Banca Transilvania iPay SDK Documentation

The Banca Transilvania iPay SDK facilitates communication between payment modules and the iPay API, offering a streamlined integration for handling payments through Banca Transilvania's iPay system.

Features

- Easy integration with the iPay API.
- Secure handling of payment transactions.
- Comprehensive error handling and logging.

Requirements

- PHP 7.4 or higher
- cURL extension
- JSON extension
- OpenSSL extension

Installation

Install the package via Composer:

```
composer require banca-transilvania/ipay-sdk
```

Usage

Here's a simple example of how to use the SDK:

```
use BTransilvania\Api\IPayClient;

$this->iPayClient = new \BTransilvania\Api\IPayClient(
    [
        'user'         => $user,
        'password'     => $password,
        'environment'  => $environment,
        'platformName' => $platformName,
        'language'     => $language
    ]
```

```
);

// Initiate a payment
$result = $this->iPayClient->register($data);

if ($result->isSuccessful()) {
    echo "Payment initiated successfully. Transaction ID: " .
$result->getTransactionId();
} else {
    echo "Payment initiation failed: " . $result->getErrorMessage();
}
```

Configuration

You can configure the SDK by passing configuration parameters to the IPayClient constructor. Here are some of the configuration options available:

- `user`: Your iPay username.
- `password`: Your iPay password.
- `environment`: Set to `sandbox` for testing or `production` for live payments.
- `platformName`: The platform name where the integration is being used.
- `language`: Language preference for the API interaction.

Technical Documentation

The main class of the SDK is located at:

`ipay-sdk/src/IPayClient.php`

Within this class, the configuration, `HttpClient`, and logger are initialized in the constructor. You have the option to provide your own `HttpClient`, but ensure that the class implements:

`ipay-sdk/src/HttpAdapter/HttpClientInterface.php`

If you do not provide an `HttpClient`, the SDK will check if GuzzleHttp is installed. If GuzzleHttp is not available, the SDK will default to using `CurlHttp`.

You may also provide your own logger. If you prefer to keep all logs centralized, provide a logger that implements either:

- `src/Logger/LoggerInterface.php` or
- `Psr\Log\LoggerInterface`

If no logger is provided, the SDK will use the default logger, which writes logs to the SDK folder under `logs/btsdk.log`.

Example Logger Usage

```
use BTtransilvania\Api\Logger\PsrLogger;
```

```
\$sdk_config = [
    'user'         => \$user,
    'password'     => \$password,
    'environment'  => \$env,
    'platformName' => 'PrestaShop ' . _PS_VERSION_,
    'language'     => \$this->context->getLanguageIso()
];

try {
    \$sdkLogger = new PsrLogger(\$this->logger);
    \$this->sdkClient = new IPayClient(\$sdk_config, null, \$sdkLogger);
} catch (\Throwable \$throwable) {
    \$this->logger->error(\$throwable->getMessage());
}
```

Available API Methods

In the `ipay-sdk/src/IPayClient.php`, you will find all the available actions:

- `public function register(array \$data)`
- `public function registerPreAuth(array \$data)`
- `public function deposit(array \$data)`
- `public function reverse(array \$data)`
- `public function refund(array \$data)`
- `public function getOrderStatusExtended(array \$data)`
- `public function getFinishedPaymentInfo(array \$data)`
- `public function paymentOrderBinding(array \$data)`
- `public function getBindings(array \$data)`
- `public function unBindCard(array \$data)`
- `public function bindCard(array \$data)`

Adding a New Action

To add a new action, you need to modify the `ipay-sdk/src/Action/ActionFacadeFactory.php`. You will also need to create a corresponding `RequestModel` and `ResponseModel`.

Modifying a Request

To modify an existing request, update the corresponding request class. For example, to add a new parameter to the order creation request, modify:

`ipay-sdk/src/Model/Request/RegisterRequestModel.php`

Add a new property and process it in the `buildRequest()` function.

Modifying a Request

To modify an existing request, update the corresponding request class. For example, to add a new parameter to the order creation request, modify:

`ipay-sdk/src/Model/Request/RegisterRequestModel.php`

Add a new property and process it in the `buildRequest()` function.

Response Handling

In `ipay-sdk/src/Model/Response/ResponseModel.php`, you will find all possible methods to interact with the response:

```php
/**
 * Checks if the response indicates a successful operation.
 *
 * @return bool True if the operation was successful, False otherwise.
 */
public function isSuccess(): bool
```

```php
/**
 * Retrieves the error message, if any, from the response.
 *
 * @return string|null The error message or null if not applicable.
 */
public function getErrorMessage(): ?string
```

```
{
   return \$this->errorMessage;
}
```

```
/**
 * Converts the response model back into an associative array.
 * Useful for cases where we need to work with the model data in
simplified data structures.
 *
 * @return array The model representation as an associative array.
 */
public function toArray(): array
{
   return (array)\$this->response;
}
```

API Reference

The Banca Transilvania iPay SDK provides multiple API calls to facilitate different operations.

