QIWI Wallet

# QIWI Wallet Pull Payments API

Version 2.1

# Table of contents

# 1. Introduction

## 1.1. Purpose of the API

This API is to be implemented by a merchant/PSP (hereinafter as “merchant”) to support QIWI Wallet Pull Payments. Pull payments are those initiated from the merchant’s website, in contrast to Push payments that are initiated in QIWI Wallet interfaces – web, mobile, terminal, etc.

## 1.2. Things to Know About QIWI Wallet

QIWI Wallet is an online and mobile payment service in Russia and other countries. QIWI Wallet is available online, with mobile applications, and with QIWI payment kiosks.

QIWI Wallet uses mobile phone number as user ID. Passwords are sent by SMS.

Users can fund their payments with QIWI Wallet from the prepaid balance of QIWI Wallet account, from the mobile phone account prepaid balance, with Visa/MasterCard debit or credit card, or in cash with a QIWI kiosk.

QIWI Wallet server interacts with the merchant’s server over HTTP protocol.

Requests from the merchants to QIWI are sent in the format of the HTTP-request parameters. In response, the data is returned in one of two formats in accordance with the value of the "Accept" header, which is transmitted in the request:

- XML (value of the “Accept” header: “application/xml”, “text/xml”);
- JSON (value of the “Accept” header: “application/json”, “text/json”).

For requests to the merchant’s server the data is transmitted as “application/x-www-form-urlencoded” content type. Response must be in XML format.

To receive notifications merchant must whitelist following IP subnets:

- 91.232.230.0/23
- 79.142.16.0/20

For security purposes, all data transmitted to QIWI Servers is encrypted using SSL. Unencrypted HTTP requests are not supported. SSL might also be used for encryption of the requests to the merchant’s servers (it is possible to use self-generated certificates). Otherwise, it is possible to use simplified signature algorithm based on HMAC-SHA1 (for details see chapter “Authorization”). Merchant must check validity of QIWI Wallet’s server certificate.

## 2. QIWI Wallet Interface



### 2.1. Creating an Invoice

To create an invoice to the user's QIWI Wallet the merchant has to send PUT-request to the following URL:

https://w.qiwi.com/api/v2/prv/{prv_id}/bills/{bill_id}

where:

- **{prv_id}** – merchant's unique ID (numeric value).
- **{bill_id}** - unique invoice identifier generated by the merchant (any non-empty string of up to 200 characters).

Request parameters:

| Name | Value format | Regexp | Description | Example |
|---|---|---|---|---|
| **user** | String of the form "tel:{phone_number}", where {phone_number} - number in international format | ^tel:\+\d{1,15}$ | The QW user's ID, to whom the invoice is issued. It is the user's phone number with tel: prefix . | +79263745223 |
| **amount** | A positive number rounded up to 2 or 3 decimal places after the comma | ^\d+(.\d{0,3})?$ | The invoice amount. The rounding up method depends on the invoice currency. | 10.52 |
| **ccy** | three-letter abbreviation | ^[a-zA-Z]{3}$ | Currency identifier (Alpha-3 ISO 4217 code) | USD |
| **comment** | any text | ^\.{0,255}$ | Comment to the invoice | Order #1234 at hosting.com |
| **lifetime** | Date, up to the seconds | ^\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}$ | Date and time in ISO 8601 format up to which the invoice is available for payment. If the | 2012-12-25T14:30:00 |

| | | | | |
|---|---|---|---|---|
| | | | invoice is not paid by this date it will become void and will be assigned a final status. | |
| **pay_source** | “mobile”, “qw” | ^((mobile)\|(qw)){1}$ | (Optional)<br>If the value is “mobile” the user’s MNO balance will be used as a funding source. If the value is “qw” – any other funding source available in QW interface. | |
| **prv_name** | any text, not more than 100 symbols | | Merchant’s name which will be shown to the payer. | Game 1 |

The response will contain the result code and in case if the invoice was successfully created – all information about the invoice (for more details see server response ).

