Better Payment Gateway API Documentation

Version 2.9.1 - 2017-07-05

Document History

Author	Version	Date	Description
Neeme Oja	1.0		First Version
Neeme Oja	2.0	18.11.2015	Major Release: • Changed checksum calculation examples • Changed company address • Removed shop_id and marked it deprecated • No longer using same credentials for test/live environments Changed inline card payment method to use an iframe Removed legacy authorize/capture/void operations
Neeme Oja	2.0.1	08.12.2015	• Billing address is required in all payment methods
Neeme Oja	2.0.2	25.02.2016	• Updated inline card payment return values
Moritz Winter	2.0.3	19.04.2016	• Added sepa_mandate parameter
Neeme Oja	2.1	19.05.2016	New payment methods: • Invoice B2B • Installments New API requests: • calculate • precheck • authorize • capture • reverse New parameters: • shipping address • company details Renamed payment request's total_amount to amount (backwards compatible)
Neeme Oja	2.2	31.05.2016	Added credit card authorization, clarified precheck parameters and fixed a few omissions
Neeme Oja	2.3	28.06.2016	Added Barzahlen payment method and recurring credit card authorizations
Moritz Winter Neeme Oja	2.4	20.07.2016	Added Paydirekt payment method. Added Payment Method Information request.
Moritz Winter Neeme Oja	2.5.2	07.12.2016	Added transaction API and common locale parameter.
Neeme Oja	2.7	17.03.2017	Added new error: amount cannot be zero or negative. Added truncated credit card data in postbacks. Removed obsolete inline credit card features. Improved description of credit card form inside iframe.
Neeme Oja	2.8	14.06.2017	Added registration API call.
Neeme Oja Moritz Winter	2.9	23.06.2017	Added transaction summary call. Added message parameter in postbacks.
Neeme Oja	2.9.1	05.07.2017	Clarified postbacks

Table of Contents

1. Introduction

This is the API documentation and implementation guide for the Better Payment Gateway (UPG or Unified Payment Gateway).

Each Merchant will be handed an API key, outgoing key and incoming key. The Merchant should specify a unique *order_id* for each transaction. This order_id will be included in the response together with UPG's unique *transaction_id*.

The merchant is identified and authenticated by the API key and a checksum calculated with request parameters and the outgoing key. The API key and incoming/outgoing keys should be treated carefully and not revealed to outsiders.

Merchant-specific credentials used to be the same for test and production environments before the API version 2.0, but new implementations of web shops should make it possible to keep them separated.

To make requests use the following URLs:

- https://testapi.betterpayment.de for test requests.
- https://api.betterpayment.de for live requests.

The production API URLs should only be used for actual live payments.

The API accepts POST requests and returns response data in JSON format. Some payment methods involve redirecting the user to the acquirer's website, in which case the response includes instructions about whether to use GET or POST redirection and what special parameters to include. Payment methods that involve redirection require success and error URLs to be defined.

2. Payment Methods

The API offers a number of different payment methods, which are described in detail below.

The following table lists different features offered for each payment method. Additionally, a risk check procedure can be enabled for each payment method, although it is strongly recommended only for the Purchase on Account method.

	Payment	Refund	Recurring Payments	Authorize, Capture & Reverse	Precheck	Other

Credit Card	X	X	X	X		Register
Direct Debit	X	X[1]	X			Create Mandate Reference
Purchase on Account	X	X		X	X	
Purchase on Account (B2B)	X	X		X	X	
Installments	X	X		X	X	Calculate Rates
giropay	X	X				
SOFORT Überweisung	X					
PayPal	X	X	X			Register
Barzahlen	X	X				
Paydirekt	X	X				

2.1. Credit Card

When paying with credit card, the merchant's web shop redirects the user to the payment form hosted by Better Payment, where card data is entered. After completing the payment, the user is redirected back to the merchant's website. If the credit card payment requires 3D verification, Better Payment will take care of redirecting the user to the bank's website for extra verification steps, so the payment does not look different to the merchant in any way.

This method can also be used inside an iframe on the merchant's website, so the user never leaves the merchant's page. After completing the payment, the web page in the merchant's success/error url can include a JavaScript code snippet that breaks out of the iframe if needed.

2.2. SEPA Direct Debit

This payment method creates a SEPA direct debit transaction based on the IBAN and BIC codes given by the end user. Unless the merchant is generating custom SEPA mandate IDs, the mandate needs to be created first in a separate call.

[1] Not all Direct Debit Acquirers support refunds, please contact us for details.

2.3. Purchase on Account (B2C and B2B)

Purchase on Account (also known as Kauf auf Rechnung in Germany) offers users a possibility to pay the invoice after receiving the goods. To minimize the possibility of a default, invoice payment always includes a risk check procedure (more information under the Risk Check chapter). If the user does not pass the risk check, the API returns an error and the merchant is advised to offer other payment options to the user.

2.4. Installments

Installments make it possible for the customer to pay the sum in several monthly parts. First, an installment calculation is called, which returns several options to choose from. The final payment request should include a reference to the installment calculation.

2.5. giropay

Giropay is a service offered by GiroSolution that redirects the user to a website where they can complete the transaction by entering their bank account information and login credentials. The user confirms the transaction with their personal TAN code, after which the browser is redirected back to the merchant's website.

