1.2. API_general_ENG

2025.06.25

PayPo API v.3.1

The document contains technical specification of REST API PayPo.pl ver. 3.0

Document version	Date	List of changes
3.1.1	2025.06.25	Changes in “Notifications“ senction
3.1.0	2023.09.20	Extension of response to the payment status verification request
3.0.9	24 February 2023	Added an example of the response http 400 error Changes concerning field-installmentCount
3.0.8	16 September 2022	Changes in RegisterTransactionPayload and Validation methods
3.0.7	25 April 2022	Changes concerning new fields for addresses
3.0.6	08 November 2021	Changes concerning new fields for product configuration
3.0.5	03 August 2021	Changes concerning notifications

3.0.4	2 October 2020	Changes to definitions and endpoints
3.0_r3	16 July 2020	Notifications on changes to the status and order amount
3.0_r2	1 June 2020	Authentication of OAuth messages
3.0_r1	12 May 2020	Clarification of address attributes description, HMAC authorization
3.0_r0	28 April 2020	Definition of basic endpoints, transaction life cycle.

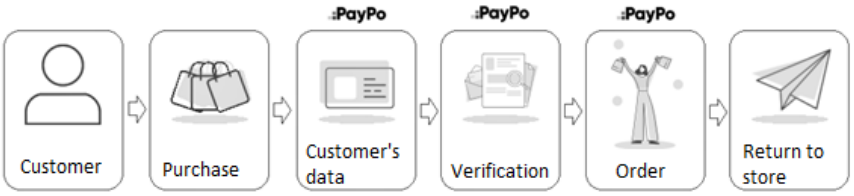
Table of Contents

- PayPo API v.3.1
- 1. Service description
- 2. Integration: general information
- 3. New order creation
  - 3.1 Payment registration
  - 3.2 Server response
  - 3.3 Buyer's data verification and return to the store
- 4. Statuses and the transaction life cycle
- 5. Notifications
  - 5.1 Changes to the payment status or transaction amount
  - 5.2 Generation of specification for wire transfer
- 6. Transaction confirmation
- 7. Payment status verification
- 8. Transaction attributes modification
  - 8.1 Payments and drafts cancellation
  - 8.2 Refunds
- 9. Authentication
  - 9.1 OAuth2
  - 9.2 HMAC
- 10. Operation statuses

1. Service description

PayPo provides deferred payment services to customers of online stores (Buyers). New transaction is registered online using API, similarly to popular payment systems.

Course of transaction:



- After products are chosen, the Buyer selects deferred payment for the purchase with PayPo.
- The Buyer's data are transferred to PayPo service.
- PayPo system verifies financial credibility of the Buyer.
- Positive verification results in the he postponement of payment for the order.
- The Buyer is redirected to the store website.

Further steps:

- The Seller confirms order completion in PayPo.
- Funds are transferred to the Seller after the order status is updated within the time limit stipulated in the contract.
- The Buyer pays the purchase costs to PayPo in accordance with the terms of PayPo deferment .

2. Integration: general information

API is available at: https://api.paypo.pl/v3/

Test server address is: https://api.sandbox.paypo.pl/v3/

Domain of API links is depends on country, ex. for Romania - https://api.paypo.**ro**/v3/

PayPo API is based on REST architecture. Communication is carried out using HTTP protocol (POST/GET/PATCH methods) and data are exchanged in JSON format. The main resource made available to the Sellers are transactions.

Sample payment registration:

```
curl -X POST_https://api.paypo.pl/v3/transactions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer fLEsy5wk9jQKK0NjbF57sbOwa4BdCPul" \
-d '{
   "id": "5c1b82ab-6c9a-4b4e-a892-ce3a7dc1396f",
   "shopId": "088fa21e-efab-4ecb-9022-a15cc8344ccd",
   "order": {
      "referenceId": "ord_98765/20",
      "providerId": "180abc",
      "description": "test",
      "additionalInfo": {"someKey": "someKeyValue"},
      "amount": "24900",
```

```
    "billingAddress": {
        "street": "Kredytowa",
        "building": "9a",
        "flat": "3",
        "zip": “00-950",
        "city": "Warszawa",
        “county“: “mazowieckie”,
        “country“: “PL“,
    },
    "shippingAddress": {
        "street": "Domaniewska",
        "building": "39",
        "flat": "",
        "zip": "02-672",
        "city": "Warszawa",
        “county“: “mazowieckie”,
        “country“: “PL“,
    },
    "shipment”: “0”,
},
"customer": {
    "name": "Anna",
    "surname": "Nowak",
    "email": "anna.n@paypo.pl",
    "phone": "+48500123456",
    "registrationInfo": {
            "isRegistered": true,
            "dateOfRegistration": "2022-01-01" ,
        },
    "transactionsInfo": {
             "numberOfTransactions": 1,
             "sumOfTransactions": “123456”,
      }
},
"configuration": {
    "returnUrl": "https://merchant.com/complete",
    "notifyUrl": "https://merchant.com/notify",
    "cancelUrl": "https://merchant.com/cancel",
```

```
    "product": {
            "productType": "CORE",
            "process": "online",
            "installmentCount": “4”,
    }
  }
}'
```