Example:

```
PUT /api/v2/prv/2042/bills/BILL-1
Accept: text/json
Authorization: Basic MjA0Mjp0ZXN0Cg==
Content-Type: application/x-www-form-urlencoded; charset=utf-8

user=tel%3A%2B79031234567&amount=10.0&ccy=RUB&comment=test&lifetime=2012-11-25T09%3A00%3A00
```

```
HTTP/1.1 200 OK
Content-Type: text/plain

{"response": {
 "result_code": 0,
 "bill": {
  "bill_id": "BILL-1",
  "amount": "10.00",
  "ccy": "RUB",
  "status": "waiting",
  "error": 0,
  "user": "tel:+79031234567",
  "comment": "test"
 }
}}
```

### *2.2. Requesting Invoice Status*

Merchant can request the current status of the invoice by sending GET-request to the following URL: https://w.qiwi.com/api/v2/prv/{prv_id}/bills/{bill_id}

where:

- **{prv_id}** - merchant's unique ID (numeric value).
- **{bill_id}** - unique invoice identifier generated by the merchant (any non-empty string of up to 200 characters).

There are no parameters for the request.

The response will contain the result code of the operation, and in case of successful execution – all information about the invoice (for more details see server response ).

Example:

```
GET /api/v2/prv/2042/bills/BILL-1
Accept: text/json
Authorization: Basic MjA0Mjp0ZXN0Cg==
Content-Type: application/x-www-form-urlencoded; charset=utf-8
```

```
HTTP/1.1 200 OK
Content-Type: text/plain

{"response": {
  "result_code": 0,
  "bill": {
    "bill_id": "BILL-1",
    "amount": "10.00",
    "ccy": "RUB",
    "status": "waiting",
    "error": 0,
    "user": "tel:+79031234567",
    "comment": "test"
  }
}}
```

### *2.3. Redirection for Invoice Payment*

Merchant has to offer a QIWI Wallet user to immediately pay the invoice by redirecting him to the QIWI Wallet payment page or by opening it in iframe to one of the following URLs:

- https://w.qiwi.com/order/external/main.action?shop=xxxx&transaction=yyyyyyyy – to redirect
- https://w.qiwi.com/order/external/main.action?shop=xxxx&transaction=yyyyyyyy&iframe=true – to use iframe

The two URLs work exactly in the same way except that the second one appears more compact and can be placed conveniently within the merchant's page.

Both URLs accept the same set of parameters

| Name | Type | Description | Example |
|---|---|---|---|
| **shop** | string | Merchant's ID in QIWI Wallet system, corresponds to **{prv_id}** parameter used to create the bill | 123 |
| **transaction** | string | Invoice ID generated by the merchant, corresponds to **{bill_id}** parameter used to create the bill | abcde12345 |
| **successUrl** | URL-encoded string | The URL to which the payer will be redirected in case of successful payment | http%3A%2F%2Fmystore.com%2Fsuccess%3Fa%3D1%26b%3D2 |
| **failUrl** | URL-encoded string | The URL to which the payer will be redirected in case of unsuccessful payment | http%3A%2F%2Fmystore.com%2Ffail%3Fa%3D1%26b%3D2 |

QIWI Wallet will redirect back to **successUrl** or **failUrl** with an additional parameter. The name of the parameter is ***order*** and the value is the invoice ID from the **transaction** parameter of the initial request. Using this parameter, merchant can render the final page depending on the order details.

Example:

- Merchant redirects to URL: https://w.qiwi.com/order/external/main.action?shop=2042&transaction=123123123&successUrl=http://mystore.com/success?a=1&b=2&failUrl=http://mystore.com/fail?a=1&b=2
- User pays the invoice using one of the available payment options;
- QIWI Wallet redirects user to http://mystore.com/success?a=1&b=2&order=1234567.

**Warning!** Don't rely on redirection to successUrl as an evidence of successful payment, because it can be simulated at the user's end. It serves only as a convenient interface solution. Wait for the QIWI Wallet's notification to the merchant's site

### *2.4. Cancelling Unpaid Invoices*

Merchant can cancel a previously issued invoice by sending a PATCH-request to the following URL (provided that the invoice has not been paid):

https://w.qiwi.com/api/v2/prv/{prv_id}/bills/{bill_id}

where:

- **{prv_id}** - merchant's respective unique ID (numeric value).
- **{bill_id}** - unique invoice identifier generated by the merchant (any non-empty string of up to 200 characters)

| Name | Value format | Regexp | Description |
|---|---|---|---|
| **status** | “rejected” | ^rejected$ | New invoice status (cancelled) |