2.6. SOFORT Überweisung

SOFORT Überweisung is an online payment service which is offered by SOFORT AG. Technically it is very similar to giropay: the user is redirected to a payment site hosted by SOFORT and returns back to the merchant's site after completing the transaction. The main difference to giropay is that instead of POST redirection, SOFORT Überweisung redirects the user with the GET method to a URL specifically generated for the transaction.

2.7. PayPal

PayPal is an online payment service offered by PayPal Inc. Technically it is very similar to SOFORT: the user is redirected to a payment site hosted by PayPal and returns back to the Merchant's site after completing the transaction.

For recurring transactions PayPal Reference Transactions are used. This is a feature PayPal only makes available on explicit request by the customer.

2.8. Barzahlen

Barzahlen is a special payment method that generates invoice slips and allows customers to pay at Barzahlen retail partner stores by cash. The customer receives the invoice by email or SMS and can also view it when completing the order in the merchant's web shop. When the cash

payment is registered by a retail partner store, the merchant is notified of the completed payment via the postback URL.

2.9 Paydirekt

Paydirekt is a payment method offered by German Banks and Sparkassen. Technically it is very similar to SOFORT: the user is redirected to a payment site hosted by Paydirekt and returns back to the merchant's site after completing the transaction.

2.10 Risk Check

Risk check is another feature offered by Better Payment. It involves validating the user's address and identity (through name, date of birth and gender) as well as calculating the user's credit score. Risk check is optional and can be enabled for every payment method, but it is always required for Purchase on Account payments. Data used in the risk check is forwarded to the scoring provider. Users always have to be informed about the risk check and give their explicit approval.

For payment methods that have risk check enabled, the risk check is always performed first, before connecting to the acquirer with payment details. If the user does not pass the risk check, the API will return an error and the merchant should inform the user and suggest alternative payment methods.

3. Payment Request Parameters

The request address is https://api.betterpayment.de/rest/payment.

The payment request parameters are divided into the following data blocks. This table sums up which data blocks are needed for each payment method.

	Common Parameters	Billing Address	Shipping Address	Risk Check	Redirection URLs	Recurring (Optional)	Special Parameters
Credit Card	X	X			X	(X)	
Direct Debit	X	X		(X)		(X)	Debit Data
Purchase on Account	X	X	(X)	X			
Purchase on Account (B2B)	X	X	(X)	X			Company Data
Installments	X	X	(X)	X			

giropay	X	X			X		
SOFORT Überweisung	X	X			X		
PayPal	X	X			X	(X)	
Barzahlen	X	X					
Paydirekt	X	X			X		

3.1. Common Parameters

The following parameters are common to all payment requests. Note that the *amount* parameter used to be called *total_amount*; this parameter is still accepted by the API, but it is recommended to start to use the new name *amount* as soon as possible.

Parameter	**Required?**	**Comments**
payment_type	Yes	“cc” (for credit card) “dd” (for direct debit) “kar” (for invoice service) "kar_b2b" (for B2B invoices) "rate" (for installments) “giro” (for giropay) “sofort” (for SOFORT Überweisung) “paypal” (for PayPal) “bar” (for Barzahlen) “paydirekt” (for Paydirekt)
api_key	Yes	Merchant-specific API key
order_id	Yes	Any alphanumeric string to identify the Merchant’s order.
merchant _reference	No	What to display as the reference (Verwendungszweck) on the customer’s bank statement. Recommended value: <merchant order id> <shop name>. **NB:** Different payment methods impose different limits on the string length. Strings are converted to ASCII characters and truncated as needed. ● Direct Debit uses a limit of 140 characters (SEPA field: SVWZ). ● giropay uses a limit of 27 characters. ● SOFORT uses a limit of 54 characters that are split in two fields, each 27 characters long. Default value if not provided: merchant’s order id & merchant name (as defined in the merchant portal).
amount	Yes	Including possible shipping costs and VAT (decimal number, e.g. 1.23)
shipping_costs	No	Should be set if the order includes any shipping costs (decimal number, e.g. 1.23)

vat	No	VAT amount (decimal number) if known
currency	No	3-letter currency code (ISO 4217). Defaults to ‘EUR’
postback_url	Yes	The URL to updates about transaction status are posted. Unlike redirection URLs, this URL is required everywhere.
customer_ip	No	If the order includes a risk check, this field can be set to prevent customers from making multiple order attempts with different personal information.
customer_ip_proxy	No	Like above, but if the order comes from behind a proxy, this additional field can be used.
original_transaction_id	No	If a precheck or installment calculation has been made (in invoice and installment payment methods), this parameter should include the resulting transaction id from these previous actions.
duration	No	Used when selecting the result of an installment calculation.
locale	No	Can be used change the language of payment forms in credit card and Paypal payments. Currently supported locales are “en” and “de”.
checksum	Yes	A checksum of posted parameters and their values; see below for a complete explanation

3.2. Billing Address

Billing information is required in all payment methods.

Parameter	**Required?**	**Comments**
address	Yes	Street address
address2	No	Second address line
city	Yes	City
postal_code	Yes	Postal code
state	No	State
country	Yes	country code, e.g. “DE”
first_name	Yes	Customer’s first name
last_name	Yes	Customer’s last name
email	Yes	Customer’s email
phone	No	Customer’s phone number

3.3. Shipping Address

Shipping address can be specified when it differs from the billing address. It is required in invoice and installment payment methods.

Parameter	Required?	Comments
shipping_address	Yes	Street address
shipping_address2	No	Second address line
shipping_company	No	Company name
shipping_city	Yes	City
shipping_postal_code	Yes	Postal code
shipping_state	No	State
shipping_country	Yes	ISO 3166-1 country code, e.g. “DE”
shipping_first_name	No	First name
shipping_last_name	No	Last name

3.4. Company Details

Company details are required in B2B Invoice orders.