3. New order creation

3.1 Payment registration

URL: https://api.paypo.pl/v3/transactions

In order to create a new payment session, basic transaction data need to be transferred to PayPo. After positive verification of the message, URL address of the verification form will be returned and the Buyer needs to be redirected to it.

After identity has been verified and financial credibility has been confirmed, PayPo refers the Buyer back to the store website (`returnUrl`) and sends a notification that the order status has been changed. The transaction life cycle is described in item 4 of the document.

Input parameters of the request (HTTP POST):

Parameter name		Type	Required	Description	Validation method
id		uuid	N	Transaction ID	Is the value in UUID format
shopId		uuid	N	Store ID allocated by PayPo (optional, if the Seller or Payment Operator integrates separate sales points)	Is the value in UUID format
order		{}	Y	Order data	-
	referenceId	string	Y	Transaction ID on the Seller's side, used to identify wire transfers	Is the field empty and is the value in string format
	providerId	string	N	Transaction ID in the Payment Operator’s system (if integrated by the Operator)	Is the value in string format

	description	string	N	Short transaction description/type	Is the value in string format
	additionalInfo	{}	N	Additional order description, e.g. content of the basket	-
	amount	int	Y	Order amount in main monetary units, e.g. grosz	Is the value larger than “0”
shipment		int	N	Delivery method: 0 - courier (default) 1 - delivery to a collection point (e.g. UPS Access point, DHL Parcel Shop) 2 - parcel kiosk 3 - parcel in RUCH 4 - pickup in store (click&collect)	Does the value comply with possible values from the range <0,4>
billingAddress			Y	Billing address:	-
	street	string	Y	Street	Is the value empty
	building	string	N	Building number	Is the value in string format, max length 16 characters
	flat	string	N	Flat number	Is the value in string format, max length 16 characters
	zip	string	N/Y	Postal code - In PL Required, in RO not Required (market specificity)	Does the value consists of 6

					characters
	city	string	Y	Billing address: city	Is the value empty, min length 2, max length 255
	county	string	N	Billing addess: county	Is the value in string format
	country	string	N	Billing address: Country code, e.gp. PL, RO (PL by default)	Does the value comply with ISO 3166-1 alpha-2
shippingAddress			Y	Delivery address:	-
	street	string	Y	Street	Is the value empty
	building	string	N	Building number	Is the value in string format, max length 16 characters
	flat	string	N	Flat number	Is the value in string format, max length 16 characters
	zip	string	N/Y	Postal code - In PL Required, in RO not Required (market specificity)	Does the value consists of 6 characters
	city	string	Y	City	Is the value empty,

						min. length 2, max length 255
	county		string	N	County	Is the value in string format
	country		string	N	Country code, e.gp. PL, RO (PL by default)	Does the value comply with ISO 3166-1 alpha-2
customer			{}	Y	Buyer's data	-
	name		string	Y	First Name	Is the value empty
	surname		string	Y	Surname	Is the value empty
	email		string	Y	Email address	Is the value correct
	phone		string	N	Mobile phone number with country prefix. If no prefix is provided, the prefix for the country will be adopted by default	Is the value correct and mobile
	registrationInfo		{}	N	Information about registration in the store (dedicated for RO market)	-
		isRegistered	boolean	N	Is the customer registered in store?	-
		dateOfRegistration	string	N	Registration date in YYYY-MM-DD format	-
	transactionsInfo		{}	N	Information about transactions in store (dedicated for RO market)	-
		numberOfTransactions	string	N	Number of transactions	-
		sumOfTransactions	integer	N	Sum of transactions in main monetary units, e.g. grosz	-

