GPWebPay-Core Documentation

Release latest

Ondra Votava

Feb 06, 2023

Contents

Edition for GPWebPay-Core latest. Updated on Feb 06, 2023.

Ondra Votava

CHAPTER 1

Contents:

1.1 Installing GPWebPay Core

1.1.1 Requirements

GPWebPay Core latest requires PHP ^7.4 or ^8.0 using the latest version of PHP is highly recommended.

GPWebPay Core requires the ext-openssl extensions

1.1.2 Composer

Simple add a dependency on‘‘pixidos/gpwebpay-core‘‘ to your project's `composer.json` file.
If you use Composer to manage the dependencies of your project:

```
composer require pixidos/gpwebpay-core
```

1.2 Configuration

You are need create configuration for gateway. Library supporting single or multiple gateways, because for every currency you are need one.

1.2.1 Attributes

The *privateKey* attribute

Absolute path to your private certificate *pem* file.

Type `string`

The *privateKeyPassphrase* attribute

Passphrase of your private certificate

Type `string`

The *publicKey* attribute

Absolute path to GPWebPay service public certificate `pem` file.

Type `string`

The *url* attribute

Absolute URL to gateway

Type `string`

Note: You are get from GPWebPay service

The *depositFlag* attribute

Specifies if the order has to be paid for automatically.
`0` = instant payment not required
`1` = payment required

Type `int`

The *merchantNumber* attribute

Your Merchant Number

Type `string`

Note: You are get from GPWebPay service.

The *responseUrl* attribute

Absolute URL to your site where GPWebPay sends a response

Type `string`

It is optional in config. You can set up url for each request in Operation object.

Warning: GPWebPay is recommendation does not use url which has parameters (after '?') because they may drop it for *"security reason"*.

1.2.2 Examples

Single gateway

```
use Pixidos\GPWebPay\Config\Factory\ConfigFactory;
use Pixidos\GPWebPay\Config\Factory\PaymentConfigFactory;

$configFactory = new ConfigFactory(new PaymentConfigFactory());

$config = $configFactory->create(
        [
                ConfigFactory::PRIVATE_KEY => __DIR__ . '/_certs/test.pem',
                ConfigFactory::PRIVATE_KEY_PASSPHRASE => '1234567',
                ConfigFactory::PUBLIC_KEY => __DIR__ . '/_certs/test-pub.pem',
                ConfigFactory::URL => 'https://test.3dsecure.gpwebpay.com/unicredit/
↪order.do',
                ConfigFactory::MERCHANT_NUMBER => '123456789',
                ConfigFactory::DEPOSIT_FLAG => 1,
                ConfigFactory::RESPONSE_URL => 'http://example.com/proccess-gpw-
↪response'
        ]
);
```

Multiple gateways

```
use Pixidos\GPWebPay\Config\Factory\ConfigFactory;
use Pixidos\GPWebPay\Config\Factory\PaymentConfigFactory;

$configFactory = new ConfigFactory(new PaymentConfigFactory());

$config = $configFactory->create(
        [
                'czk' => [
                        ConfigFactory::PRIVATE_KEY => __DIR__ . '/_certs/test.pem',
                        ConfigFactory::PRIVATE_KEY_PASSPHRASE => '1234567',
                        ConfigFactory::PUBLIC_KEY => __DIR__ . '/_certs/test-pub.pem',
                        ConfigFactory::URL => 'https://test.3dsecure.gpwebpay.com/
↪unicredit/order.do',
                        ConfigFactory::MERCHANT_NUMBER => '123456789',
                        ConfigFactory::DEPOSIT_FLAG => 1,
                ],
                'eur' => [
                        ConfigFactory::PRIVATE_KEY => __DIR__ . '/_certs/test2.pem',
                        ConfigFactory::PRIVATE_KEY_PASSPHRASE => '12345678',
                        ConfigFactory::PUBLIC_KEY => __DIR__ . '/_certs/test-pub2.pem
↪',
                        ConfigFactory::URL => 'https://test.3dsecure.gpwebpay.com/
↪unicredit/order.do',
                        ConfigFactory::MERCHANT_NUMBER => '123456780',
                        ConfigFactory::DEPOSIT_FLAG => 1,
                ],
        ],
        'czk' // what gateway is default
);
```

1.3 Services

After you create the *Configuration* you are need instance a few next services.

1.3.1 SignerProvider

Signer provider is the services what provide *Signer* for each gateway.

```
use Pixidos\GPWebPay\Signer\SignerFactory;
use Pixidos\GPWebPay\Signer\SignerProvider;

$signerProvider = new SignerProvider(new SignerFactory(), $config->
↪getSignerConfigProvider());
```

1.3.2 RequestFactory

Request factory is helper what provide creating *Request* object from *The Operation*

```
use Pixidos\GPWebPay\Factory\RequestFactory;
$requestFactory = new RequestFactory($config->getPaymentConfigProvider(),
↪$signerProvider);
```

1.3.3 ResponseFactory

Service for creating *Response* from received params

```
use Pixidos\GPWebPay\Factory\ResponseFactory;
$responseFactory = new ResponseFactory($config->getPaymentConfigProvider());
```

1.3.4 ResponseProvider

Is service what validate and can processed *Response*

```
use Pixidos\GPWebPay\ResponseProvider;

$provider = new ResponseProvider(
        $config->getPaymentConfigProvider(), $signerProvider
);
```

1.4 Operation and Request

For creating request to GPWebPay API you are need create two objects. `Operation` and `Request`