Parameter	Required?	Comments
company	Yes	Company name
company_vat_id	No	Company VAT ID
company_trade_register	No	Company trade registry no.

3.5. Risk Check Parameters

If the order includes a risk check (which is always the case with Purchase on Account), the following additional parameters are required.

Parameter	Required?	Comments
risk_check_approval	Yes	This field must be set to “1”. The customer must explicitly give his or her approval to risk check. If the value is anything else, an error is triggered.
date_of_birth	Yes	Customer’s date of birth; format “YYYY-MM-DD”
gender	Yes	“m” for male, “f” for female

3.6. Redirection URLs

For payment methods that involve browser redirection, these URLs are required. Note that postback_url works differently from the URLs below, because it does not use browser redirection.

Parameter	Required?	Comments
success_url	Yes	Where to redirect the user after a successful transaction.
error_url	Yes	Where to redirect the user after a failed transaction.

3.7. Direct Debit Parameters

The following additional parameters are required for a direct debit payment. If the optional sepa_mandate parameter is supplied, the create_mandate_reference request can be skipped when requesting a Direct Debit payment.

Parameter	Required?	Comments
iban	Yes	IBAN account number
bic	Yes	BIC code
account_holder	Yes	Bank account holder's name
sepa_mandate	No	Mandate Reference Number (max. 35 characters)

4. Recurring Payment Parameters

Some payment methods can use recurring payments. For credit card payments, authorizations can be recurring as well. Here is a summary of the recurring payment flow:

1. Create a regular payment with a special parameter:
   recurring=1
2. Store the transaction_id parameter from the response of the first payment request
3. Create any number of repeat payments with the following parameters:
   recurring=1
   original_transaction_id=<transaction_id from the first transaction>

Parameter	Required?	Comments
recurring	Yes	Set this to 1 in all recurring payments.
original_transaction_id	Yes	Leave empty for the first payment; set it to the transaction_id of the

	(except first time)	first payment in all repeated payments.

5. Signing Requests With Checksums

To prevent tampering with payment data, the parameters require signing with a checksum. Each merchant is assigned a private outgoing and incoming key. The API applies the following checksum calculation algorithm to verify the authenticity of the posted data.

Checksum algorithm:

- Make a key/value list of all used parameters (except the checksum itself), in the same order they are posted to the API request.
- Convert the parameters into a query string using the application/x-www-form-urlencoded format (RFC 1738). That is, spaces are encoded as plus (+) characters and special characters are encoded with a percentage sign followed by two hexadecimal digits. For instance, if the company parameter has the value “John & Sons”, the resulting string would be “company=John+%26+Sons”
- Append the outgoing key to the query string.
- Calculate an SHA1 digest of the resulting string.
- Set the value of the checksum parameter to the SHA1 digest.

Checksum Example for Credit Request

In this example, the outgoing key is `4d422da6fb8e3bb2749a` and the API key is `aab1fbbca555e0e70c27`. Note that for the sake of a simple demonstration, not all required parameters are included.

Data of the POST request parameters:

```
api_key=aab1fbbca555e0e70c27&currency=EUR&merchant_reference=123&order_id=123&payment_type=cc&shipping_costs=3.50&amount=17.50
```

The checksum is SHA1(query string + outgoing key):

```
9b6b075854fc3473c09700e20e19af3fbc3ff543
```

Example in PHP

```
$params = array(
      "api_key" => "aab1fbbca555e0e70c27",
      "currency" => "EUR",
      "merchant_reference" => "123",
      "order_id" => "123",
      "payment_type" => "cc",
      "shipping_costs" => "3.50",
      "amount" => "17.50");
```

```
$outgoing_key = "4d422da6fb8e3bb2749a";
$query = http_build_query($params, NULL, "&", PHP_QUERY_RFC3986);
$checksum = sha1($query . $outgoing_key);
printf($checksum);
```

```
9b6b075854fc3473c09700e20e19af3fbc3ff543
```

6. Payment Request Responses

All requests return responses in the JSON format. The following parameters are possible:

Parameter	**Comments**	
transaction_id	UPG's own transaction ID	
order_id	The order ID specified by the merchant	
status_code	Transaction's status code	
status	Transaction status in words for easier readability	
error_code	Error code; if this is 0, everything is fine.	
error_message	If the error code is other than 0, the error message will be included here.	
client_action	Possible values are: • "postform" — build a POST form from the data included in the action_data array and immediately submit it to the url included in action_data. • "redirect" — redirect to the url specified in action_data	
action_data	If client_action is specified, this array includes the following data:	
	Parameter	**Comments**
	url	Target url of the action
	fields	In case of a POST form, this array specifies the key/value pairs of the data to be posted.