configuration			{}	Y	Return and notification addresses	-
	returnUrl		string	Y	Return address after the payment has been processed	Correct URL
	notifyUrl		string	Y	The address which notifications will be sent to	Correct URL
	cancelUrl		string	N	Optional return address in a situation where the Buyer renounces PayPo payment; if the address is not defined, returnUrl will be used by default	If the provided value is the correct URL
	product			N	Product configuration	
		productType	string	N	Product type: • CORE • PNX	Is value match with value in the list
		process	string	N	Process type	-
		installmentCount	int	N	Number of installment	Is the value in the range from 1 to 12

Requests authentication is described in detail in item 9 of the document. In API ver. 3.0.x the message header must contain OAuth token.

3.2 Server response

In response to calling the method `/transactions`, PayPo returns the operation status and URL of the form which the Buyer needs to be redirected to.

Sample server response:

```
HTTP 201 Created

{
   "transactionId": "5909da74-af95-41e9-b8e2-12e61c3c6f27",
   "redirectUrl": "https://process.paypo.pl/5909da74-af95-41e9-b8e2-12e61c3c6f27"
}
```

3.3 Buyer's data verification and return to the store

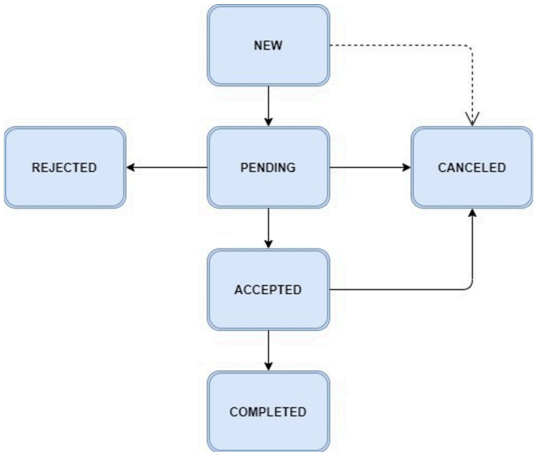
After redirection to PayPo the Buyer's data are completed and verified. After the payment has been processed, the Buyer is redirected to `returnUrl`. GET parameters of the return address contain payment status so that the relevant message is displayed in the browser.

https://merchant.com/thankyou?status=[OK|ERR]

If the message initiating the transaction also contained cancelUrl attribute, resignation during the verification process results in being redirected to that address (e.g. to the selection screen with other payment methods at checkout).

4. Statuses and the transaction life cycle

Payment statuses:



Status	Description
NEW	New transaction
PENDING	Payment is being processed, it is awaiting confirmation of Buyer's identity or of funds availability
ACCEPTED	Payment accepted
COMPLETED	Transaction confirmed by the Seller, payment ready to be cleared
REJECTED	Refusal to complete the payment
CANCELED	Transaction cancelled

After successful registration the payment status becomes `NEW`, after the Buyer is redirected the status changes to PENDING, then the transaction may be accepted (`ACCEPTED`) or rejected (`REJECTED`) by PayPo system.

Transaction may also be accepted (`ACCEPTED`) after first having been rejected (`REJECTED`). This is possible in a situation where the Buyer gets access to a link enabling him to return to payment and PayPo grants the deferred payment.

Payment session may be cancelled before it is confirmed and before the status changes to `COMPLETED` .

When verifying the statuses, Merchant should take into account their hierarchy, e.g. in the case of receiving the `COMPLETED` status, it should be considered that the statuses `NEW` , `PENDING` and `ACCEPTED` have already been sent. The situation is similar for the `REJECTED` and `CANCELED` statuses - the `NEW` and `PENDING` statuses should be considered as received.

5. Notifications

5.1 Changes to the payment status or transaction amount

After payment status or transaction amount is changed, detailed information is sent in a notification to `notifyUrl` . The Seller's system should respond to the notification by returning HTTP 200 status.

