Dragonpay

Release v3.2.9

Jeff Claud

Jul 27, 2023

CONTENTS

This library will help you integrate your application with Dragonpay payment gateway.

Contents:

CHAPTER

ONE

OVERVIEW

1.1 Changed Log

v3.2.8

- Fixed composer 2 deprecation notice
- Update Web Service Production URL

v3.2.9

- Refactored codebase
- Allowed changing payment url, web service url and send billing info url programmatically
- Can now get all available processors.

1.2 Requirements

1. PHP >= 7.1
2. SoapClient

1.3 Installation

```
composer require crazymeeks/dragonpay v3.2.9
```

Alternatively, you can specify DragonPay as a dependency in your project's existing composer.json file:

```
 {
   "require": {
      "crazymeeks/dragonpay": "^v3.2.9"
   }
}
```

CHAPTER

TWO

QUICKSTART

This page provide a quick introduction on how to use Dragonpay library. If you have not already installed Dragonpay, go check the *Installation* page

2.1 Making your first payment

```
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Dragonpay;

class ExampleClass
{

    public function postCheckout()
    {
        $parameters = [
            'txnid' => 'TXNID', # Varchar(40) A unique id identifying this specific␣
↪transaction from the merchant site
            'amount' => 1, # Numeric(12,2) The amount to get from the end-user (XXXX.XX)
            'ccy' => 'PHP', # Char(3) The currency of the amount
            'description' => 'Test', # Varchar(128) A brief description of what the␣
↪payment is for
            'email' => 'some@merchant.ph', # Varchar(40) email address of customer
            'param1' => 'param1', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed
            'param2' => 'param2', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed

        ];


        $merchant_account = [
            'merchantid' => 'MERCHANTID',
            'password'   => 'MERCHANT_KEY'
        ];
        // Initialize Dragonpay
        $dragonpay = new Dragonpay($merchant_account);
        // Set parameters, then redirect to dragonpay
        $dragonpay->setParameters($parameters)->away();
```

(continues on next page)

(continued from previous page)

```
    }
}
```

2.2 SOAP/XML Web Service Model(Recommended)

For **GREATER SECURITY**, you can use the API using XML Web Service Model. Under this model, the parameters are not passed through browser redirect which are visible to end-users. Instead parameters are exchanged directly between the Merchant site and Payment Switch servers through SOAP calls. The PS will return a token which you will be used to redirect to PS. Just make sure you have `SoapClient` enabled/installed on your system and call `getToken()` method.

```
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Dragonpay;
use Crazymeeks\Foundation\PaymentGateway\Dragonpay\Token;

class ExampleClass
{

    public function postCheckout()
    {
        $parameters = [
            'txnid' => 'TXNID', # Varchar(40) A unique id identifying this specific␣
↪transaction from the merchant site
            'amount' => 1, # Numeric(12,2) The amount to get from the end-user (XXXX.XX)
            'ccy' => 'PHP', # Char(3) The currency of the amount
            'description' => 'Test', # Varchar(128) A brief description of what the␣
↪payment is for
            'email' => 'some@merchant.ph', # Varchar(40) email address of customer
            'param1' => 'param1', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed
            'param2' => 'param2', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed

        ];

        $merchant_account = [
            'merchantid' => 'MERCHANTID',
            'password'   => 'MERCHANT_KEY'
        ];
        // Initialize Dragonpay
        $dragonpay = new Dragonpay($merchant_account);
        // Get token from Dragonpay
        $token = $dragonpay->getToken($parameters);
        // If $token instance of Crazymeeks\Foundation\PaymentGateway\Dragonpay\Token,␣
↪then proceed
        if ( $token instanceof Token ) {
            $dragonpay->away();
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

2.3 Using Credit Card

To use credit card payment, please make sure you have SoapClient installed/enabled on your system and make call to `useCreditCard($parameters)` method. This method will throw `Crazymeeks\Foundation\Exceptions\SendBillingInfoException` when error occurred. **Note:** credit card is only available in production.