The response will contain the result code of the operation, and in case of successful execution – all information about the invoice (for more details see server response ).

```
PATCH /api/v2/prv/2042/bills/BILL-2
Accept: text/json
Authorization: Basic MjA0Mjp0ZXN0Cg==
Content-Type: application/x-www-form-urlencoded; charset=utf-8

status=rejected
```

```
HTTP/1.1 200 OK
Content-Type: text/plain

{"response": {
  "result_code": 0,
  "bill": {
    "bill_id": "BILL-2",
    "amount": "10.00",
    "ccy": "RUB",
    "status": "rejected",
    "error": 0,
    "user": "tel:+79031234567",
    "comment": "test"
  }
}}
```

### *2.5. Refunds*

Merchant can process a full or partial refund using the PUT-request described below. This request creates a reversed transaction for the initial one. In case if the transmitted amount exceeds the initial invoice amount or the amount left after the previous refunds, the remaining amount will be refunded.

To process this request, merchant has to send PUT-request to the following URL: https://w.qiwi.com/api/v2/prv/{prv_id}/bills/{bill_id}/refund/{refund_id}

where:

- **{prv_id}** - merchant’s respective unique ID (numeric value).
- **{bill_id}** - unique invoice identifier generated by the merchant (any non-empty string of up to 200 characters).
- **{refund_id} –** refund identifier, unique number of refund specific to the invoice **{bill_id}**.

Request parameters:

| Name | Value format | Regexp | Description |
|---|---|---|---|
| **amount** | A positive number rounded up to have 2 or 3 decimal places after the comma | ^\d+(\.\d{0,3})?$ | The refund amount should be less or equal to the amount of the initial transaction. The rounding up method depends on the invoice currency. |

The response will contain the result code of the operation, and in case of successful execution – all information about the invoice (for more details see server response ).

Example:

```
PUT /api/v2/prv/2042/bills/BILL-1/refund/1
Accept: text/json
Authorization: Basic MjA0Mjp0ZXN0Cg==
Content-Type: application/x-www-form-urlencoded; charset=utf-8

amount=5.0
```

```
HTTP/1.1 200 OK
Content-Type: text/plain

{"response": {
  "result_code": 0,
  "refund": {
    "refund_id": 1,
    "amount": "5.00",
    "status": "success",
    "error": 0
  }
}}
```

### *2.6. Refund Status Verification*

Merchant can verify the status of the refund by sending GET-request to the following URL: https://w.qiwi.com/api/v2/prv/{prv_id}/bills/{bill_id}/refund/{refund_id}

Where:

- **{prv_id}** - merchant's respective unique ID (numeric value).
- **{bill_id}** - unique invoice identifier generated by the merchant (any non-empty string of up to 200 characters).
- **{refund_id} –** refund identifier, unique number for refunds processed for a particular invoice.

There are no parameters for the request.

The response will contain the result code of the operation, and in case of successful execution – all information about the invoice (for more details see server response ).

Example:

```
GET /api/v2/prv/2042/bills/BILL-1/refund/1
Accept: text/json
Authorization: Basic MjA0Mjp0ZXN0Cg==
Content-Type: application/x-www-form-urlencoded; charset=utf-8
```

```
HTTP/1.1 200 OK
Content-Type: text/plain

{"response": {
 "result_code": 0,
 "refund": {
  "refund_id": 1,
  "amount": "5.00",
  "status": "success",
  "error": 0
 }
}}
```

### *2.7. Server Response*

The server response represents an object “Response”, which is composed of “Result code” element and one of the following objects depending on the request type: ”Invoice”, ”Refund”. The result is serialized in XML or JSON:

***The result code of an operation***

| Name | Value format | Regexp | Description |
|---|---|---|---|
| **result_code** | Integer from 0 to 5000 | ^\d{1,4}$ | Error code received after the operation execution (see error codes ) |

