lifetime Id Functions..... | 22 |
| 5.3.3.1 Request Parameters for Creation of MUVA or LID..... | 22 |
| 5.3.3.2 Response Parameters for Creation of MUVA or LID..... | 24 |
| 5.3.3.3 Request Parameters for Activation/Deactivation of MUVA or LID..... | 25 |
| 5.3.3.4 Response Parameters for Activation/Deactivation of MUVA or LID..... | 25 |
| 5.3.3.5 Generating Static QR for Existing MUVA or LID..... | 25 |
| 5.3.3.6 Request Parameters for Retrieving MUVA or LID..... | 25 |
| 5.3.3.7 Response Parameters for Retrieving MUVA or LID..... | 26 |
| 5.3.3.7 Updating Lifetime Users for MUVA and LID..... | 26 |
| 5.3.3.8 Response parameters for Updating Lifetime Users for MUVA and LID..... | 27 |
| 5.3.4 Transaction History Functions..... | 28 |
| 5.3.4.1 Request for Transaction History..... | 28 |
| 5.3.4.2 Response for Transaction History..... | 28 |
| 5.3.4.3 Request for History of Successfully Completed Transactions..... | 28 |
| 5.3.4.4 Response for History of Successfully Completed Transactions..... | 28 |
| 5.4 Customization of Payment Selection..... | 29 |
| 5.4.1 Simple Control..... | 29 |
| 5.4.1.1 Filtering Payment Channels..... | 29 |
| 5.4.1.2 Pre-selecting Payment Channels..... | 30 |
| 5.4.2 Advanced Control..... | 31 |
| 5.4.2.1 Determining Available Payment Channels..... | 31 |
| 5.4.2.1.1 GetAvailableProcessors Request Parameters..... | 32 |

| | |
|---|----|
| 5.4.2.1.2 GetAvailableProcessors Response Parameters..... | 32 |
| Appendix 1 – Currency Codes..... | 34 |
| Appendix 2 – Error Codes..... | 35 |
| Appendix 3 – Status Codes..... | 36 |

1. About this Document

This document describes the Application Programming Interface (API) between Payment Switch (PS) and the Merchant's e-commerce website. The PS is responsible for communicating with the financial partner's (eg. Bank) payment gateway for payment requests using a separate API. Upon validating the request, it redirects the end-user to his funding source of choice. The information needed by the PS to process a merchant payment for a transaction is transmitted using the API described in this document.

This document provides an overall introduction to the system, including its general architecture and structure. It then goes into detail on how to actually implement the system.

If you have any questions please do not hesitate to contact **sales@dragonpay.ph**.

2. Intended Audience

The intended audience for this document is technical personnel or programmers with background knowledge of programming and e-commerce. The examples in this document are written in Microsoft C# .NET. However, the programmer is free to implement the interfaces using other programming languages as long as they conform to Web standards such as HTTP GET, Name-Value Pair, and JSON/RESTcalls.

3. Change Log

| Version | Date | Changes |
|---------|----------------|--|
| 2.00 | Sept 9, 2019 | Base Version for REST v1 implementation |
| 2.01 | Jan 16, 2020 | Additional fields for Collect |
| 2.02 | Mar 20, 2020 | Transaction History retrieval endpoints |
| 2.03 | Jun 17, 2020 | Added missing Section 5.2.3 |
| 2.04 | Jul 10, 2020 | Added lifetime id activation support |
| 2.05 | Jul 14, 2020 | Modified transaction history inquiry to use GET param |
| 2.06 | May 15, 2021 | Added note to use clear text passwords in frontend application |
| 2.07 | Jul 9, 2022 | Added <i>PreferredId</i> parameter for 5.3.3.1 |
| 2.08 | Nov 21, 2022 | Added INPY to sample procId in 5.4.1.2 |
| 2.09 | Jan 11, 2023 | Added amount/ccy/procId to 5.2.3; added note on txnid format for lifetime id's |
| 2.10 | Mar 2, 2023 | Added documentation for MUVA vs LID |
| 2.11 | | Added Bin optional field for MUVA creation |
| 2.12 | Jan 27, 2024 | Updated email information |
| 2.13 | Feb 26, 2024 | Added Updating lifetime user for LID and Muva |
| 2.14 | Mar 03, 2024 | Added RefNo, MobileNo, ProcMsg and Fee on 5.3.1.2 Response Parameters |
| 2.15 | March 14, 2024 | Added ABQR sample request, note and sample response. |
| 2.16 | May 9, 2024 | Added information for generating static QR for MUVA/LID |
| 2.17 | July 15, 2024 | Updated Appendices Terminology updated- Collection API key (old password) |

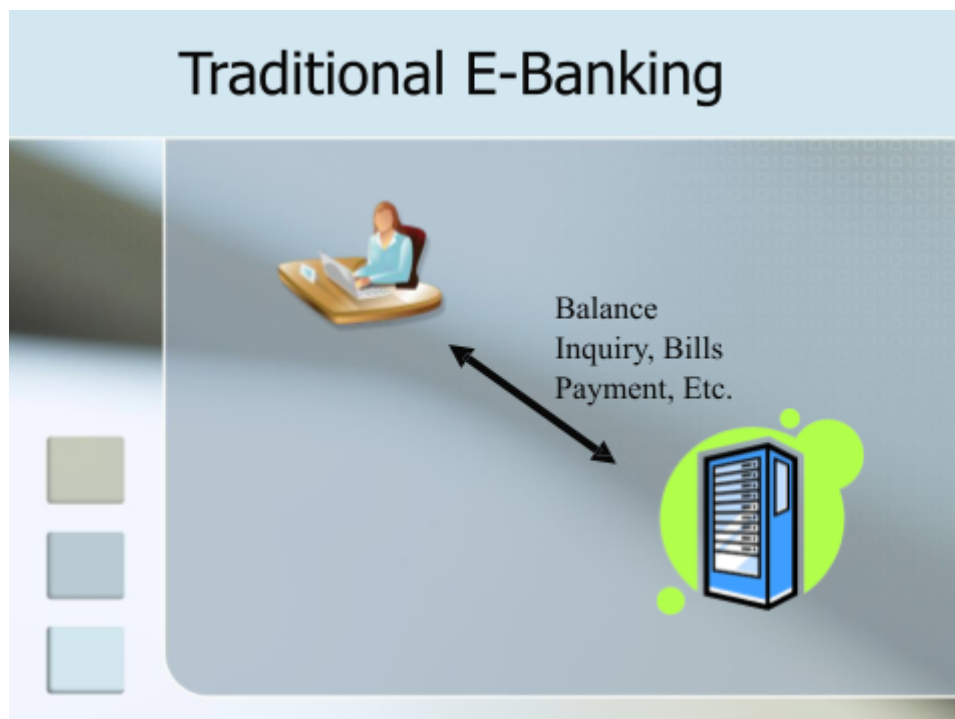
4. Introduction

E-commerce is gaining more and more acceptance by the general public each day. Its full potential, however, is hampered by the lack of available online payment options. While credit card remains to be the most popular online payment option, most consumers shy away from it for fear of getting their card information compromised. Online merchants are also very wary of credit cards because of high fraud rate. And for those selling high-ticket items, the percentage-based fee structure of credit cards is not appealing. Furthermore, only a small percentage of the population has access to credit cards because of credit history requirements.