6.1. Simple Payment Response

This is an example of a transaction that has been processed by the UPG API with no need for browser redirection. The merchant can run this API call in the back-end and display the end user an immediate result. In this example, an invoice request has passed risk check and has been marked as pending, because the user has yet to pay the bill.

```
{
  transaction_id: "a38ef37a-19ab-4f90-ae8d-2951be67fd40",
  order_id: "12345",
  status_code: 2,
```

```
  status: "pending"
}
```

6.2. Action Required: Form

This is an example of a response that instructs the merchant to build a form and immediately submit it. client_action specifies that there is a form that should be posted, and action_data includes the target url and fields with their values. This way the UPG API does not have to render the form, eliminating one link in the redirection chain. This action is currently only used by the giropay payment method.

```
{
  transaction_id: "09cb2327-9394-4ab4-be18-6e9d85e000fd",
  order_id: "123023",
  status_code: 1,
  status: "started",
  error_code: 0,
  client_action: "postform",
  action_data: {
    url: "https://payment.girosolution.de/payment/start",
    fields: {
      sourceId: "6da0f976433e18641662198d98557955"
      merchantId: "4432523",
      projectId: "3423",
      urlRedirect: "https://api.betterpayment.de/payment_response/7",
      urlNotify: "https://api.betterpayment.de/payment_postback/7",
      amount: "15.90",
      transactionId: "09cb2327-9394-4ab4-be18-6e9d85e000fd",
      vwz: "Order 123023 at Merchant",
      bankcode: "12345679",
      hash: "d9e2b54623855fbb2848bd8f9cc7ef65"
    }
  }
}
```

6.3. Action Required: Redirection

Some payment processors, such as SOFORT, generate a URL for each transaction, where the user can be redirected with no POST parameters required. In such cases, client_action is set to “redirect” and the target URL is included in action_data.

```
{
  transaction_id: "0b3c2327-9394-4ab4-be18-6e9d85e652fd",
  order_id: "1230238",
  status_code: 1,
  status: "started",
  error_code: 0,
  client_action: "redirect",
```

```
  action_data: {
                                                    url:
"https://www.sofort.com/payment/go/508712aa8572615d6151de6111"
  }
}
```

6.4. Error Response

If the API returns an error, the response will look like this.

```
{
  error_code: 101,
  error_message: "Merchant not found."
}
```

6.5. Return Values in URLs

The following data is defined in GET parameters when redirecting back to the shop's success/error URLs.

Parameter	Comments
order_id	The original order ID specified in the payment request
transaction_id	UPG transaction ID. This is useful when inquiring more information about a transaction and will be used in additional requests in the future.
checksum	A checksum calculated with the two parameters and the incoming key.

It is recommended to verify the authenticity of the data by validating the checksum. This checksum is calculated like with outgoing data, only the incoming key is used this time.

Checksum Example

The incoming key is `7b851aa07bb16788f05a`.

The query string of the two parameters (leaving out the checksum parameter) is

`order_id=123&transaction_id=4d13e292-c52c-4d3f-94d2-20740e30f68a`

The checksum is SHA1(query string + incoming key):

`e905ea2c47da74f8b5ab32c55edf821e3bbef250`

7. Precheck Request

The precheck request is a way to run the risk check before submitting the actual payment request. This should be done after the customer has selected the payment method that requires a risk check (e.g. Purchase on Invoice). The parameters for the precheck request are the same as

in the main payment request with a risk check, but URLs are not required. If the merchant's order_id is not known at the time of the precheck, it can be left blank.

If an installment calculation has been made before the precheck, the precheck request should include the selected duration and the original_transaction_id parameters. The result of a successful precheck request will have the status "registered".

The API endpoint for the precheck request is https://api.betterpayment.de/rest/precheck.

8. Authorize, Capture and Reverse Requests

8.1. Authorize

This request preauthorizes a transaction. With invoice payments, the parameters are identical to the actual payment request, and indeed the invoice payments can be completed using the payment request as an alias to authorization. The only exception to the parameter set is reauthorization, which does not require billing or shipping addresses, only new amounts and an original_transaction_id that points to a previously authorized transaction.

If a precheck request has been made before authorization, original_transaction_id parameter should be set the transaction id resulting from the precheck. The result of a successful invoice authorization request will have the status "pending". The result of a successful credit card authorization request will have the status “authorized”.

Note that like the payment request, authorize can return a client action in some cases. When authorizing credit cards, the end user has to input the card number, so the API returns a redirection action with a URL. When authorizing a recurring credit card payment that refers to a previous successful authorization, the transaction returns a successful response without redirection. The parameters for credit card authorization are identical to those of the payment request. Among other things, success, error and postback URLs are required.

The API endpoint for the authorization is https://api.betterpayment.de/rest/authorize.

8.2. Capture

This request captures an authorized transaction. It is possible to do partial captures, but the amount may not exceed the original transaction (if it does, the transaction must be reauthorized first, which may not be supported by all payment providers). The result of a successful capture will have the status "complete".

The API endpoint for the capture is https://api.betterpayment.de/rest/capture.

Parameter	Required?	Comments
api_key	Yes	Merchant's api key
transaction_id	Yes	ID of the transaction to be captured
amount	No	Amount, if different from original transaction's amount
vat	No	VAT of the captured amount, if known
checksum	Yes	Checksum of the parameters

8.3. Reverse

This request reverses (cancels) a previous authorization. The difference to the refund request is that at this point, no money has actually changed hands. Depending on the payment processor, it may be possible to make partial reversals. Successful full reversals will have the status "reversed" whereas partial reversals will stay at "pending" (because the remaining amount can still be captured).

The API endpoint for the reversal is https://api.betterpayment.de/rest/reverse.

Parameter	Required?	Comments
api_key	Yes	Merchant's api key
transaction_id	Yes	ID of the transaction to be reversed
amount	No	Amount, if different from original transaction's amount
vat	No	VAT of the reversed amount, if known
checksum	Yes	Checksum of the parameters

9. Registration Request

With some payment methods, it is possible to register payment information for later billing. This is often used with credit card payments. The difference to authorization is that authorization reserves an amount on the customer's account that has to be captured or voided, but registration simply stores the credit card data and returns the transaction ID as a token. This transaction ID can be used to make the actual transaction later.