```
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Dragonpay;
use Crazymeeks\Foundation\PaymentGateway\Dragonpay\Token;

class ExampleClass
{

    public function postCheckout()
    {
        $parameters = [
            'txnid' => 'TXNID', # Varchar(40) A unique id identifying this specific␣
↪transaction from the merchant site
            'amount' => 1, # Numeric(12,2) The amount to get from the end-user (XXXX.XX)
            'ccy' => 'PHP', # Char(3) The currency of the amount
            'description' => 'Test', # Varchar(128) A brief description of what the␣
↪payment is for
            'email' => 'some@merchant.ph', # Varchar(40) email address of customer
            'param1' => 'param1', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed
            'param2' => 'param2', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed

            'firstName' => 'John',
            'lastName'  => 'Doe',
            'address1'  => '#123 Chocolate Hills',
            'address2'  => 'Sweet Mountain',
            'city'      => 'Hillside',
            'state'     => 'Bohol',
            'country'   => 'PH',
            'zipCode'   => '1201',
            'telNo'     => '63 2029',
        ];

        $merchant_account = [
            'merchantid' => 'MERCHANTID',
            'password'   => 'MERCHANT_KEY'
        ];
```

(continues on next page)

(continued from previous page)

```
        $testing = false; # Set Payment mode to production
        // Initialize Dragonpay
        $dragonpay = new Dragonpay($merchant_account, $testing);
        $dragonpay->useCreditCard($parameters)->away();

        # If you want to use SOAP, just chain call
        # getToken($parameters) method like below
        # $dragonpay->useCreditCard($parameters)->getToken($parameters)->away();

    }
}
```

If you want to use token(recommended), you can do it using below code:

```
$dragonpay->useCreditCard($parameters)->getToken($parameters)->away();
```

CHAPTER

THREE

FILTERING PAYMENT CHANNELS

Available payment channels

`Dragonpay::ONLINE_BANK` `Dragonpay::OTC_BANK` `Dragonpay::OTC_NON_BANK`
`Dragonpay::PAYPAL` `Dragonpay::GCASH` `Dragonpay::INTL_OTC`

Payment Channels are grouped together by type. E.g `Online Banking`, `Over-the-Counter/ATM`, etc. You can set payment channel by calling `filterPaymentChannel()` method and pass one of the available payment channels above.

```php
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Dragonpay;

class ExampleClass
{

    public function postCheckout()
    {
        $parameters = [
            'txnid' => 'TXNID', # Varchar(40) A unique id identifying this specific␣
↪transaction from the merchant site
            'amount' => 1, # Numeric(12,2) The amount to get from the end-user (XXXX.XX)
            'ccy' => 'PHP', # Char(3) The currency of the amount
            'description' => 'Test', # Varchar(128) A brief description of what the␣
↪payment is for
            'email' => 'some@merchant.ph', # Varchar(40) email address of customer
            'param1' => 'param1', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed
            'param2' => 'param2', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed

        ];

        $merchant_account = [
            'merchantid' => 'MERCHANTID',
            'password'   => 'MERCHANT_KEY'
        ];
        // Initialize Dragonpay
        $dragonpay = new Dragonpay($merchant_account);
        // Filter payment channel
        $dragonpay->filterPaymentChannel( Dragonpay::ONLINE_BANK );
        // Set parameters, then redirect to dragonpay
```

(continues on next page)

(continued from previous page)

```
        $dragonpay->setParameters($parameters)->away();

    }
}
```

3.1 Pre-selecting Payment Channels

If you want to go directly to a payment channel without having to select from the dropdown list and without stopping by the Dragonpay selection page, you can chain call the `withProcid($procid)` method. This method will throw `Crazymeeks\Foundation\Exceptions\InvalidProcessIdException` when processor id is not supported.

Available Processors:

`Processor::CREDIT_CARD` `Processor::GCASH` `Processor::PAYPAL` `Processor::BAYADCENTER` `Processor::BITCOIN` `Processor::CEBUANA_LHUILLIER` `Processor::CHINA_UNIONPAY` `Processor::DRAGONPAY_PREPARED_CREDITS` `Processor::ECPAY` `Processor::LBC` `Processor::MLHUILLIER` `Processor::ROBINSONS_DEPT_STORE` `Processor::SM_PAYMENT_COUNTERS`

```
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Dragonpay;
use Crazymeeks\Foundation\PaymentGateway\Options\Processor;

class ExampleClass
{

    public function postCheckout()
    {
        $parameters = [
            'txnid' => 'TXNID', # Varchar(40) A unique id identifying this specific␣
↪transaction from the merchant site
            'amount' => 1, # Numeric(12,2) The amount to get from the end-user (XXXX.XX)
            'ccy' => 'PHP', # Char(3) The currency of the amount
            'description' => 'Test', # Varchar(128) A brief description of what the␣
↪payment is for
            'email' => 'some@merchant.ph', # Varchar(40) email address of customer
            'param1' => 'param1', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed
            'param2' => 'param2', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed

        ];

        $merchant_account = [
            'merchantid' => 'MERCHANTID',
            'password'   => 'MERCHANT_KEY'
        ];
        // Initialize Dragonpay
        $dragonpay = new Dragonpay($merchant_account);
        // Set parameters, then redirect to dragonpay
        $dragonpay->setParameters($parameters)
```

(continues on next page)

(continued from previous page)