Online bank debit payment presents a very effective alternative to this dilemma. Opening a bank account is certainly simpler than opening a credit card account. This presents a larger potential customer base to online merchants. The online banking interface is also inherently more secure than the usual credit card interface. This gives assurance to the customer that the transaction is safe. And because there is no concept of chargebacks with debit payments, merchants are also assured of payments for their products or services.

4.1 *What is online bank debit payment?*

In a typical online banking session, bank customers can perform basic functions such as balance inquiry, bills payment, checkbook reorder, and funds transfer remotely from their homes or offices. The bank's online interface is simply accessed using a web browser over a secure channel (https).

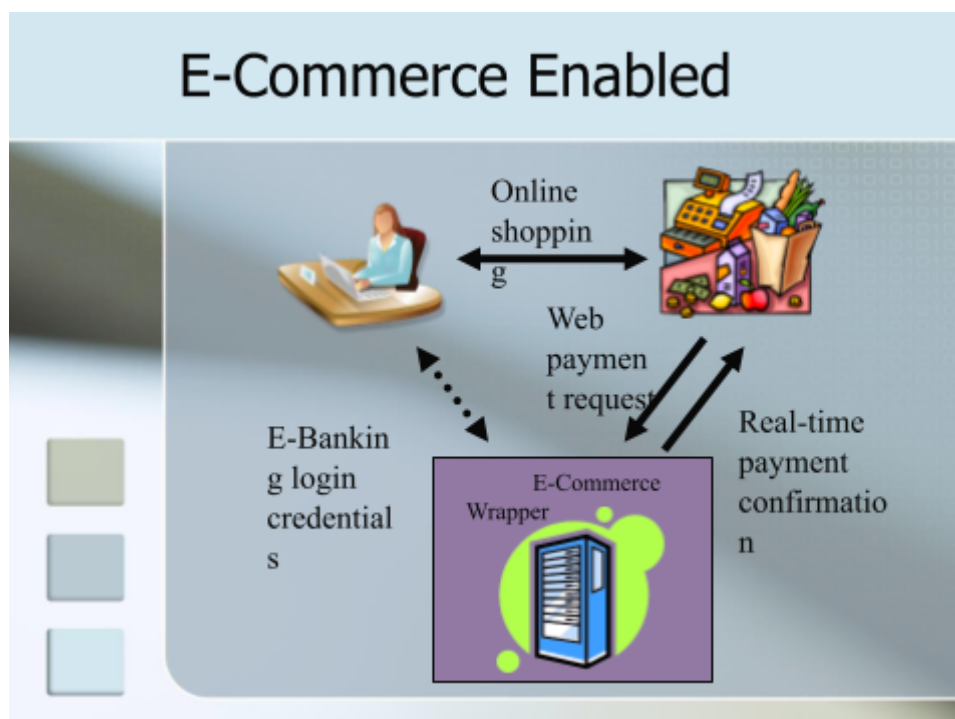


Under this scenario, the bank's system assumes that it is transacting with a live person. It responds to the requests sent by the bank customer over the browser. These requests are made by navigating through the web interface's menu system and by filling up on-screen forms.

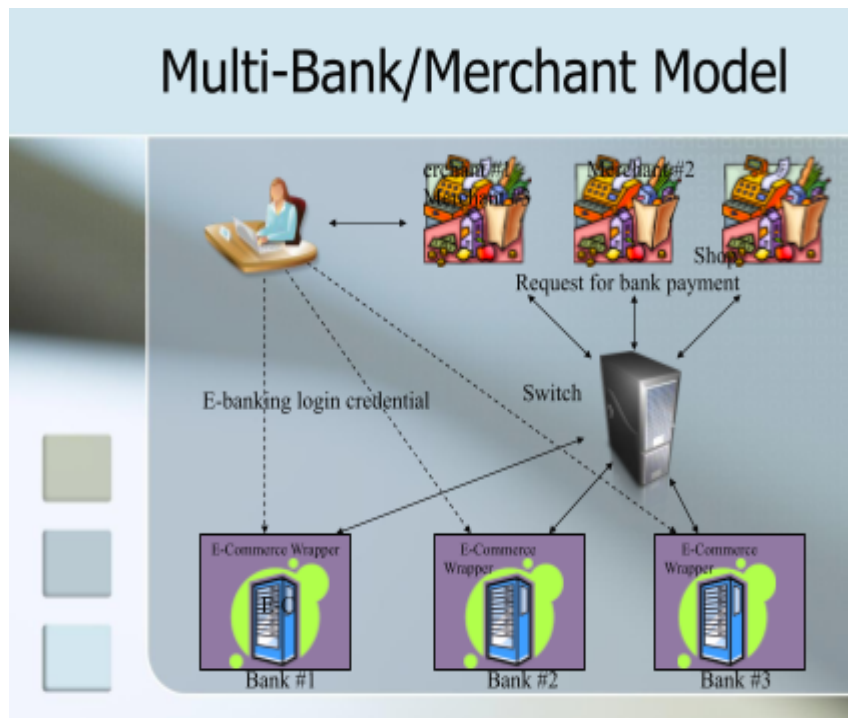
Online banking systems are normally not designed to work with e-commerce merchants or online stores which require machine-to-machine communication. They do not have the capability to accept requests programmatically from 3rd party websites or applications (ex. Shopping cart systems) for debiting the bank account of a particular customer. Subsequently, online banking systems also do not have the capability to communicate with a 3rd party system to inform it if a payment was done successfully or not.

Because of these limitations, it is currently impossible for online merchants to bill customers using their bank accounts in an automated, single-flow process. Merchants normally resort to off-line means such as asking the customer to deposit to their bank account over-the-counter and fax them the deposit slip as proof of payment. This makes it impossible to do e-commerce which require real-time responses (ex. airline ticketing, digital downloads). For merchants with high-volume transactions, the manual validation of deposit slips is also not a scalable solution.

PS seeks to address the problem by providing a "wrapper" interface to the online banking system. This will provide 3rd party online store applications with a programmatic interface to request for payments from the customer's bank, and for the bank to provide real-time feedback or confirmation if the payment was successful or not. In doing so, PS can enable any existing online banking platform to provide e-commerce functionality without or with very little changes, if any.



PS will also perform the role of a traffic cop. It will route the payment request to the appropriate bank chosen by the customer. It will accept payments from the customer in behalf of the merchant, and it will settle with the merchants on a scheduled basis.



M

4.2 How does online bank debit payment work?

All online transactions generally follow the same pattern.