Example of serialization in XML:

```
…
<result_code>0</result_code>
…
```

Example of serialization in JSON:

```
…
“result_code”: 0,
…
```

***Information on invoice***

Parameters of the invoice:

| Name | Value format | Regexp | Description |
|---|---|---|---|
| **bill_id** | Any non-empty string up to 200 characters | ^.{1,200}$ | Unique invoice identifier generated by the merchant |
| **amount** | A positive number rounded up to have 2 or 3 decimal places after the comma | ^\d+(\.\d{0,3})?$ | The refund amount. The rounding up method depends on the invoice currency |
| **ccy** | three-letter abbreviation | ^[a-zA-Z]{3}$ | Currency identifier (Alpha-3 ISO 4217 code) |
| **status** | Alphanumeric identifier | ^[a-z]{1,15}$ | Invoice status (see Invoice status ) |
| **error** | Integer from 0 to 5000 | ^\d{1,5}$ | Error code (see error codes ) |
| **user** | String of the form “tel:{phone_number}”, where {phone_number} - number in international format | ^tel:\+\d{1,15}$ | The QW user’s ID, to whom the invoice is issued. It is the user’s phone number with tel: prefix . |
| **comment** | Any text | ^\.{0,255}$ | Comment to the invoice |

Example of XML serialization:

```
…
<bill>
<bill_id>bill1234</bill_id>
<amount>99.95</amount>
<ccy>USD</ccy>
<status>paid<status>
<error>0</error>
<user>tel:+79161231212</user>
<comment>Invoice from ShopName</comment>
</bill>
…
```

Example of JSON serialization:

```
…
“bill”: {
 “bill_id”: “bill1234”,
 “amount”: “99.95”,
 “ccy”: “USD”,
 “status”: “paid”,
 “error”: 0,
 “user”: “tel:+79161231212”,
 “comment”: “Invoice ffrom ShopName”
}
…
```

***Information on refund***

Parameters of the refund:

| Name | Value format | Regexp | Description |
|---|---|---|---|
| **refund_id** | Integer up to 4 numbers. | ^\d{1,4}$ | The refund identifier, unique number for refunds processed for a particular invoice. |
| **amount** | A positive number rounded up to have 2 or 3 decimal places after the comma | ^\d+(\.\d{0,3})?$ | The actual amount of the refund |
| **status** | alphanumeric identifier | ^\[a-z]{1,15}$ | Refund status (see Payment statuses ) |
| **error** | Integer from 0 to 5000 | ^\d{1,4}$ | Error code (see error codes ) |
| **user** | String of the form “tel:{phone_number}”,<br>where {phone_number} - number in international format | ^tel:\+\d{1,15}$ | The QW user’s ID, to whom the invoice is issued. It is the user’s phone number with tel: prefix . |

Example of XML serialization:

```
…
<refund>
<refund_id>12</refund_id>
<amount>99.95</amount>
<status>success<status>
<error>0</error>
<user>tel:+79161231212</user>
</refund>
…
```

Example of JSON serialization:

```
…
“refund”: {
 “refund_id”: “12”,
 “amount”: “99.95”,
 “status”: “success”,
 “error”: 0,
 “user”: “tel:+79161231212”
}
…
```

## 3. Merchant interface

This API enables merchant to receive the notifications on invoice status change. To receive these notifications merchant's server should be able to accept and process POST-requests with object "Invoice " containing all relevant data, serialized in the form of HTTP-request parameters + parameter "command", which always has "bill" value (application/x-www-form-urlencoded). In response to the request the result code will be returned (see Appendices). The response should be in XML.

Example request to merchant's server:

```
POST /qiwi-notify.php HTTP/1.1
Accept: application/xml
Content-type: application/x-www-form-urlencoded
Authorization: Basic MjA0Mjp0ZXN0Cg==

bill_id=BILL-
1&status=paid&error=0&amount=1.00&user=tel%3A%2B79031811737&prv_name=TEST&ccy=RUB&co
mment=test&command=bill
```

```
HTTP/1.1 200 OK
Content-Type: text/xml

<?xml version="1.0"?> <result><result_code>0</result_code></result>
```

Any result code other than 0 ("Success"), will be perceived by the QIWI server as a temporary error, thus the server will continue repeating requests with increased time intervals within next 24 hours (50 attempts in total) until it gets code 0 ("Success"). If the code 0 ("Success") has not been received within 24 hours the QIWI server will stop sending the requests and will send an email to the merchant with new Invoice status code and indication on the possible technical issues on the merchant's server side.