1. Bind Card

Binds a card to a user for future transactions.

Parameters

- `bindingId` (string): The unique identifier of the saved card.

Example

```
$data = [
    'bindingId' => 'cb02cbbb-5a81-426a-92f2-e78f5cafdfff'
];

$result = $iPayClient->bindCard($data);
```

2. Deposit

Deposits funds for a specified transaction.

Parameters

- `orderId` (string): The unique identifier for the transaction.

- `amount` (int): The amount to deposit in the smallest currency unit.

Example

```
$data = [
    'orderId' => '111aa4b9-baeb-4676-bbde-c5e4d5070cce',
    'amount'  => '1200'
];
$result = $iPayClient->deposit($data);
```

3. Get Bindings

Retrieves saved cards for a specific client.

Parameters

- `clientId` (string): Unique customer identifier in the merchant's system.
- `showExpired` (bool): Whether to include expired cards.

Example

```
$data = [
    'clientId'   => 'cb02cbbb-5a81-426a-92f2-e78f5cafdfff',
    'showExpired' => true
];
$result = $iPayClient->getBindings($data);
```

4. Get Finished Payment Info

Gets information about a completed payment.

Parameters

- `orderId` (string): The unique identifier for the transaction.
- `token` (string): A temporary token linked to the payment page.
- `language` (string): The language setting.

Example

```
$data = [
    'orderId' => '111aa4b9-baeb-4676-bbde-c5e4d5070cce',
    'token'  => '111aa4b9baeb4324324',
    'language' => 'en'
];
$result = $iPayClient->getFinishedPaymentInfo($data);
```

5. Get Order Status Extended

Retrieves extended status information for an order.

Parameters

- `orderNumber` (string): The specific order number from your request.

Example

```
$data = [
    'orderNumber' => '8042112'
];
$result = $iPayClient->getOrderStatusExtended($data);
```

6. Payment Order Binding

Associates a saved card with an order for payment.

Parameters

- `mdOrder` (string): Unique order number on the payment page.
- `bindingId` (string): The unique identifier of the saved card.

Example

```
$data = [
    'mdOrder'   => 'cb02cbbb-5a81-426a-92f2-e78f5cafdfff',
    'bindingId' => '22b963cf-cb2b-4ca0-9ca8-a3630492543e'
];
$result = $iPayClient->paymentOrderBinding($data);
```

7. Refund

Initiates a refund for a specific transaction.

Parameters

- `orderId` (string): The unique identifier for the transaction.
- `amount` (int): The amount to refund in the smallest currency unit.

Example

```
$data = [
    'orderId' => '111aa4b9-baeb-4676-bbde-c5e4d5070cce',
    'amount'  => '1200'
];
$result = $iPayClient->refund($data);
```

8. Register Payment

Registers a new payment request.

Parameters

- `orderNumber` (string): The specific order number for this transaction.
- `amount` (int): The transaction amount in the smallest currency unit.
- `currency` (string): The currency code as per ISO 4217.
- `description` (string): A description of the transaction.
- `returnUrl` (string): The URL to which the customer will be redirected after payment.
- `orderBundle` (array): Additional details about the order, such as customer information.

Example

```
$data = [
    'orderNumber' => '209126',
    'amount'      => '1000',
    'currency'    => 'RON',
    'description' => 'testBT',
    'returnUrl'   => 'https://magazinulmeu.ro/finish.html',
    'orderBundle' => [
        'orderCreationDate' => '2020-09-29',
        'customerDetails'   => [
            'email' => 'email@test.com',
            'phone' => '40740123456'
        ]
    ]
];
$result = $iPayClient->register($data);
```

9. Reverse Payment

Cancels a specific transaction.

Parameters

- `orderId` (string): The unique identifier for the transaction.

Example

```
$data = [
    'orderId' => '111aa4b9-baeb-4676-bbde-c5e4d5070cce'
];
$result = $iPayClient->reverse($data);
```

10. Unbind Card

Removes a saved card from a user's account.

Parameters

- `bindingId` (string): The unique identifier of the saved card.

Example

```
$data = [
    'bindingId' => 'cb02cbbb-5a81-426a-92f2-e78f5cafdfff'
];
$result = $iPayClient->unBindCard($data);
```

Error Handling

The SDK provides comprehensive error handling to manage different failure scenarios. Each API response contains information on whether the request was successful, and if not, includes an error message that describes the issue.

Example of handling errors:

```
if (!$result->isSuccessful()) {
    echo "Error: " . $result->getErrorMessage();
}
```

Logging

The SDK includes logging functionality to help developers trace issues.

Testing

To test your integration, use the `sandbox` environment. The sandbox environment provides a safe space to test all API calls without initiating real payments.

Contributing

Contributions are welcome, and any help is greatly appreciated! See the `CONTRIBUTING.md` file for how to get started.

Support

For support, please contact contact@bancatransilvania.com or visit the Banca Transilvania ePOS website.