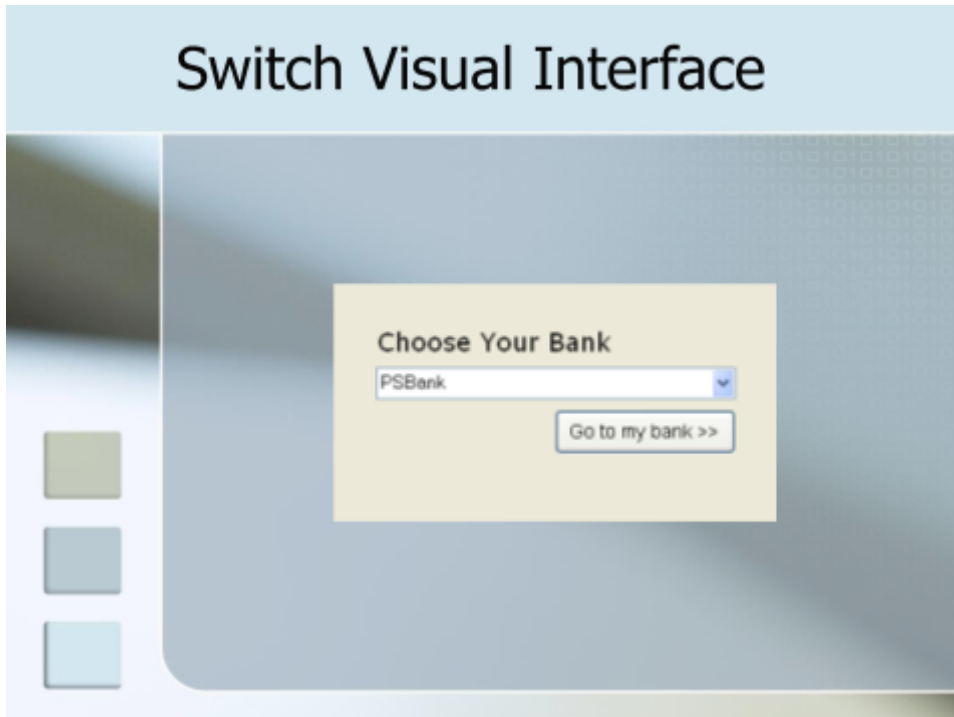
1. Customer surfs an online store
2. Customer clicks on items that he wants
3. Item is placed in an online shopping cart
4. Customer goes to *Checkout*
5. Customer is presented with several payment options
6. Customer clicks on the payment option he prefers
7. Payment processing is performed
8. Online shopping is completed

Where the shopping experience generally vary is in step #7. Different payment options have different process flows. Credit card payments are usually more straightforward – you enter your card details; click a button to confirm; and it's done. Most of the time, the customer does not have to leave the store's *Checkout* page.

With most other payment options (ex. PayPal, BancNet), however, the customer's browser is first redirected to the secure website of the payment processor. From there, he is asked to enter his credentials (ex. PayPal account id and password, BancNet ATM card number and PIN). When all information is entered correctly and

the transaction is confirmed, the customer's browser is redirected back to the online store (step #8) where the shopping is completed.

The PS process flow follows general convention of the other payment options. From the *Checkout* page, the customer is redirected to PS and is presented with a list of banks to choose from.



Customer picks his bank from the list and clicks the button to proceed. PS will then transfer the request to the bank using the API described in this document. At this stage, the bank will generally perform the following operations:

1. Prompt for the necessary credentials (online banking id and password)
2. Let the customer choose from a list of available bank accounts (ex. checking account, savings account)
3. Confirm with customer if he wants to charge the transaction against his chosen account. At this stage, some banks may perform additional authentication (ex. prompting for a transaction password, retrieving confirmation via SMS or email, random number generator)

When payment processing is completed, customer is sent back to the PS using the return API described in this document.

PS keeps track of all payment transaction requests and their statuses. It talks to the bank systems in real-time, as well as, with the merchant shopping systems. It performs the role of the traffic cop and ensures all messages are routed to the appropriate party.

5. Payment Switch API

This section of the document describes the Merchant Payment Switch (PS) API in detail, covering the various functions used, as well as, codes that can be used to integrate them.

5.1 System Requirements

In order to integrate with the PS, Merchant must fulfill the following prerequisites:

1. Merchant website/app must be capable of getting the required data from customers (ex. amount, item description, email)
2. Merchant website/app can send HTTP request data to PS system when a customer wishes to pay the Merchant.
3. Merchant website/app must have a Postback URL to accept real-time confirmation from PS.

Each Merchant is assigned the following:

- merchant id – unique code identifying the Merchant
- collection API key – a unique key assigned to the Merchant for checksum validation.

To authenticate REST/JSON payment requests, PS uses the industry-standard HTTP Basic Authentication method wherein the assigned merchant id and collection API key is Base64-encoded and passed as part of the HTTP request header.

For .NET developers, you can use a code like this to include the basic authentication in the request header:

```
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
request.Credentials = new System.Net.NetworkCredential(merchantId,
collection API key);
```

If you prefer to manually add basic authentication to the headers, the same can be achieved by doing something like this:

```
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
var credentials = System.Text.Encoding.UTF8.GetBytes(merchantId + ":" +
collection API key);
string token =
System.Convert.ToBase64String(credentials);
request.Headers.Add("Content-Type", "application/json");
request.Headers.Add("Authorization", "Basic " + token);
```

Note: collection API key should never be hardcoded in the front-end application where it is viewable in clear text. All API calls should only be made from the merchant's backend servers for security.

5.2 Requesting a Payment

The general flow for making a payment request is as follows:

1. Merchant system requests for a payment to be collected from the PS via JSON/REST, passing all the necessary parameters
2. PS replies with a message containing a status, refno, and url
3. Merchant system performs a browser redirect to the passed url
4. Upon payment completion, PS will call the Merchant system's postback url and redirect the browser back to the Merchant system's return url.

You may use the following URL's as the service entry point for Version 1 (v1) implementation.

Service Production base URL:

`https://gw.dragonpay.ph/api/collect/<version-no>`

Service Test base URL:

`https://test.dragonpay.ph/api/collect/<version-no>`

For merchants migrating from the first generation API, using `<version-no>` of "v1" will provide backward compatibility. The *url* returned will be similar in flow to the original API. For new integrations, we recommend using `<version-no>` of "v2". Thus, the url to be used would normally be:

Service Production base URL:

`https://gw.dragonpay.ph/api/collect/v1`

Service Test base URL:

`https://test.dragonpay.ph/api/collect/v1`

Note that for "v2" the **procId** parameter is required.

5.2.1 Request Parameters

These are the parameters passed by the Merchant via JSON/REST to request for a payment to be collected. Make sure to set the header *Content-Type* to *application/json*.