Depending on the payment service provider, the system may try to make a small authorization on the credit card and immediately void it, to ensure that the card number is actually usable. Because this method involves the end user entering credit card data, it requires redirection.

Paypal preapproved payments may be possible, but require a separate agreement between the merchant and Paypal.

Here's a summary of the registration flow:

1. Call the registration endpoint.
2. Redirect the end user to the returned URL.
3. End user inputs credit card data and gets redirected back to Merchant's shop.
4. Transaction status should be "registered"; keep the transaction_id
5. Make a payment request with original_transaction_id=(id from the register call). Set recurring=1 if you intend to make several payments.

The API endpoint for registration is https://api.betterpayment.de/rest/register.

Parameter	Required?	Comments
payment_type	Yes	"cc" (for credit card) "paypal" (for PayPal)
api_key	Yes	Merchant-specific API key
success_url	Yes	Where to redirect the user after a successful registration.
error_url	Yes	Where to redirect the user after a failed registration.
postback_url	Yes	The URL to updates about transaction status are posted.
locale	No	Can be used change the language of payment forms in credit card and Paypal payments. Currently supported locales are "en" and "de".
checksum	Yes	A checksum of posted parameters and their values.

10. Installment Calculation Request

The installment calculation returns a selection of monthly payment options for the installment payment method. The id of the transaction and duration of the selected payment option should be included when making the actual payment, precheck or authorization.

Since installment calculation can also be triggered with JavaScript, this method does not require checksums. The api endpoint is https://api.betterpayment.de/rest/rate.

Parameter	Required?	Comments
payment_type	Yes	Set this to “rate”
api_key	Yes	Merchant’s api key
amount	Yes	Total amount of the transaction in decimal
vat	No	Amount of VAT; not required, but recommended if known

10.1. Installment Calculation Response

The following is an example response of the installment calculation. To save space, only two different duration options are included, but usually the API returns more.


```
{
  "transaction_id":"b60cebf1-43d8-4993-8d46-f3639be4207f",
  "status_code":1,
  "status":"started",
  "error_code":0,
  "installments":[
    { "installment": [
        { "amount":"67.17", "due":"2016-06-05" },
        { "amount":"67.17", "due":"2016-07-05" },
        { "amount":"67.17", "due":"2016-08-05" } ],
      "original_amount":"200.00",
      "total_amount":"201.51",
      "minimum_installment_fee":"0",
      "duration":"3",
      "interest_rate":"4.95",
      "effective_interest_rate":"5.09",
      "currency":"EUR"
    },
    { "installment": [
        { "amount":"33.79", "due":"2016-06-05" },
        { "amount":"33.79", "due":"2016-07-05" },
        { "amount":"33.79", "due":"2016-08-05" },
        { "amount":"33.79", "due":"2016-09-05" },
        { "amount":"33.79", "due":"2016-10-05" },
        { "amount":"33.79", "due":"2016-11-05" } ],
      "original_amount":"200.00",
      "total_amount":"202.74",
      "minimum_installment_fee":"0",
      "duration":"6",
      "interest_rate":"4.95",
```

```
      "effective_interest_rate":"5.03",
      "currency":"EUR"
    },
  ]
}
```

11. Refund Request Parameters

The address for payment refunds is https://api.betterpayment.de/rest/refund. There may be more than one refund per transaction, but the combined amount of refunds may not exceed the total amount of the transaction. For certain invoice/installment payment processors, it is possible to make a refund announcement (by setting announcement=1), which does not trigger the actual refund, but notifies the payment processor to stop dunning notifications (ask Better Payment for more details).

Parameter	Required?	Comments
api_key	Yes	Merchant-specific API key
transaction_id	Yes	ID of the transaction to be refunded
amount	Yes*	Amount to be refunded; can be smaller, but not larger, than the original amount. *This is required in all cases except refund announcements.
comment	No	Optional comment; for internal bookkeeping, not visible to the end user
announcement	No	Set this to 1 to make a refund announcement
checksum	Yes	A checksum of posted parameters and their values

11.1. Refund Response

```
{
  transaction_id: "0b3c2327-9394-4ab4-be18-6e9d85e652fd",
  refund_id: "6vrg2365-5232-rv27-ac2g-d852fd5e6e96",
  status_code: 1,
  status: "successful"
}
```

12. Create Mandate Reference

A Direct Debit/SEPA Payment relies on a mandate. This can be supplied via the optional sepa_mandate parameter or a mandate reference code has to be generated first.

This code must be communicated to the user/bank account holder and is the basis for mandates.

After the call to create_mandate_reference the Direct Debit payment request has to be made with the field original_transaction_id set to the transaction_id of the response.

12.1. Create Mandate Reference Call

The address for this parameter GET/POST is:

https://api.betterpayment.de/rest/create_mandate_reference

Parameter	Required?	Comments
api_key	Yes	Merchant-specific API key
payment_type	Yes	“dd” (for direct debit)

12.2. Create Mandate Reference Response

Parameter	Comments
transaction_id	UPG-specific ID of the transaction
status_code	Status code of the transaction
status	Transaction status in words
token	The Mandate Reference
order_id	Merchant’s order ID

13. SEPA payment legal implications

In order to conform with International and German Law information about the SEPA purchase needs to be shown to the paying customer in the case of a initial SEPA payment. Here is one example how such a information text might look like[2]:

> Mandate reference: <mandate reference number>
> I hereby legitimate <merchant name> to conduct payments via direct debit to settle outstanding amounts. I further instruct my bank to service direct debit payments initiated by <merchant name>.