Notification parameters (HTTP POST):

Parameter name	Type	Description
merchantId	uuid	Seller's ID
shopId	uuid	POS ID (if defined at transaction registration)
referenceId	string	Transaction ID on the Seller's side
transactionId	uuid	PayPo transaction ID
transactionStatus	string	Order status, in line with the table in item 4; an exception is the status "NEW" which is not forwarded
transactionUrl	string	Link with the address to deferred payment
amount	int	The current transaction amount in main monetary units, e.g. grosz
lastUpdate	timestamp	Update date in RFC3339 format

Notification headers also contain the document type and MAC. MAC calculation and verification algorithm is described in item 9.2 of the document.

Sample notification:

```
POST /notifyUrl HTTP/1.1
X-PayPo-Signature: dfcffcd0cbed7bbbd11b7d23f3a2e7a7458ecf90
{
   "merchantId": "19c692be-a893-468c-a65f-b8de442e5443",
   "referenceId": "ord_987654",
   "transactionId": "00102030",
   "transactionStatus": "PENDING",
```

```
  "amount": 24900,
  c: „2020-03-05T10:54:02+01:00”,
}
```

If there is no response or if the received status is different than HTTP 20x, PayPo system will retry sending the notification in line with the following pattern (configuration of retries may differ in non-production environments).

The first hour after successful verification of transaction	Every 10 minutes
Another 5 hours	Every 20 minutes
Another 18 hours	Every 60 minutes

In case when Client was redirected to `returnUrl` and the Seller's system has not yet received anotification about the payment status, Seller should verify its status manually as described in "Payment status verification" section.

5.2 Generation of specification for wire transfer

Upon generation of specification for wire transfer, a notification is sent to `notifyUrl`. The system should respond to the notification by returning HTTP 200 status, if notification is successfully received, and the relevant status 4xx or 5xx, if an error occurs.

Notification parameters (HTTP POST):

Parameter name	Type	Required	Description
merchantId	uuid	Y	Seller's ID
referenceId	string	Y	Transaction ID on the Seller's side
transactionId	uuid	Y	PayPo transaction ID
transactionStatus	string	Y	Order status, in line with the table in item 4
amount	int	Y	The current transaction amount in main monetary units, e.g. grosz
lastUpdate	timestamp	Y	Update date in RFC3339 format
settlementStatus	string	Y	Payment status, in line with the table in item 7
message	string	Y	Additional information
shopId	uuid	T	POS ID (if defined at transaction registration)

Sample notification:

```
{
```

```
  "merchantId":"0e1576d8-e760-4336-8bc4-c20a549ac035",
  "referenceId":"QQBF6HAWVGI972291WQQ",
  "transactionId":"9207c39a-1d1f-4954-a312-fe5dcd1a1f8a",
  "transactionStatus":"COMPLETED",
  "amount":1212,
  "lastUpdate":"2021-07-27T18:44:51.000+02:00",
  "settlementStatus":"PAID",
  "message":"Transaction is settled",
  "shopId":"c8847732-b5d7-4528-8a7b-54ede5e43789"
}
```


6. Transaction confirmation

After payment is successfully verified, notification is sent and Buyer is redirected to the store website, PayPo system waits for the Seller to confirm order completion. To confirm the order, the Seller's system should send a request for status change to `COMPLETED`. Each accepted payment may be confirmed individually (autodelivery parameter), configuration is custom-made at the Seller's request.The autodelivery parameter is required for the activated PayPo payment gateway service.

Unconfirmed orders may be cancelled by PayPo after a given time: configuration is custom-made at the Seller's request.

URL: https://api.paypo.pl/v3/transactions/${UUID}

Input parameters (HTTP PATCH):

Parameter name	Value	Description
status	COMPLETED	Confirmation of order completion

Correct request receipt will be confirmed by a response with the status 200. After payment status has been updated, the relevant notification will be sent to notifyUrl provided upon payment registration.

Sample response:


```
HTTP 200 OK
{
   "code": 200,
   "message": "Transaction updated successfully",
}
```

7. Payment status verification

The current order status may be viewed at any time by calling HTTP GET. The request does not require any parameters other than the payment ID sent in the URL. The query should be authenticated with Oauth token.

URL: https://api.paypo.pl/v3/transactions/${UUID}

Sample response:

```
HTTP 200 OK

{
   "merchantId": "19c692be-a893-468c-a65f-b8de442e5443",
   "referenceId": "ord_987654",
   "transactionId": "cd975bc6-a755-4141-b7a0-d7e8f7a308ef",
   "transactionStatus": "COMPLETED",
   "amount": 24900,
   "settlementStatus": "PAID",
   "lastUpdate": "2020-03-05T10:54:02",
}
```

Apart from basic parameters of the transaction, the response also contains payment settlement status `settlementStatus`; it may take one of the following three values:

Settlement status	Description
NEW	New transaction
CONFIRMED	Order completion has been confirmed, the transaction is ready to be settled
PAID	The transaction has been settled with the Seller

When verifying transaction status, on merchant's request, PayPo may return extended version of response, containing information about processed transaction refunds.