1.4.1 The Operation

For basic operation must create *OrderNumber*, *Amount* and *Currency* objects with values of your payment order.

```php
use Pixidos\GPWebPay\Data\Operation;

$operation = new Operation(
        $orderNumber,
        $amount,
        $currency,
   'czk', // optional, if you leave empty so will be use default
        new ResponseUrl('http://example.com/proccess-gpw-response') // optional when you setup in config
);
```

more *Parameters* you can simple add by method *addParam(IParam $param)*

For example:

```php
$operation->addParam(new PayMethods(PayMethod::CARD(), PayMethod::GOOGLE_PAY()));
```

1.4.2 Request and Rendering

Request you create by *RequestFactory*

```php
$request = $requestFactory->create($operation);
```

And render the payment button

```php
echo sprintf('<a href="%s">This is pay link</a>', $request->getRequestUrl());
```

or you can render `form` for `post` method

```php
<form action="<?= $request->getRequestUrl(true) ?>">
        <?php
        /** @var IParam $param */
        foreach ($request->getParams() as $param) {
                echo sprintf('<input type=hidden value="%s" name="%s">%s', $param->getValue(), $param->getParamName(), "\n\r");
        }
        ?>
        <input type="submit" value="Pay">
</form>
```

1.4.3 Parameters

OrderNumber

Ordinal number of the order. Every request from a merchant has to contain a unique order number.

> **Warning:** Is not your order number! For specify you order number use *MerOrderNum* parameter

You are have two ways how specify this.

```php
// you can create on time base on any other integer unique generator.
$orderNumber = new OrderNumber(time());
```

Amount

Because the amount is the smallest units of the relevant currency For CZK = in hellers, for EUR = in cents.

You are have two ways how specify this.

```
// The conversion will make Amount self
$amount = new Amount(1000.00);
// or create the conversion by yourself
$amount = new Amount(100000, false);
```

Warning: It will be deprecated in next major version. Please replace for *AmountInPennies*

AmountInPennies

Amount of order

```
$amount = new AmountInPennies(100000); // represent 1000.00 Kč|Euro
```

Currency

Currency identifier according to ISO 4217 (see Addendum ISO 4217).

You are simple create this, because in class `Pixidos\GPWebPay\Enum\Currency` you are have all constants with ISO code and methods for create the enum.

```
use Pixidos\GPWebPay\Enum\Currency as CurrencyEnum;

$currency = new Currency(CurrencyEnum::CZK())
```

MerOrderNum

Order identification for the merchant. If not specified, the *OrderNumber* value is used

```
use Pixidos\GPWebPay\Param\MerOrderNum;

$merOrderNum = new MerOrderNum(123455);
```

1.5 Response

For processing response from GPWebPay your are need *ResponseFactory* and *ResponseProvider*

1.5.1 Create response

```
$params = $_GET; // or $_POST depends on how you send request
$response = $responseFactory->create($params);
```

Now you are have two options.

Processing by your code.

```
// verify response signatures
if (!$provider->verifyPaymentResponse($response)) {
        // invalid verification
}
// success verification
if ($response->hasError()) {
        // here is you code for processing response error
}
// here is you code for processing response
```

Or use ResponseProvider and callbacks

```
// success callbacks
$provider
        ->addOnSuccess(
                static function (ResponseInterface $response) {
                        // here is your code for processing response
                }
        )
        ->addOnSuccess(
                static function (ResponseInterface $response) {
                        // here is your another code for processing response
                }
        );

// error callback
$provider->addOnError(
        static function (GPWebPayException $exception, ResponseInterface $response) {
                // here is you code for processing error
        }
);

// and next step is call
$provider->provide($response);
```

1.5.2 Parameters

ResultText

A text description of the error identified by a combination of PRCODE and SRCODE. The contents are coded using the Windows Central European (Code Page 1250).

return `string`

```
$response->getResultText();
```

PrCode

Primary code. For details, see "List of return codes in GPWebPay doc".

return `int`

```
$response->getPrcode();
```

SrCode

Secondary code. For details, see "List of return codes in GPWebPay doc".

return `int`

```
$response->getSrcode();
```

OrderNumber

Contents of the field from the request.

return `string`

```
$response->getOrderNumber();
```

MerOrderNumber

Contents of the field from the request, if included.

return `string|null`

```
$response->getMerOrderNumber();
```

UserParam1

Hash numbers of the payment card. Hash is a unique value for each and every card and merchant – that is if the payment is made by the same card at the same merchant, the resulting hash is identical, if the same card is used at another merchant, there is another hash.

return `string|null`

Note: Only if the merchant has this functionality enabled

```
$response->getUserParam1();
```

Md

Any merchant's data returned to the merchant in the response in the unchanged form – only "whitespace" characters are removed from both sides. The field is used to satisfy various demands of the e-shops. The field may only contain ASCII characters ranging from 0x20 to 0x7E.

Note: GPWebPay core use this field to store information about used gateway.

So method `$response->getParam(string $paramName)` return value contain gateway info.

---

return `string|null`

```
$response->getMd();
```

Digest and Digest1

Digest is a check signature of the string generated as a concatenation of all the fields sent in the given order

Digest1 is same as *Digest* but (without the DIGEST field) and on the top of that also the MERCHANTNUMBER field (the field is not sent, the merchant has to know it, the field is added to the end of the string).

return `string`

```
$response->getDigest();
$response->getDigest1();
```

1.6 Changelog

Changed in version 2.0.0.

Config refactoring

BC Breaks: drop Provider, use Request factory instead config have new signature, see documentation Interfaces renamed from I<name> to <name>Interface New Features add ResponseProvider