[2] All examples provided for SEPA payment legal texts are not guaranteed to be compliant and we can't take any legal risks.

Surname, name: <surname>,<name>

Street, number: <Street, No.>

Postcode, city: <post code, city>

Holder name: <holder name>

IBAN: <iban>

BIC: <bic>

Notice: Within eight weeks time from the date of charge, you can demand a full refund of the charged amount. Please regard the terms and conditions for refunds of your bank, which apply.

<checkbox (required)> I hereby confirm that I have the authority to grant the mandate to the SEPA direct debit transaction(s) displayed above. I hereby grant the mandate.

Be aware that the text for a recurring mandate differs from this.

Additionally a customer must receive a pre notification if a recurring payment occurs, the text of the prenotification might look something like:

We debit your bank account with € 12.70 maturing March 13th 2015.

The customer must also be informed about the mandate number (not to be confused with mandate reference). This number can be shown after the payment has been processed, or be sent via eMail later.

14. Barzahlen Integration

When using Barzahlen as a payment method, the return data will include the checkout token parameter. To display the Barzahlen payment slip, the merchant should embed a simple HTML code snippet that loads the JavaScript hosted by Barzahlen. The slip will display the nearest Barzahlen locations as well as the option to have the slip sent to a phone number as an SMS message (emails get sent to the customer automatically). It will be automatically loaded in a modal dialog that the customer can close. Optionally, a button can be displayed that reopens the modal dialog after closing.

Successful Barzahlen purchases will be returned as “pending”, because the customer has not paid for the order yet. When a successful payment has been registered, the API makes a postback call and updates the transaction status to “complete”. If the slip expires before the customer pays, another postback call is made with the transaction status “reversed”. The expiration period depends on the merchant’s contract with Barzahlen.

Barzahlen transactions can be refunded, in which case the customer receives the refund slip by email automatically. When the refund slip is cashed, another postback call is made with the status as “refunded”.

14.1. Barzahlen Example Response

```
{
  transaction_id: "bd982c29-62f4-46de-8582-9f01cd7a076e",
  order_id: "123000",
  error_code: 0,
  status_code: 2,
  status: "pending",
  checkout_token: "djF8Y2hrdHxzbHAtYmEwNmMzNTEtZ....ZZ2ptM1hjL3M9"
}
```

14.2. Embedding Barzahlen Payment Slips

At the very bottom of the web shop page, insert the following code:

```
<script src="https://cdn.barzahlen.de/js/v2/checkout.js"
class="bz-checkout" data-token="[checkout token from the API
response]"></script>
```

For test versions, use the JavaScript URL

```
https://cdn.barzahlen.de/js/v2/checkout-sandbox.js
```

The modal dialog can be displayed again by embedding a <button> element whose class is set to “bz-checkout-btn”. For more information on Barzahlen integration, see the documentation at https://docs.barzahlen.de/api/v2/#setup.

15. Payment Method Information

Some payment methods, particularly the ones that use invoice and risk checks, need end user agreement. Since the legal disclaimers can vary between payment providers, there is a method to pull the disclaimer texts from Better Payment API. This request uses the GET method (that can be also called via JavaScript) and does not require checksums.

The address for card registration is https://api.betterpayment.de/rest/payment_info.

Parameter	Required?	Comments
api_key	Yes	Merchant's api key
payment_type	Yes	Shorthand code for the payment method (see the main payment request for details).
locale	No	Optional language flag for disclaimer texts ("de" or "en"). Note that some payment providers only support the German language.

15.1. Payment Method Information Response

Note that the content may include HTML code, such as links to the payment provider.

```
{
disclaimer: "I agree to the risk check performed by Firma AG. More
information on terms and conditions available <a
href='http://www.firma.de/terms'>here</a>."
}
```

16. Postbacks

On completing the transaction, the API will perform a POST request to the URL specified in postback_url before redirecting the user to the Merchant's website. The purpose of this additional request is to pass transaction information to the Merchant without the end user seeing the contents, and to ensure the Merchant is informed about completing the transaction even if the user for some reason does not return to the Merchant's website. The following data will be posted in all postback requests:

Parameter	Comments
transaction_id	UPG-specific ID of the transaction
status_code	Status code of the transaction; see below
status	Transaction status in words; see below
order_id	Merchant's order ID
checksum	Checksum of the above data using the incoming key
message	Additional info about the transactions status

Attention! The collection of parameters included in the postback message can change without notice. It will be backwards-compatible (i.e. nothing will be removed) but new features are introduced from time to time. Therefore, when verifying the postback contents against the merchant's incoming key, the set of parameters should not be fixed or whitelisted, but the calculation should always be made against the entire incoming data.

If the postback URL is not responding or responds with any status other than 200 OK, the API will attempt to re-post the data every ten minutes. The maximum number of attempts is 10. Note that if the transaction is successful or pending, the end user will be redirected back to the shop's success URL even if the postback URL is not responding.

16.1. Extra Parameters for Direct Debit Payment Postbacks

In case of direct debit payments, the following additional parameters will be included in the postback data. IBAN and BIC codes will be returned regardless of whether the original transaction used IBAN/BIC.