```
                ->withProcid(Processor::CREDIT_CARD)
                ->away();
    }
}
```

Or if you prefer using SOAP/XML web service

```
$token = $dragonpay->getToken($parameters);
if ( $token instanceof \Crazymeeks\Foundation\PaymentGateway\Dragonpay\Token ) {
    // use procid
    $dragonpay->withProcid(Processor::CREDIT_CARD)->away();
}
```

3.2 Payment Mode

By default, the payment mode of this library is sandbox. To change this to production, just pass boolean `false` to second parameter of Constructor of `Crazymeeks\Foundation\PaymentGateway\Dragonpay`.

```
$merchant_account = [
    'merchantid' => 'MERCHANTID',
    'password'   => 'MERCHANT_KEY'
];
$testing = false;
// Initialize Dragonpay
$dragonpay = new Dragonpay($merchant_account, $testing);
```

3.3 Exceptions

You can wrap your code in a `try{}catch(){}` and use `Crazymeeks\Foundation\Exceptions\PaymentException` so you can catch error and see error message safely when something went wrong.

```
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Dragonpay;
use Crazymeeks\Foundation\Exceptions\PaymentException;

class ExampleClass
{

    public function postCheckout()
    {
        $parameters = [
            'txnid' => 'TXNID', # Varchar(40) A unique id identifying this specific␣
↪transaction from the merchant site
            'amount' => 1, # Numeric(12,2) The amount to get from the end-user (XXXX.XX)
            'ccy' => 'PHP', # Char(3) The currency of the amount
            'description' => 'Test', # Varchar(128) A brief description of what the␣
↪payment is for
```

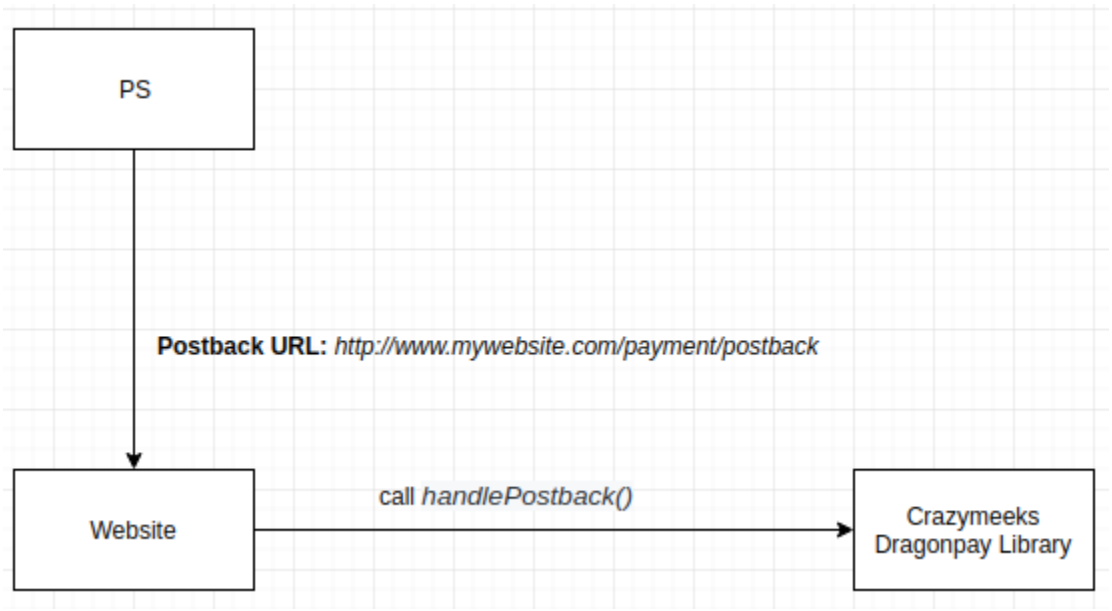
(continues on next page)

(continued from previous page)

```
            'email' => 'some@merchant.ph', # Varchar(40) email address of customer
            'param1' => 'param1', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed
            'param2' => 'param2', # Varchar(80) [OPTIONAL] value that will be posted␣
↪back to the merchant url when completed

        ];

        $merchant_account = [
            'merchantid' => 'MERCHANTID',
            'password'   => 'MERCHANT_KEY'
        ];

        $dragonpay = new Dragonpay($merchant_account);
        // Set parameters, then redirect to dragonpay
        try {
            $dragonpay->setParameters($parameters)->away();
        } catch(PaymentException $e){
            echo $e->getMessage();
        } catch(\Exception $e){
            echo $e->getMessage();
        }