Endpoint: <baseurl>/<txnid>/post
Method: POST

Note: the *txnid* must be alphanumeric (dashes are allowed) and must not exceed 40- chars in length.

| Parameter | Data Type | Description |
|--------------------------|---------------|---|
| Amount | Numeric(12,2) | The amount to collect (XXXX.XX) |
| Currency | Char(3) | The currency (see Appendix 1) |
| Description | Varchar(128) | A brief description of the request |
| Email | Varchar(40) | Email address of customer |
| MobileNo | Varchar(20) | [OPTIONAL] mobile no of customer |
| ProcId | Varchar(4) | [OPTIONAL] preferred payment channel |
| Param1 | Varchar(80) | [OPTIONAL] value that will be posted back to merchant postback/return url when completed |
| Param2 | Varchar(80) | [OPTIONAL] value that will be posted back to merchant postback/return url when completed [REQUIRED for AGGREGATOR setup] |
| Expiry | DateTime | [OPTIONAL] payment expiry period (best effort) |
| BillingDetails | BillingInfo | [OPTIONAL] billing details of the customer needed for credit card transactions |
| SenderShippingDetails | ShippingInfo | [OPTIONAL] shipping details of the sender needed for COD transactions |
| RecipientShippingDetails | ShippingInfo | [OPTIONAL] shipping details of the recipient needed for COD transactions |
| IpAddress | Varchar(16) | [OPTIONAL] IP address of end-user |
| UserAgent | Varchar(256) | [OPTIONAL] Browser user agent of end-user |

You may keep the *ProcId* blank if you want the user to perform the selection on Dragonpay's side (recommended). If you wish to pre-select the channel, please refer to Section 5.4 of this document on advanced controls.

For credit card transactions, the merchant system must pass the *BillingDetails* field. For Cash on Delivery (COD) transactions, the merchant system must pass the sender and recipient shipping addresses.

The BillingInfo structure is as follows:

| Parameter | Data Type | Description |
|-----------|--------------|--|
| FirstName | Varchar(60) | First name of customer |
| LastName | Varchar(60) | Last name of customer |
| Address1 | Varchar(120) | Street address |
| Address2 | Varchar(120) | Village, subdivision, etc. |
| City | Varchar(40) | City or municipality |
| State | Varchar(40) | State or province |
| Country | Varchar(2) | 2-char ISO country code (ex. PH, US, CA) |
| ZipCode | Varchar(12) | [OPTIONAL] zip code |
| TelNo | Varchar(40) | Telephone number of customer |
| Email | Varchar(40) | Email address of customer |

The ShippingInfo structure is as follows:

| Parameter | Data Type | Description |
|------------|--------------|---|
| FirstName | Varchar(60) | First name of customer |
| MiddleName | Varchar(60) | Middle name of customer |
| LastName | Varchar(60) | Last name of customer |
| Address1 | Varchar(120) | Street address |
| Address2 | Varchar(120) | Village, subdivision, etc. |
| Barangay | Varchar(60) | Barangay |
| City | Varchar(40) | City or municipality |
| Province | Varchar(40) | State or province |
| Country | Varchar(2) | 2-char ISO country code (ex. PH, US, CA) |
| ZipCode | Varchar(12) | [OPTIONAL] zip code |
| Landmark | Varchar() | [OPTIONAL] landmarks or directions to help courier locate the address |
| TelNo | Varchar(40) | Telephone number of customer |
| Email | Varchar(40) | Email address of customer |

At a bare minimum, a sample JSON payment request payload for Metrobankdirect online can look like:

<https://test.dragonpay.ph/api/collect/v2/test20200118001/post>

```
{
  "Amount": "100.00",
  "Currency": "PHP",
  "Description": "Sample transaction",
  "Email": "sample@dev.null.ph",
  "ProcId": "MBTC"
}
```

Below is a sample JSON payload to request collection with GCash:

<https://test.dragonpay.ph/api/collect/v2/test20200118002/post>

```
{
  "Amount": "100.00",
  "Currency": "PHP",
  "Description": "Sample transaction",
  "Email": "sample@dev.null.ph",
  "ProcId": "GCSH"
}
```

Below is a sample JSON payload to request collection with QRPH (ABQR):

<https://test.dragonpay.ph/api/collect/v2/test20200118002/post>

```
{
  "Amount": "100.00",
  "Currency": "PHP",
  "Description": "Sample transaction",
  "Email": "sample@dev.null.ph",
  "ProcId": "ABQR"
}
```

Below is a full sample JSON payload to request collection with credit cards:

<https://test.dragonpay.ph/api/collect/v2/test20200118004/post>

```
{
  "Amount": "100.00",
  "Currency": "PHP",
  "Description": "Sample Transaction",
  "Email": "sample@dragonpay.ph",
  "ProcId": "CC",
  "Param1": "Test parameter 1",
  "Param2": "Test parameter 2",
  "BillingDetails": {
    "FirstName": "Robertson",
    "MiddleName": "Sy", "LastName":
    "Chiang",
    "Address1": "123 Sesame Street", "Address2":
    "Childrens Television Workshop", "City": "Marikina",
    "Province": "Metro Manila", "Country":
    "PH",
    "ZipCode": "1800",
    "TelNo": "86556820",
    "Email": "sample@dragonpay.ph"
  },
  "IpAddress": "123.12.4.67",
  "UserAgent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/48.0.2564.103 Safari/537.36"
}
```

Below is a full sample JSON payload to request collection with COD + Delivery:

<https://test.dragonpay.ph/api/collect/v2/test20200118005/post>

```
{
  "Amount": "100.00",
  "Currency": "PHP",
  "Description": "Sample transaction",
  "Email": "sample@dragonpay.ph",
  "ProcId": "BAE",
  "Param1": "Test parameter 1",
  "Param2": "Test parameter 2",
  "SenderShippingDetails": {
    "FirstName": "Juan",
    "MiddleName": "dela",
    "LastName": "Cruz",
    "Address1": "170 Salcedo Street",
    "Address2": "Legaspi Village",
    "Barangay": "San Lorenzo",
    "City": "Makati",
    "Province": "Metro Manila",
    "Country": "PH",
    "ZipCode": "1229",
    "Landmark": "Near Greenbelt Park",
    "TelNo": "79751111",
    "Email": "operations@companyabc.com"
  },
  "RecipientShippingDetails": {
    "FirstName": "Robertson",
    "MiddleName": "Sy",
    "LastName": "Chiang",
    "Address1": "123 Sesame Street",
    "Address2": "Childrens Television Workshop",
    "Barangay": "San Roque",
    "City": "Marikina",
    "Province": "Metro Manila",
    "Country": "PH",
    "ZipCode": "1800",
    "Landmark": "Near Sta Lucia Mall",
    "TelNo": "86556820",
    "Email": "sample@dragonpay.ph"
  },
  "IpAddress": "123.12.4.67",
  "UserAgent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.103 Safari/537.36"
}
```

5.2.2 Response Parameters

After posting the transaction details via REST, PS replies back with the following JSON message:

| Parameter | Data Type | Description |
|-----------|--------------|--|
| RefNo | Varchar(10) | A unique sequence assigned by PS to this request |
| Status | Char(1) | (S)uccessful or (F)ailed payment initiation request. |
| Message | Varchar(128) | If status is (F)ailed, this may contain additional reason as to why it failed. |
| Url | Varchar(256) | The url that merchant system should redirect the browser to. |

If *Status* returned is (S)uccess, Merchant system should then redirect the customer's browser to the provided *Url* parameter. Take note that *Status* here is not referring to the collection status of the transaction itself, but just of the payment initiation request. So a (S)uccess only means that a payment request has been initiated. It does not mean that the payment has already been collected successfully.