Sample extended response:

```
HTTP 200 OK

{
"merchantId": "19c692be-a893-468c-a65f-b8de442e5443",
"referenceId": "ord_987654",
"transactionId": "cd975bc6-a755-4141-b7a0-d7e8f7a308ef",
"transactionStatus": "COMPLETED",
"amount": 24900,
```

```
"settlementStatus": "PAID",
"lastUpdate": "2020-03-05T10:54:02",
"refunds": [
 {
  "referenceRefundId": "3e12a361-d193-4f3a-88b5-b8fda405a529",
  "amount": 8655,
  "created": "2023-09-08T13:46:04+02:00",
 },
 {
  "referenceRefundId": null,
  "amount": 855,
  "created": "2023-09-08T13:46:04+02:00",
 }
 ]
}
```


8. Transaction attributes modification

The Seller may modify the transaction status and the order amount. The permissible status changes are cancellation (`CANCEL`) and confirmation (`COMPLETED`) described above. A dedicated method of refunds registration is used to modify the order amount /refunds.

8.1 Payments and drafts cancellation

URL: https://api.paypo.pl/v3/transactions/${UUID}

Input parameters (HTTP PATCH):

Parameter name	Value	Description
status	CANCELED	Order cancellation

Payment cancellation may be requested by the Seller, if it is not possible to complete the order.

Sample response:


```
HTTP 201 OK
{
  "code": 201,
  "message": "Transaction updated successfully",
}
```


Calling the cancellation of a confirmed transaction (status `COMPLETED`) will be rejected with error 409 Conflict.

Successful processing of change to the payment status `ACCEPTED` will also be confirmed with the relevant message in a notification sent to notifyUrl. The collected funds (if any) will be returned to the Buyer's bank account.

Drafts are cancelled the same way as payment. After having received the request, PayPo verifies on its side whether it concerns a payment or a draft.

8.2 Refunds

Refund of funds, in full or in part, for an order is possible for transactions which have been accepted (`ACCEPTED` or `COMPLETED` status). The collected funds (if any) will be returned to the Buyer's bank account. In the case of refund for a transaction having the `ACCEPTED` status, the refund will be confirmed automatically and the status will change to `COMPLETED`.

To each refund request the Seller should add the refund ID (referenceRefundId).

URL: https://api.paypo.pl/v3/transactions/${UUID}/refunds

Input parameters (HTTP POST):

Parameter name	Type	Required	Description
amount	int	Y	Refund amount stated in main monetary units, e.g. grosz
referenceRefundId	string	N	Refund ID, max. length 68 characters

Due to the specific nature of deferred payment service, it is not possible to increase the order value.

A few successive partial refunds may be made on condition that the total amount of the transferred refunds does not exceed the order value. If the entire purchase value is refunded, the ‘amount’ attribute should contain the current order value.

Sample refund request:

```
curl -X POST https://api.paypo.pl/v3/transactions/cd975bc6-a755-4141-b7a0-d7e8f7a308ef/refunds \
-H "Content-Type: application/json" \
-H "Authorization: Bearer fLEsy5wk9jQKK0NjbF57sbOwa4BdCPulc \
-d '{"amount": 24900}'
```

Sample response:

```
HTTP 201 Created
{
   "code": 201,
   "message": "Refund created successfully",
}
```

Similarly, to the status change, successful refund processing will be confirmed in a notification with updated order value sent to notifyUrl.

9. Authentication

PayPo API requires communication authentication for HTTP methods modifying the transactions POST and PATCH as well as for HTTP GET requests.

Values of the attributes required to sign messages: Seller's ID merchantId, unique key merchantApiKey and OAuth ClientId and ClientSecret are available from PayPo transaction panel level.

9.1 OAuth2

The required method of requests authentication from the Seller's system is OAuth standard ver. 2.0. PayPo implements this method in line with the protocol specification https://oauth.net/2/.