    }
}
```

3.4 Postback handler

According to DP's official documentation, **postback URL** is invoked directly by the PS and does not expect any return value. PS will 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. **Please keep in mind the HTTP method of your postback URL should be POST($_POST) not GET($_GET).**



This library provides simple feature for this out of the box so you can handle data when PS invoked your _postback URL._ Just call *handlePostback()* method. *handlePostback()* will return the following array so you can do whatever you want to this returned data:

```
array(
    'txnid' => '109019',
    'refno' => '0398739',
    'status' => 'S',
    'message' => 'loioeiu8398!)()39483',
    'digest'  => '0oi30430aoi!)04490',
    'description' => 'Success'
)
```

3.5 Usage

Using closure/anonymous function:

```
$merchant_account = [
    'merchantid' => 'MERCHANTID',
    'password'   => 'MERCHANT_KEY'
];
$dragonpay = new Dragonpay($merchant_account);
$dragonpay->handlePostback(function($data){
    // do your stuff here like save data to your database.
    $insert = "Insert INTO mytable(`txnid`, `refno`, `status`) VALUES ($data['txnid'],
↪$data['refno'])";
    mysql_query($insert);

    # or if you are in Laravel, you can use Model or DB Facade...
    // DB::table('mytable')->insert($data);

}, $_POST);
```

Or if you are using Laravel framework, use `$request->all()` or `$request->toArray()` instead of $_POST.

```
$dragonpay->handlePostback(function($data){
    // do your stuff here like save data to your database.
    $insert = "Insert INTO mytable(`txnid`, `refno`, `status`) VALUES ($data['txnid'],
↪$data['refno'])";
    mysql_query($insert);

    # or if you are in Laravel, you can use Model or DB Facade...
    // DB::table('mytable')->insert($data);

}, $request->all());
```

Or you can also create your own class that implements `Crazymeeks\Foundation\PaymentGateway\Handler\PostbackHandlerInterface`

```
namespace YourNameSpace;

use Crazymeeks\Foundation\PaymentGateway\Handler\PostbackHandlerInterface;

class MyPostBackHandler implements PostbackHandlerInterface
{
    public function handle(array $data)
    {
        // do your stuff here like save data to your database.
        $insert = "Insert INTO mytable(`txnid`, `refno`, `status`) VALUES ($data['txnid'],
↪ $data['refno'])";
        mysql_query($insert);

        # or if you are in Laravel, you can use Model or DB Facade...
        // DB::table('mytable')->insert($data);
    }
}
$merchant_account = [
'merchantid' => 'MERCHANTID',
'password'   => 'MERCHANT_KEY'
];
$dragonpay = new Dragonpay($merchant_account);
$dragonpay->handlePostback(new MyPostBackHandler(), $_POST);
# If you are in Laravel, use $request->all() or $request->toArray() instead of $_POST.
# $dragonpay->handlePostback(new MyPostBackHandler(), $request->all());
```

3.6 Cancellation of Transaction

To cancel a transaction, just call `action()` method and pass object of `Crazymeeks\Foundation\PaymentGateway\Dragonpay\Action\CancelTransaction` with transaction id as constructor parameter. `action()` method will throw `Crazymeeks\Foundation\Exceptions\Action\CancelTransactionException` when error occured.

```
$merchant_account = [
   'merchantid' => 'MERCHANTID',
   'password'   => 'MERCHANT_KEY'
];
```

(continues on next page)

(continued from previous page)

```
$txnid = 'SAMPLE-TXNID-10910';
$dragonpay = new Dragonpay($merchant_account);
try{
    $dragonpay->action(new \Crazymeeks\Foundation\PaymentGateway\Dragonpay\Action\
↪CancelTransaction($txnid));
}catch(\Crazymeeks\Foundation\Exceptions\Action\CancelTransactionException $e){
    // Error transaction cancellation
}
```

3.7 Transaction Status Inquiry

If you want to check transaction status, just call `action()` method of pass object of `Crazymeeks\Foundation\PaymentGateway\Dragonpay\Action\CheckTransactionStatus`. You may pass either `txnid` or `refno` in the constructor of this class.