For Version 1 (v1) implementation, the *RefNo* parameter will return a token and not the actual Dragonpay refno. You may just send the browser to *Url* and PS will handle the rest.

For Version 2 (v2) implementation, the *RefNo* parameter is the actual unique reference no of the transaction on the Dragonpay side. If merchant is passing a pre-selected *ProcId* in the collect request in conjunction with the **GetAvailableProcessors** service discussed further in Section 5.4, they must check if that *ProcId* has its *MustRedirect* flag set or not. If it is set, then browser must be redirected to the *Url* next. Otherwise, the *Url* must be just pointing to a regular instruction page that the merchant's UI can hide or display in its desired format.

NOTE: For ABQR, there is an additional parameter in the response, which is "QRPH". You can utilize this parameter by appending the value of the QRPH parameter to <https://gw.dragonpay.ph/Bank/GenerateQRPHCode.aspx?code=> to retrieve the QR code. This QR code can be used in case you want to customize your UI to display only the QR code.

Below is the sample JSON response of *ProcId*=ABQR.

```
{
  "RefNo": "LMRW4THXB4",
  "Status": "S",
  "Message": "Successfully created refno",
  "Url": "https://test.dragonpay.ph/Bank/ProcessAllBank.aspx?procid=ABQR&refno=LMRW4THXB4&amount=100.00&ccy=PHP&description=Deposit&billerId=&email=abc%40gmail.com&digest=e982109e45cebb18d76914149ba49fea86b3ffc7&expiry=3%2f14%2f24+01%3a17&merchantid=TEST&txnid=fpOZoc9A26t0m5wiCQ811U12M1a",
  "QRPH": "00020101021128760011ph.ppml.p2m01110PDVPHM1XXX03157771480000000220416529481372466409105030005204601653036085802PH5909Dragonpay6015City Of Mandalu62280010ph.allbank0510LMRW4THXB488310012ph.ppml.qrph01110PDVPHM1XXX6304F40B"
}
```


5.2.3 Payment Completion Response Parameters

When payment processing has completed, the PS should redirect back the customer's browser to the Merchant's registered callback URL's and pass along the parameters below.

| Parameter | Description |
|-----------|--|
| txnid | A unique id identifying this specific transaction from the merchant side |
| refno | A common reference number identifying this specific transaction from the PS side |
| status | The result of the payment. Refer to Appendix 3 for codes. |
| message | If <i>status</i> is SUCCESS, this should be the PG transaction reference number. If <i>status</i> is FAILURE, return one of the error codes described in Appendix 2. If <i>status</i> is PENDING, the message would be a reference number to complete the funding. |
| amount | The amount processed |
| ccy | The currency of the amount processed |
| procid | The Payment Channel used to process the payment |
| digest | A sha1 checksum digest of the parameters along with the secret key. |

PS recognizes two kinds of callback URL's – the *postback URL* and the *return URL*. The *postback URL* is invoked directly by the PS and does not expect any return value. Because the invocation is directly done by the PS, it is very difficult to fake. The merchant can perform additional source IP address validation to ensure it is the PS making the call. The *postback URL* handler should return with a simple `content-type:text/plain` containing only the single line: `result=OK`.

The *return URL* is passed to the customer's browser via an HTTP redirect. The merchant normally responds with a visual web page reply informing the customer the status of the transaction.

It is not necessary for the merchant to implement both callback URL's, although it is recommended. PS will always invoke the *postback URL* first before the browser redirect to the *return URL*. Thus, the ideal process flow is: upon receiving the *postback URL* call, the merchant's system performs the necessary database updates and initiate whatever back-end process is required. Then when it receives the *return URL* call, it counter-checks the status in the database and provides the visual response. If merchant does not provide both callback URL's, PS will only invoke the one provided.

An HTTP GET from PS to either callback URL's may look something like this:

```
http://www.abcstore.com/Postback.aspx?txnid=1234&refno=5678&status=S&
message=72843747212&digest=a4b3d08462.....
```

The digest is computed using the SHA1 algorithm. Below is a sample code showing how to generate the SHA1 digest using C# .NET:

```
String digest = GetSHA1Digest(String.Format("{0}:{1}:{2}:{3}:{4}",
    Request["txnid"].ToString(),
    Request["refno"].ToString(),
    Request["status"].ToString(),
    Request["message"].ToString(),
    Application["secretkey"].ToString()));
```

Then compare against the passed digest:

```
if (GetSHA1Digest(message) != Request["digest"].ToString())
{
    // display some error message and abort processing
}
else
{
    // if status = 'SUCCESS', process customer order for
    shipment
}
```

In cases wherein the transaction *status* returned is PENDING, the merchant may receive an asynchronous call to the *postback URL* in the future once the funding is completed. The format will just be similar to the HTTP GET callback described above. If a *postback URL* is not defined for the merchant, PS will invoke the *return URL* instead. The merchant should take care in checking the *status* and should only ship goods or render service when *status* value has become SUCCESS.

Special note for Lifetime Id's or Virtual Accounts (VA's), the dynamically generated *txnid* follows the format "<lifetimeid>-yyMMddHHmm" for LID and "<MUVA>-yyMMddHHmm" for MUVA.

5.3 Additional Support Functions

The PS provides some supplementary functions allowing merchants to more tightly integrate and automate their systems.

5.3.1 Transaction Status Inquiry

The merchant can programmatically inquire the status of a transaction by using this function.

5.3.1.1 Request Parameters

These are the parameters passed by the Merchant to the PS via JSON/REST request for a transaction status. The merchant id and collection API key must be passed to the endpoint using standard HTTP Basic Auth username and password.

Endpoint: <baseurl>/refno/<refno>

Method: GET

| Parameter | Data Type | Description |
|-----------|-------------|--|
| refno | Varchar(20) | A unique Dragonpay refno assigned to the specific transaction from the merchant side |

Alternatively, if you do not have the Dragonpay refno, you may use the merchant-assigned transaction id using this endpoint.