Parameter	Comments
mandate_id	The SEPA mandate ID
iban	The IBAN number of the transaction
bic	The BIC code of the transaction

16.2. Extra Parameters for Credit Card Payment Postbacks

After successful credit card payments or authorizations, the postback data includes truncated credit card data. This data can be displayed to the end user when reusing the card for recurring payments.

Parameter	Comments
card_last_four	Last four digits of the credit card number, e.g. 1881
card_expiry_year	Expiration year of the credit card, e.g. 8
card_expiry_month	Expiration month of the credit card, e.g. 2020
card_brand	Brand of the credit card (in caps, e.g. VISA, AMEX, MASTER)

16.3. Checksum Example

The incoming key is `7b851aa07bb16788f05a`.

The raw post string (leaving out the checksum parameter) is

```
transaction_id=4d13e292-c52c-4d3f-94d2-20740e30f68a&status_code=3&stat
us=complete&order_id=123
```

The checksum is SHA1(query string + incoming key):

```
7e544606ea146d9ecd0f6a2297e48a724ea50a7a
```

17. Transaction Statuses Explained

Code	Name	Description	Further Action
1	started	Transaction has been started; user has been redirected to the acquirer's website.	Transactions that remain in this state for longer than an hour should be considered aborted by the user.
2	pending	Credit card payments: The user submitted the payment, but the acquirer has marked the transaction as pending; this usually means that it takes longer than average to complete the transaction. Invoice payments: The order has been accepted, but payment has to be confirmed manually by the merchant or captured over the API (depending on the acquirer).	The pending status is treated as a successful payment; the API returns a successful state or redirects to the merchant's success URL
3	complete	Credit card payments: The acquirer has marked the transaction as completed. Invoice payments: the merchant marked pending transaction as paid in the merchant portal	Payment is successful; API returns a successful state or redirects to success URL
4	error	There has been an error with the transaction.	API returns an error state or redirects to the shop's error URL
5	canceled	User has canceled the payment on the acquirer's website.	API redirects to the error URL or returns a canceled status
6	declined	The acquirer has declined the payment. Payments that require a risk check are marked as declined if the risk check fails.	API returns a declined state or redirects to the shop's error URL
7	refunded	There are partial or full refunds for this transaction.	This status is usually not returned by the API but visible in the merchant portal after refunds have been made.
8	authorized	The payment has been authorized	This payment should be captured or

		successfully but not yet executed.	reversed later.
9	registered	A credit card has been registered for this transaction but payment has not been executed. OR: A successful precheck has been completed but the transaction is not yet authorized.	This transaction should be completed using the payment request.
10	debt collection	A debt collection process has been initiated for this transaction.	The transaction has gone into debt collection; awaiting further status updates.
11	debt paid	The debt collection process has been completed for this transaction.	The debt has been collected.
12	reversed	Authorization has been reversed (canceled).	
13	chargeback	A Chargeback has been issued for a completed or pending transaction.	

18. Transaction Requests

18.1. List of Transactions

A list of transactions can be requested through a GET request to

https://api.betterpayment.de/rest/transactions

The response includes the last 50 transactions. Optionally the parameters from and to can be used to specify a timeframe the transactions occurred in, the result of such a query returns all queries in the timespan. It is also possible to specify transaction status or statuses in a comma-separated list (see the chapter “Transaction Statuses Explained” for a list of status codes and their meanings).

Parameter	Required?	Comment
api_key	Yes	API key of the merchant
from	No	Optional earliest date in ISO-8601, e.g. 2016-10-05T17:50:28+02:00
to	No	Optional latest date in ISO-8601, e.g. 2016-10-05T18:50:28+02:00
status	No	Optional status code or a comma-separated list of status codes, e.g. status=3 or status=2,3
currency	No	Optional 3-letter currency code
checksum	Yes	Checksum of the sent parameters

An example request would look like this:

```
resp = conn.get("https://testapi.betterpayment.de/rest/transactions",

"api_key" => "81d345b3d68cbd51c9fe",

"from" => "2016-10-05T17:50:28+02:00",

"to" => "2016-10-05T18:50:28+02:00",

"checksum" => "d9f4e651f88121479d8c6cda4441ecba65687415")
```

A typical JSON response would look like this:

```
puts resp.body

=>
```

[{"transaction_id":
"f17f33b3-d439-4770-b2fe-562b5318d896","created_at":"2016-08-02T11:27:
19.053+02:00","status_code":1,"status":"started","amount":23.42,"curre
ncy":"EUR","order_id":"Foo Merchant Order
420","payment_method":"paypal"},

{"transaction_id":
"d18c33e3-k339-4471-a2ce-a6db5f1fd8c6","created_at":"2016-08-01T11:22:
15.053+02:00","status_code":2,"status":"pending","amount":0.23,"curren
cy":"EUR","order_id":"Foo Merchant Order 559","payment_method":"cc"}]

18.2. Transaction Summary

When the number of transactions is large, but going through individual transactions is not desired, it is possible to query statistics on transactions. This will return the total sum of transaction amounts and the number of transactions found in the data set. The url of this GET request is https://api.betterpayment.de/rest/transactions/summary

The parameters are the same as in the above transactions query. Note that for the total amount to make sense, the transactions should be in the same currency. Many merchants only use one currency, but if this is not the case, it is advised to limit the results to a specific currency using the currency parameter.