In order to get the authentication token, you need to call API method `/oauth/tokens`. The returned string is JWT encoded (https://tools.ietf.org/html/rfc7519). By default, the token is valid for 30 minutes.

Sample request:

```
curl -X POST https://api.paypo.pl/v3/oauth/tokens \

-H "Accept: application/json" \

-H "Content-Type: application/x-www-form-urlencoded" \

—data-raw "grant_type=client_credentials&client_id=${ClientId}&client_secret=${ClientSecret}"
```

Sample response:

```
HTTP 200 OK

{

   "token_type": "Bearer",

   "expires_in": 1800,

   "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIwMDRhZjNmOC1mMjczLTRlZjItOWFhOS1jO
WI0MDRjNzJjN2IiLCJqdGkiOiJkMmY1MzQ3M2M5ZWM2YjExYmZhN2Q1YTlmMzQ4MTE5YzM5Y2Nj
ZGFkOWMwZTllZjVlOTZlMDhkYWI3ZGFjMTljMzA1MWRmNGU2M2JhZjQwNiIsImlhdCI6MTYwMzk2
ODM1MCwibmJmIjoxNjAzOTY4MzUwLCJleHAiOjE2MDM5NzAxNTAsInN1YiI6IiIsInNjb3BlcyI6WyJt
ZXJjaGFudCJdfQ.V9YkODDb4EbmSp-8q_yv6vS0Q5-
VIi43Hi8BH9toA8ISKjqeyhyojCS_qDMVyPGXiOZIBi_oDHNzpXtWBuOiGP1EfYQs7NXIZhrlG72h9-
nbssuG25pycUCoF2SdGLfoGr6_uHFsODokKy6XFE2ogW0q9pqQByYPOVNyvjXG-
AYkxAURNK0l17M3oA9x2Fr1Vz95rJnVJsgodpi-
kojyYloNASJMlpXrhhAxtAE5na6mgrmx7SBsLpNmDgl_z4M9wfr1gRTwIUdqKD8yAxOV9mlYzt8TgEhz
LK2n-OaBzMwCKqPpWF0VnybckCUUyfuFAshQl3CE80lYXSDQqKUWiw"

}
```

9.2 HMAC

Notifications use authentication scheme with message hash in the header. The following data connected by the “+” sign are used to calculate the hash value.

- HTTP: POST
- URL_PATH (without the domain part) of endpoint, e.g. `/notifyUrl/${referenceId}`
- Message content in JSON format

Sample code calculating the hash in PHP:

```
$json = json_encode($payload, JSON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE);

// „POST+/notifyUrl+{...}”

$data = $method . "+" . $endpoint . "+" . $json;

$hash = base64_encode(hash_hmac('sha256', $data, $merchant_api_key, true));

$data = $method . "+" . parse_url('https://caly-url.pl/notifyUrl', PHP_URL_PATH) . "+" . $json;
```

Hash verification in the Seller’s system is not required but it is recommended for security purposes.

10. Operation statuses

HTTP statuses

In response to each request the HTTP status is returned and it informs about the result of request processing.

200 OK – successful receipt of message

201 Created – successful creation of resource

400 Bad Request – incorrect message data of format

401 Unauthorized – authentication error, e.g. invalid token

403 Forbidden – no permission to access the resource

404 Not Found – URL or resource has not been found

405 Method Not Allowed – operation is not possible for a given resource

409 Conflict – resource identification conflict (e.g. PayPo order number and Seller’s order number identify various objects)

503 Service Unavailable – service is temporarily unavailable

API calls return JSON object both for successful and failed processing. The returned HTTP response code is the main operation status determinant. In addition, in the case of error 400, detailed feedback (errors) is returned. Please see below sample responses concerning error 400.

Sample response in the case of an error:

- in the billing address,

```
HTTP 400
```

```
{
   "code": 400,
   "message": "Bad request",
   "errors": {
      { "path": „billingAddress.zip",
        "message" "Missing mandatory parameter” },
   }
}
```

- in the order data,

```
HTTP 400
{
   "code": 400,
   "message": "Bad request",
   "errors": {
     { "path": "order.referenceId",
       "message": "This value should not be blank." },
   }
}
```

- in the amount of the refund (if greater than the value of the order).

```
HTTP 400
{
   "code": 400,
   "message": "Refund amount XXXX can not be greater than order amount YYY."
}
```