Endpoint: <baseurl>/txnid/<txnid>

Method: GET

| Parameter | Data Type | Description |
|-----------|-------------|--|
| txnid | Varchar(40) | A unique id identifying this specific transaction from the merchant side |

5.3.1.2 Response Parameters

The endpoint above returns a JSON-formatted record with the following fields:

| Parameter | Data Type | Description |
|-------------|---------------|--|
| RefNo | Varchar(20) | A unique sequence assigned by PS to this request |
| MerchantId | Varchar(20) | A unique code assigned to Merchant |
| TxnId | Varchar(40) | A unique id identifying this specific transaction from the merchant side |
| RefDate | DateTime | Timestamp when transaction was requested |
| Amount | Numeric(12,2) | The amount to get from the end-user (XXXX.XX) |
| Currency | Char(3) | The currency of the amount (see Appendix 1) |
| Description | Varchar(128) | A brief description of what the payment is for |
| Status | Char(1) | Transaction status (see Appendix 3) |
| Email | Varchar(40) | Email address of customer |
| MobileNo | Varchar(15) | Mobile number of customer |
| ProcId | Varchar(4) | Payment channel selected |
| ProcMsg | Varchar(180) | Messages for transaction specific to procId |
| SettleDate | DateTime | Timestamp when transaction was completed |
| Param1 | Varchar(80) | [OPTIONAL] value that will be posted back to merchant url when completed |
| Param2 | Varchar(80) | [OPTIONAL] value that will be posted back to merchant url when completed |
| Fee | Float | Fee charged by Dragonpay to Merchant |

If the transaction cannot be located, an HTTP Error Status 404 (not found) will be returned.

5.3.2 Cancellation of Transaction

The merchant can programmatically cancel a pending transaction by using this function.

5.3.2.1 Request Parameters

These are the parameters passed by the Merchant to the PS via REST request for voiding of a pending transaction.

Endpoint: <baseurl>/void/<txnid>

Method: GET

| Parameter | Data Type | Description |
|-----------|-------------|--|
| txnid | Varchar(40) | A unique id identifying this specific transaction from the merchant side |

5.3.2.2 Response Parameters

The service will respond with a single *status* string:

| Parameter | Description |
|-----------|--|
| Status | Returns zero (0) if successful, else a negative number |
| Message | Extra description regarding the cancellation request |

If the transaction cannot be located, an HTTP Error Status 404 (not found) will be returned.

5.3.3 Multiple-Use Virtual Accounts and Lifetime Id Functions

The merchant can programmatically manage Lifetime Id's (LID) or Multiple-Use Virtual Accounts (MUVA) by using these functions. LID's are alphanumeric and are generally use with non-bank OTC channels. MUVA's are 16-digit numbers used with Instapay and PESONet. For single-use VA payments, use the standard flow (5.2).

5.3.3.1 Request Parameters for Creation of MUVA or LID

These are the parameters passed by the Merchant to the PS via REST request for a lifetime id.

Endpoint: <https://<env>.dragonpay.ph/api/collect/v1/lifetimeid/create>
Method: POST

| Parameter | Data Type | Description |
|-------------|-------------|--|
| Prefix | Char(2) | The LID prefix assigned to the merchant |
| Bin | Char(6) | [OPTIONAL] provide the assigned 6-digit BIN instead of the 2-char LID prefix for MUVA use |
| Name | Varchar(60) | Name of the customer assigned to this id |
| Email | Varchar(60) | Email of the customer assigned to this id |
| Remarks | Varchar(80) | Additional remarks regarding this id |
| PreferredId | Varchar(10) | [OPTIONAL] Merchant-supplied id. This is a required field for MUVA and must be exactly 10-digits long for MUVA use. (Note: requires coordination with Dragonpay to enable) |

Endpoint: <https://<env>.dragonpay.ph/api/collect/v2/lifetimeid/create>
Method: POST

| Parameter | Data Type | Description |
|-------------|-------------|--|
| Prefix | Char(2) | The LID prefix assigned to the merchant |
| Bin | Char(6) | [OPTIONAL] provide the assigned 6-digit BIN instead of the 2-char LID prefix for MUVA use |
| Name | Varchar(60) | Name of the customer assigned to this id |
| Email | Varchar(60) | Email of the customer assigned to this id |
| Remarks | Varchar(80) | Additional remarks regarding this id |
| PreferredId | Varchar(10) | [OPTIONAL] Merchant-supplied id. This is a required field for MUVA and must be exactly 10-digits long for MUVA use. (Note: requires coordination with Dragonpay to enable) |
| GenerateQR | Boolean | [OPTIONAL] returns a QR string relative to the created MUVA/LID. |

Sample request for regular LID using merchant-assigned prefix with merchant-supplied 6-alphanumeric id. This will return a lifetime id of "TT123456".

```
{
  "Prefix": "TT",
  "Name": "Juan dela Cruz",
  "Email": "juan.dela.cruz@gmail.com",
  "Remarks": "Customer 123456",
  "PreferredId": "123456"
}
```

Sample request of a MUVA using Dragonpay-assigned BIN 705299 and merchant-supplied 10-digit id. This will return a MUVA value of "70529912345890".

```
{
  "Bin": "705299",
  "Name": "Juan dela Cruz",
  "Email": "juan.dela.cruz@gmail.com",
  "Remarks": "Customer 123456",
  "PreferredId": "1234567890"
}
```

By default Static QR is only available in API v2. However, merchants using API v1 can use the endpoint mentioned in section **5.3.3.5 Generating Static QR for Existing MUVA or LID** to generate a Static QR tied to existing LID/MUVA.

```
{
  "Prefix": "TT",
  "Name": "Juan dela Cruz",
  "Email": "juan.dela.cruz@gmail.com",
  "Remarks": "Customer 123456",
  "GenerateQR": true
}
```

Sample payload for merchants using MUVA

```
{
  "Bin": "705299",
  "Name": "Juan dela Cruz",
  "Email": "juan.dela.cruz@gmail.com",
  "Remarks": "Customer 123456",
  "GenerateQR": true
}
```

5.3.3.2 Response Parameters for Creation of MUVA or LID

For API Version 1, The service will respond with a single *lifetime id/VA* string:

| Parameter | Description |
|-----------------|---|
| LifetimeId/MUVA | Returns generated LID or MUVA, or empty string if failed. |

For API Version 2 the service will return the following parameters.

| Parameter | Description |
|------------|--|
| LifetimeId | The generated LID, returns null if failed to generate or using MUVA. |
| MUVA | The generated MUVA, returns null if failed to generate or using LID. |
| QRPH | String representation of the generated QR. Returns empty if failed or the GeneratedQR parameter is set to false. |

5.3.3.3 Request Parameters for Activation/Deactivation of MUVA or LID

These are the parameters passed by the Merchant to the PS via REST request for a lifetime id. Replace <verb> with either *activate* or *deactivate*.

Endpoint:<baseurl>/lifetimeid/<verb>/<id>

Method: GET

| Parameter | Data Type | Description |
|-----------|-------------|---|
| Id | Varchar(16) | The LID or MUVA to activate or deactivate |

5.3.3.4 Response Parameters for Activation/Deactivation of MUVA or LID

The service will respond with an http status 200 if successful else status 404.