The result set will include two values: *count* and *total_amount*. Example:

{"count": 89, "total_amount": 3169.29}

18.3. Single Transaction

The details of a single transaction can be requested through a GET request to

https://api.betterpayment.de/rest/transactions/TRANSACTION_ID

Where the required parameter TRANSACTION_ID is an existing transaction_id.

```
resp =
conn.get("https://testapi.betterpayment.de/rest/transactions/f17f33b3-
d439-4770-b2fe-562b5318d896", "api_key" => "81d345b3d68cbd51c9fe",

"id" => "f17f33b3-d439-4770-b2fe-562b5318d896"
```

```
"checksum" => "d9f4e651f88121479d8c6cda4441ecba65687415")
```

A typical JSON response would look like:

```
puts resp.body

=>

[{"transaction_id":
"f17f33b3-d439-4770-b2fe-562b5318d896","created_at":"2016-08-02T11:27:
19.053+02:00","status_code":1,"status":"started","amount":23.42,"curre
ncy":"EUR","order_id":"Foo Merchant Order 23","payment_method":"cc"}]
```

19. Errors Explained

The API can return the following errors with explanation messages. Please note that these messages are not meant to be displayed to the end user because most of them are of very technical nature. Instead, the Merchant's system should decide what to do in each case (for instance, suggest a new payment method when the risk check fails). This list is still subject to change as features are added to the API.

Code	Message	Reason	Action
101	Merchant not found.	Merchant does not exist or was not recognized.	Check api_key.
102	Transaction not found.	Transaction does not exist. Rare; may occur when redirecting back from the acquirer.	Try again later; contact Better Payment.
103	The checksum does not match.	Either the calculation algorithm or the outgoing key is invalid.	Check the checksum calculation algorithm and the outgoing key. Pay particular attention not to confuse outgoing and incoming keys.
104	Unsupported payment type.	Payment method does not exist or merchant does not have this payment method.	Check payment_type.
105	(No longer used)	(No longer used)	(No longer used)
106	The payment processor is not responding.	Acquirer's service is not responding.	Try again later.
107	There has been an error with the payment processor.	Acquirer's service is not functioning and is returning errors.	Try again later.
108	Payment error.	Acquirer returned a payment-related error. Includes more information in the message if available.	Look into the error message; contact Better Payment.
109	Merchant does not have payment processor for this payment type.	No acquirer has been defined for the merchant.	Contact Better Payment.
110	Customer did not agree to the risk check process required by this payment method.	Customer must explicitly agree to the risk check process.	Check risk_check_approval; remind the customer that risk check must be approved.
111	Risk check processor not found.	System error; rare	Contact Better Payment.
112	Customer did not pass the risk check.	Customer had insufficient risk check score, wasn't identified by	Inform the customer and suggest alternative payment methods.

		the risk check service or the address does not exist.	
113	There has been an error with the risk check.	Error in the risk check processor's service.	Try again later; contact Better Payment.
114	Too many risk check attempts from this address.	Customer from the same IP has attempted to submit an order (and failed the risk check) multiple times, possibly with different personal/address data.	Suggest alternative payment methods.
115	Payment provider not found.	System error; rare	Contact Better Payment.
116	Risk check adapter not found.	System error; rare	Contact Better Payment.
117	Payment adapter not found.	System error; rare	Contact Better Payment.
118	Recurring payment could not find the original transaction.	You are trying to run a recurring transaction, but the original transaction was not found.	Check original_transaction_id.
119	This payment processor does not support recurring payments.	You are trying to make a recurring payment with an acquirer that doesn't support recurring payments.	Make a new non-recurring transaction.
120	Recurring payment could not find the original payment token.	The tokens needed to make a repeat transaction with the acquirer could not be found. Most likely the acquirer did not support recurring payments at the time of the original transaction.	Make a new non-recurring transaction.
121	This payment processor does not support refunds.	Refunds cannot be made for this acquirer.	If the transaction absolutely has to be refunded, contact Better Payment.
122	The refunded amount cannot exceed the original amount.	Combined amount of past refunds for this transaction is larger than the original amount.	Check the amount to be refunded.
123	This currency is not supported.	The selected currency is not supported by this acquirer.	Try another currency.
124	Invalid country code.	The country has not been specified in the correct format.	Make sure the country code is in the ISO 3166-1 format.
125	Invalid or missing return URLs.	No valid return URLs have been specified in the payment request.	Check the parameters success_url, error_url and postback_url.
126	Invalid bank account information.	The bank data of a direct debit transaction does not pass the validation.	Check IBAN and BIC.
127	This payment processor	The payment processor does not	Use *payment* call or contact Better

	does not support authorization.	support authorize/capture/reverse operations.	Payment.
128	Transaction has not been authorized for capture or reverse operation.	The status of transaction is other than “authorized”, that is, it has already been captured or reversed or was not authorized to begin with.	Create a new payment.
129	(No longer used)	(No longer used)	(No longer used)
130	Error converting legacy bank information to IBAN/BIC.	(No longer used, because legacy parameters are no longer accepted.)	Use IBAN/BIC instead.
131	This payment processor does not support Mandate generation.	SEPA Mandate generation cannot be used with this payment processor.	Use *payment* call or contact Better Payment.
132	Reserved	Reserved	
133	This payment processor does not support generic Postbacks.	The payment processor can't handle generic Postbacks and the Transaction will not be updated.	Contact Better Payment.
134	Amount cannot be zero or negative.	Payments with zero or negative amounts are rejected by all acquirers and therefore not accepted by Better Payment either.	Only send payments with an amount greater than zero.