Because there could be several notification attempts for one invoice merchant must not deposit money to user or provide service twice.

In order to help in identifying the reasons of notification errors we recommend that the result codes returned by the merchant are in accordance with Notification codes table.

## 4. Authorization

Upon registration the merchant will receive an additional password which should be used for authorization when sending requests to QIWI and when receiving responses from QIWI to the merchant's server.

### 4.1. Authorization when sending requests to QIWI server

The password to the API is verified during the authorization. The password is transmitted using the standard rules of basic-authorization for HTTP-requests:

The header "Authorization" is added to the request. The value of this parameter is composed of the word "Basic", blank character and encrypted BASE64 pair login:password, where login – merchant's unique identifier, password – password to QIWI Wallet API.

```
Authorization: Basic bG9naW46cGFzc3dvcmQ=
```

BASE64("login:password") = "bG9naW46cGFzc3dvcmQ="

The transmission of the password is secured as all requests to QIWI are SSL-encrypted.

### 4.2. Authorization when receiving notifications on the merchant's server

It is preferred that the merchant's server is enabled with SSL-encryption. In this case the password transmission is secured. If the SSL-certificate is self-generated and is not issued by one of the standard certification centers, this certificate can be uploaded to the QIWI's server via the merchant's console ("Merchant settings" section of the website). Then the certificate becomes trusted.

The authorization is performed by one of the following ways:

1. Plaintext password transmission

   The pair login:password is transmitted according to the rules of basic-authorization and is being verified on the merchant's server. If the request is not SSL-encrypted, the transmission is not secured and the data becomes vulnerable to interception.

2. Simplified signature algorithm

   The header "X-Api-Signature" is added to the HTTP-request. The value of the header is composed of the HMAC-SHA1 function using special secret key as a key and all invoice parameters placed in alphabetical order and separated by "|" as message:

```
HMAC-SHA1(key, {amount}|{bill_id}|{ccy}|{command}|{comment}|{error}|{prv_name}|{status}|{user})
```

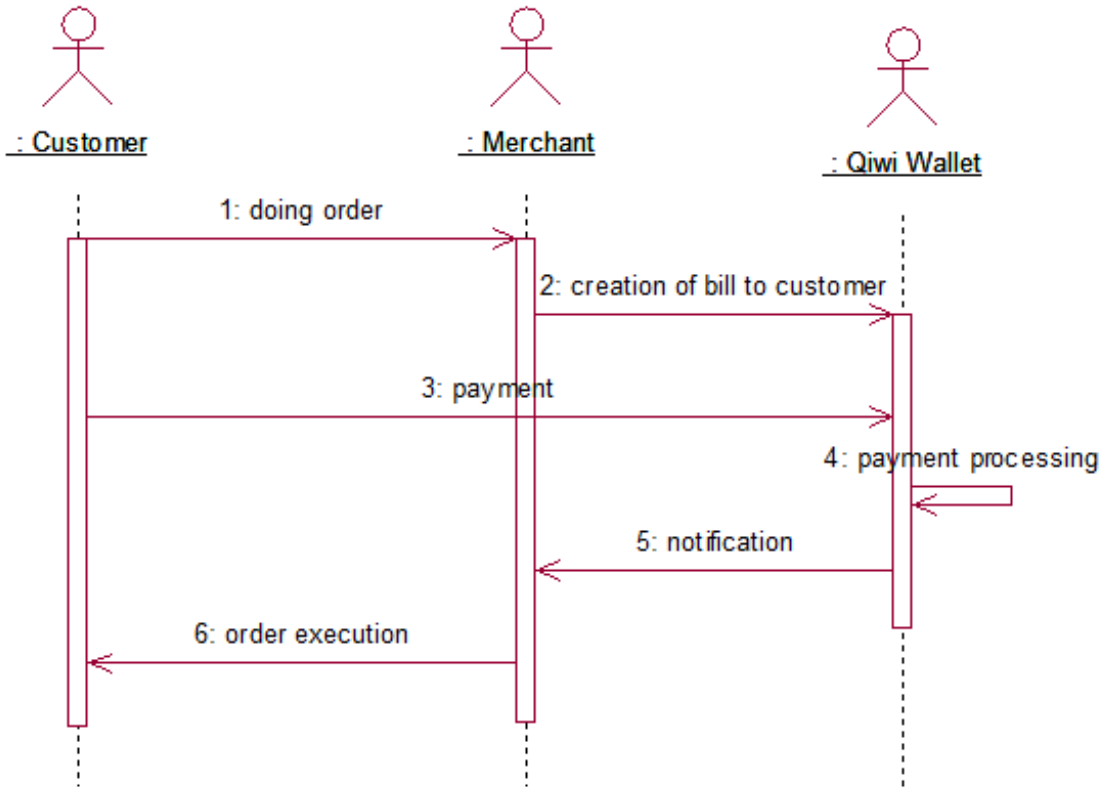
**password –** password used to access QIWI Wallet API;