5.3.3.5 Generating Static QR for Existing MUVA or LID

Generate a static QRPH tied to the provided existing LID.

NOTE: For merchants who wanted to utilize static QR for existing LIDs. A way for v1 merchants to use static QR.

Endpoint:<baseurl>/lifetimeid/<id>/generate-static-qr

Method: POST

| Parameter | Data Type | Description |
|-----------|-------------|---|
| Id | Varchar(16) | The LID or MUVA to generate a static QR for |

5.3.3.6 Request Parameters for Retrieving MUVA or LID

These are the parameters passed by the Merchant to the PS via REST request for a lifetime id.

Endpoint:<baseurl>/lifetimeid/<id>

Method: GET

| Parameter | Data Type | Description |
|-----------|-------------|-----------------------------|
| Id | Varchar(16) | The LID or MUVA to retrieve |

5.3.3.7 Response Parameters for Retrieving MUVA or LID

The service will respond with the following record:

| Parameter | Data Type | Description |
|------------|--------------|--|
| userId | Varchar(12) | The assigned LID/MUVA |
| merchantId | Varchar(20) | The merchant id who owns the lifetime id |
| custName | Varchar(60) | The registered name of the customer using the id |
| custEmail | Varchar(60) | The registered email address of the customer using the id |
| remarks | Varchar(80) | Additional remarks regarding the customer using the id |
| status | Char(1) | (A)ctive or (I)nactive |
| created | Timestamp | Date and time when the id was created |
| staticQr | VARCHAR(512) | The generated static QR string tied to the LID/MUVA provided. Returns null if QR is not yet generated. |

5.3.3.7 Updating Lifetime Users for MUVA and LID

These are the parameters passed by the Merchant to the PS via REST request for a lifetime id and Va number.

Endpoint: <baseurl>/lifetimeid/<LID or VA>

Method: PATCH

| Parameter | Data Type | Description |
|------------------|-------------|---|
| Lifetime id / VA | Varchar(12) | The assigned Lifetime id / VA |
| Name | Varchar(60) | The registered name of the customer using the Lifetime id / VA |
| Email | Varchar(60) | The registered email address of the customer using the Lifetime id / VA |
| remarks | Varchar(80) | Additional remarks regarding the customer using the Lifetime id / VA |

Sample request for regular LID using merchant-assigned prefix with merchant-supplied 6-alphanumeric id. This will return a lifetime id of "TT123456".

```
{
  "Name": "Juan dela Cruz",
  "Email":
    "juan.dela.cruz@gmail.com",
  "Remarks": "Customer 123456"
}
```

Sample request of a MUVA using Dragonpay-assigned BIN 705299 and merchant-supplied 10-digit id. This will return a MUVA value of "70529912345890".

```
{
    "Name": "Juan dela Cruz",
    "Email":
    "juan.dela.cruz@gmail.com",
    "Remarks": "Customer 123456"
}
```

5.3.3.8 Response parameters for Updating Lifetime Users for MUVA and LID

The server will respond with an HTTP Code 200 if successful, and no response body will be thrown, else, HTTP Error (e.g 404, etc.) will be thrown, and a brief message regarding the error.

| Parameter | Description |
|-----------|--|
| Message | Extra description for updating request |

5.3.4 Transaction History Functions

The merchant can programmatically retrieve a list of transactions

5.3.4.1 Request for Transaction History

To get the list of all transactions between a date/time range via REST, you may call this endpoint.

Endpoint: `<baseurl>/transactions?startdate={start}&enddate={end}`

Method: GET

Both `{start}` and `{end}` may use the format `yyyy-MM-dd` to indicate a date range. If you need to indicate the time as well, use the format `yyyy-MM-ddThh:mm:ss`. Since this is an HTTP GET parameter, make sure to url-encode the colons (":") to %3A. If no specific hour is indicated with the `{end}` parameter, it is assumed to be up to 23:59:59.999 of that date.

5.3.4.2 Response for Transaction History

The system will respond with an array of records containing the details of all transactions within the period.

5.3.4.3 Request for History of Successfully Completed Transactions

To get the list of all transactions between a date/time range via REST, you may call this endpoint.

Endpoint:

`<baseurl>/transactions/settled?startdate={start}&enddate={end}`

Method: GET

See note on 5.3.4.1 for date/time format.

5.3.4.4 Response for History of Successfully Completed Transactions

The system will respond with an array of records containing the details of all transactions within the period.

5.4 Customization of Payment Selection

There may be instances wherein the merchant would want to filter the payment channels that they want to appear in Dragonpay's payment selection page, or they may want to skip the Dragonpay page altogether and go straight to the payment details for a specific channel. There is support for these features and this section discusses them in detail.

There are two general forms of customization:

1. Simple control of what payment options appear in Dragonpay's dropdown list
2. Moving the payment selection process to the merchant side and calling Dragonpay in the background

5.4.1 Simple Control

With the simple method, the process flow is still essentially the same – merchant redirects the page to Dragonpay for the buyer to make the payment selection. However, merchant can control to a certain degree which options appear in the payment selection list, or merchant can make a pre-selection to a specific channel.

5.4.1.1 Filtering Payment Channels

Dragonpay payment channels are grouped together by type – ex. Online banking, Over-the-Counter/ATM, etc. Merchants can programmatically instruct Dragonpay which grouping to show when the user is redirected to the payment gateway by using the "mode" parameter.

| Mode Value | Grouping Type |
|------------|----------------------------------|
| 1 | Online Banking |
| 2 | Over-the-Counter Banking and ATM |
| 4 | Over-the-Counter non-Bank |
| 8 | E-Wallets (inc. Bitcoins) |
| 16 | (reserved internally) |
| 32 | PayPal |
| 64 | Credit Cards |
| 128 | Mobile (Gcash) |
| 256 | International OTC |
| 512 | Bancnet |
| 1024 | Auto Debit Arrangement (ADA) |
| 2048 | Cash on Delivery (COD) |
| 4096 | Installments |

"Mode" is a bitmask which can be OR-ed to achieve the result intended. The following example will only show the online banking options:

```
https://gw.dragonpay.ph/Pay.aspx?merchantid=ABC&txnid=1234&...&mode=1
```

Merchants who avail of PayPal or GCash from Dragonpay but do not want them to appear in the dropdown list, may specify a "mode=7" to display only the basic alternative payments in the dropdown list.

5.4.1.2 Pre-selecting Payment Channels

Dragonpay has basic support to allow merchants to go directly to a payment channel without having to select it from the dropdown list. The following are sample processor id's which can be used to go straight to the selection:

| Proc Id | Name |
|---------|---------------------------|
| BDO | BDO Online Banking |
| CC | Credit Cards |
| CEBL | Cebuana Lhuillier |
| DPAY | Dragonpay Prepaid Credits |
| ECPY | ECPay |
| GCSH | Gcash |
| PYPL | PayPal |
| MLH | M. Lhuillier |
| RDS | Robinsons Dept Store |
| SMR | SM Payment Counters |
| 711 | 7-Eleven (if applicable) |
| INPY | Instapay (if applicable) |

Merchants who want to receive Gcash or PayPal payments may put separate radio buttons at their checkout page to give user the capability to go straight to that channel without stopping by the Dragonpay payment selection page by passing a "procid" parameter.