```
$merchant_account = [
    'merchantid' => 'MERCHANTID',
    'password'   => 'MERCHANT_KEY'
];
$txnid = 'SAMPLE-TXNID-10910';

$dragonpay = new Dragonpay($merchant_account);
$status = $dragonpay->action(new \Crazymeeks\Foundation\PaymentGateway\Dragonpay\Action\
↪CheckTransactionStatus($txnid));
```

3.8 Transaction Status Inquiry Response

```
stdClass Object
(
    [RefNo] => XMNUQ7M9W5
    [MerchantId] => MERCHANTID
    [TxnId] => TXNID-145076875
    [RefDate] => 2022-05-19T16:37:11.915
    [Amount] => 1
    [Currency] => PHP
    [Description] => Test Description
    [Status] => S
    [Email] => some@merchant.ph
    [MobileNo] =>
    [ProcId] => BOG
    [ProcMsg] => [000] BOG Reference No: 20220519163731
    [SettleDate] => 2022-05-19T16:37:31.76
    [Param1] => param1
    [Param2] => param2
    [Fee] => 0
)
```

3.9 Advanced Control

Please read Dragonpay documentation then read through 5.4.2 Advanced Control

```
$merchant_account = [
    'merchantid' => 'MERCHANTID',
    'password'   => 'MERCHANT_KEY'
];
$dragonpay = new Dragonpay($merchant_account);
$amount = Dragonpay::ALL_PROCESSORS;
$processors = $dragonpay->getPaymentChannels($amount);
```

Response

```
Array
(
    [0] => stdClass Object
        (
            [procId] => BDO
            [shortName] => BDO
            [longName] => BDO Internet Banking
            [logo] => ~/images/bdologo.jpg
            [currencies] => PHP
            [url] =>
            [realTime] => 1
            [pwd] =>
            [defaultBillerId] =>
            [hasTxnPwd] =>
            [hasManualEnrollment] => 1
            [type] => 1
            [status] => A
            [remarks] => Use your BDO Retail Internet Banking (RIB) account to make a␣
↪payment. Read our <a href='http://www.dragonpay.ph/bdorib-how-to' target='_blank'>BDO␣
↪RIB guide</a> for more details.
            [dayOfWeek] => XXXXXXX
            [startTime] => 06:00
            [endTime] => 21:30
            [minAmount] => 1
            [maxAmount] => 1000000
            [mustRedirect] =>
            [surcharge] => 0
            [hasAltRefNo] =>
            [cost] => 0
        )

    [1] => stdClass Object
        (
            [procId] => BDOA
            [shortName] => BDO ATM
            [longName] => Banco de Oro ATM
            [logo] => ~/images/bdologo.jpg
            [currencies] => PHP
            [url] =>
```

(continues on next page)

(continued from previous page)

```
            [realTime] =>
            [pwd] =>
            [defaultBillerId] =>
            [hasTxnPwd] =>
            [hasManualEnrollment] =>
            [type] => 2
            [status] => A
            [remarks] => Pay at any BDO ATM nationwide. Payments are processed next day.
↪<a href='http://www.dragonpay.ph/bdo-atm-how-to/' target='_blank'>Click here for␣
↪details</a>. Payments are processed next day.
            [dayOfWeek] => XXXXXXX
            [startTime] => 00:00
            [endTime] => 00:00
            [minAmount] => 200
            [maxAmount] => 1000000
            [mustRedirect] =>
            [surcharge] => 0
            [hasAltRefNo] => 1
            [cost] => 0
        )

)
```

Note: If an amount value greater than zero is passed, it 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 `Dragonpay::ALL_PROCESSORS`.

3.10 Updating payment url and web service url

If for some intance Dragonpay updated their payment and web service url(most likely will not happen).

Payment URL is the url where customer will be redirected to process and complete payment. **Web Service URL** is the url where we request token. **Send Billing Info URL** sending billing info for billing info for credit card payment

```
$merchant_account = [
    'merchantid' => 'MERCHANTID',
    'password'   => 'MERCHANT_KEY'
];
$dragonpay = new Dragonpay($merchant_account);

// Payment Url
$newPaymentUrl = "https://test.dp.com/Pay.aspx";
// Web Service Url
$newWebSrvcUrl = "https://test.dp.com/WebService.aspx";
$newBillingInfoUrl = "https://test.dp.com/WebServiceBilling.aspx";
$dragonpay->setPaymentUrl($newPaymentUrl)
        ->setBillingInfoUrl($newBillingInfoUrl)
        ->setWebServiceUrl($newWebSrvcUrl);
```

Note: The code above will change the api urls of the sandbox. You just need to pass *boolean false* as 2nd parameter of *Dragonpay* class. It should look like this:

```
$is_sandbox = false;
$dragonpay = new Dragonpay($merchant_account, $is_sandbox);
```

3.11 Tips

Do not use email domain `@example.com`. It seems the Payment switch does not accept it.

3.12 Miscellaneous

If you found any security issues or bugs, it will be a big help if you raise an issue or email the author directly and will address it right away.

3.13 Author

Jeff Claud