The following example will direct the buyer to our Gcash payment page from the merchant's checkout page:

```
https://gw.dragonpay.ph/Pay.aspx?merchantid=ABC&txnid=1234&...&procid=GCSH
```

For PayPal and credit card acceptance, Merchant is required to apply for a separate acquirer's id with the respective payment gateways. Contact our Sales for assistance.

5.4.2 Advanced Control

If the merchant wishes to keep the payment user experience as close to their checkout page as possible, Dragonpay provides support to perform this to a certain extent.

The general process flow is as follows:

1. Call the *GetAvailableProcessors* web service to retrieve a list of all supported payment channel.
2. Merchant dynamically render its checkout page depending on the result set of *GetAvailableProcessors*.
3. Merchant redirects the browser to the Payment Url passing the selected processor id (*procid*) as parameter.
4. When payment is completed, Dragonpay invokes the Merchant's Postback / Return Url's.

5.4.2.1 Determining Available Payment Channels

Depending on various factors, some payment channels may not be available at all times. Merchants who implement the techniques mentioned in the previous section to perform payment channel selection at their checkout page, and use the *procid* parameter to send the user directly to the payment channel instruction, have to be careful not to send the user to an inactive channel.

Merchant can query Dragonpay for the available payment channels at a particular point in time by using the REST Web Service *GetAvailableProcessors*.

Merchants are strongly discouraged from statically listing Dragonpay payment channels as there are various rules that determine their availability. These include:

1. Some processors are only available on certain days of the week (ex. Not available on weekends or non-banking days).
2. Some processors are only available between certain times of the day (ex. Goes down nightly for maintenance).
3. Some processors have limits on the minimum or maximum amount that can be processed through them.
4. Scheduled or unscheduled system maintenance.

For these reasons, Merchants who want to customize the user experience by moving the payment selection onto their checkout page have to be aware of all these rules. Otherwise, customers may encounter problems.

5.4.2.1.1 GetAvailableProcessors Request Parameters

To retrieve the list of all processor channels, you can use this:

Endpoint: <baseurl>/processors

To retrieve the list of available processor channels for a particular amount, you can use this:

Endpoint: <baseurl>/processors/available/<amount>

| Parameter | Data Type | Description |
|-----------|---------------|-------------------------------|
| amount | Numeric(12,2) | The amount of the transaction |

If an **amount** value greater than zero is passed, *GetAvailableProcessors* will return a list of channels available for that amount. But if you want to retrieve the full list regardless of the amount so you can cache it locally and avoid having to calling the web method for each transaction, you can set **amount** to a special value of -1000.

5.4.2.1.2 GetAvailableProcessors Response Parameters

The web service will return an array of records in a REST/JSON or XML/SOAP envelope format. Each record contains the following fields:

| Parameter | Data Type | Description |
|------------|--------------|---|
| procId | Varchar(4) | A unique code assigned to this processor |
| shortName | Varchar(15) | A brief name for this processor. Can be used if UI space is limited. |
| longName | Varchar(40) | A longer, more descriptive name of the processor. Can be used if UI space allows. |
| logo | Varchar(160) | A url pointing to the logo of this procid that Dragonpay uses |
| currencies | Varchar(80) | A comma-delimited list of currencies that this procid can support. |
| type | Integer | Bitmask. Refer to Section 5.4.1.1 for various meanings |
| status | Char(1) | Can be (A)ctive or (I)nactive. As of this writing, GetAvailableProcessors only return (A)ctive procid's. |
| remarks | Varchar(320) | This string may be displayed by merchant in its checkout page to give user more details or descriptions about what this procid is about. |
| dayOfWeek | Char(7) | A string mask corresponding to the 7 days of the week starting from Sunday and ending Saturday. If an "X" is in the mask position, that means the procid is available on that day; else, it is unavailable and should not be displayed. |
| startTime | Char(5) | Starting time when this procid is available to process (in 24-hr "HH:MM" format) |
| endTime | Char(5) | Ending time after which this procId is no longer |

| | | |
|--|--|-----------------------|
| | | available to process. |
|--|--|-----------------------|

| | | |
|--------------|---------------|---|
| minAmount | Numeric(12,2) | The smallest amount this procid can process. |
| maxAmount | Numeric(12,2) | The amount over-and-above which procid is not allowed to process. |
| mustRedirect | Bool | This flag tells the Merchant whether a browser redirect is mandatory. |
| surcharge | Numeric(12,2) | The amount added for payments using this channel |
| hasAltRefNo | Bool | Has a 10-digit alternate refno used when paying |

Additional Notes:

1. If **dayOfWeek** is "0XXXXX0", for example, that means it is not available on Sundays and Saturdays, but is available from Monday to Friday.
2. It is strongly recommended that Merchant uses the **remarks** field to display tips or additional description when the channel is selected. This field will also inform the user if there are any surcharges that may be applied for using this channel.
3. If **startTime=endTime**, then this procId is available 24-hrs a day. If **endTime** is "00:00", but **startTime** is not "00:00", then **endTime** should be interpreted as the stroke of midnight.
4. The **minAmount** and **maxAmount** fields should be implemented as follows –
if (amount >= minAmount && amount < maxAmount) then proceed with this channel, else do not show this channel. That is **not** amount <= maxAmount.
5. It is recommended that the *GetAvailableProcessors* web service be invoked by a scheduled cron job every 30 mins to every hour with **amount** = -1000. While the field values generally will not change, the status can change during the day for various reasons. For example, a bank partner may have an unscheduled downtime. If Merchant does not refresh its internal copy of this list, it may think the channel is still active whereas it has already been deactivated temporarily (or permanently) on Dragonpay's side.

Appendix 1 – Currency Codes

| Code | Description |
|------|-----------------|
| PHP | Philippine Peso |
| USD | US Dollar |

Appendix 2 – Error Codes

| Code | Description |
|------|----------------------------|
| 000 | Success |
| 101 | Invalid payment gateway id |
| 102 | Incorrect secret key |
| 103 | Invalid reference number |
| 104 | Unauthorized access |
| 105 | Invalid token |
| 106 | Currency not supported |
| 107 | Transaction cancelled |
| 108 | Insufficient funds |
| 109 | Transaction limit exceeded |
| 110 | Error in operation |
| 111 | Security Error |
| 112 | Invalid parameters |
| 201 | Invalid Merchant Id |
| 202 | Invalid Merchant Password |

Appendix 3 – Status Codes

| Code | Description |
|------|-------------|
| S | Success |
| F | Failure |
| P | Pending |
| U | Unknown |
| V | Void |