**{parameter} –** value of the respective parameter of the invoice taken from the notification;

### *5. Typical interaction flow*

User submits an order on the merchant's website and then merchant sends request for creation of invoice to the QIWI's server. If successful, the user has to log in into the system using one of the QIWI Wallet interfaces and pay the invoice.

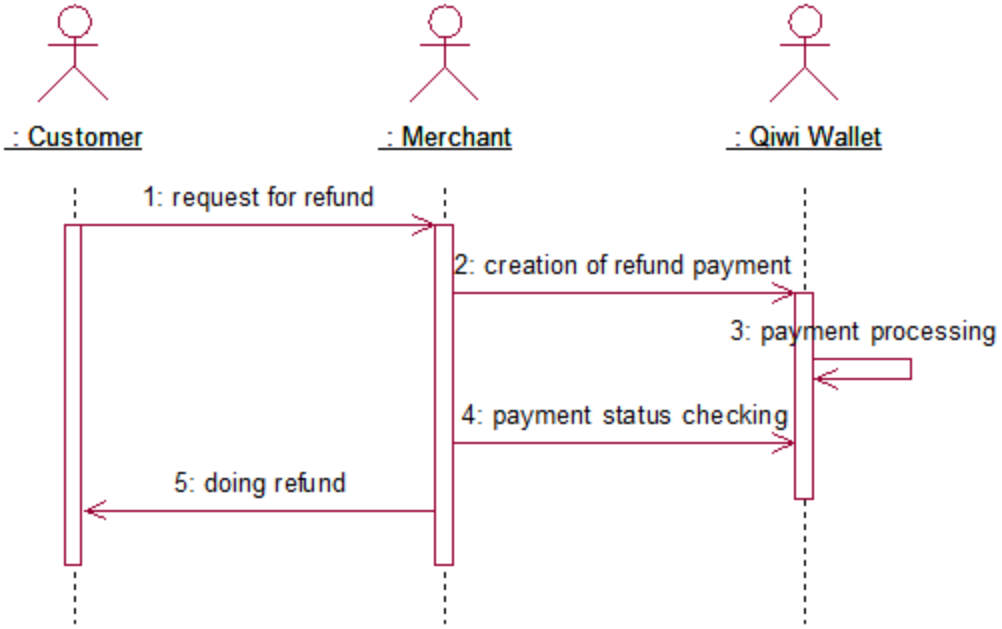
Once the invoice is created, QIWI will send a notification on its status: either paid or cancelled by the user. Merchant delivers services/goods when the invoice gets paid.




Merchant can request the status of the created invoice , or cancel it (provided that it has not been paid yet) at any moment.

If it is necessary to refund a part of the invoice amount or the full amount, merchant has to send a request for refund .

To make sure that the payment has been successfully processed merchant can periodically request the invoice status until the final status is received.




This scenario can be repeated multiple times until the invoice is completely refunded (whole invoice amount has been returned to the user)

## 6. Appendices

### 6.1. Invoice status

| Status code | Description | Final status? |
|---|---|---|
| **waiting** | Invoice issued, pending payment | No |
| **paid** | Invoice has been paid. | Yes |
| **rejected** | Invoice has been rejected. | Yes |
| **unpaid** | Payment processing error. Invoice has not been paid. | Yes |
| **expired** | Invoice expired. Invoice has not been paid. | Yes |

### 6.2. Payment status

| Status code | Description | Final status? |
|---|---|---|
| **processing** | Payment is pending | No |
| **success** | Payment is successful | Yes |
| **fail** | Payment is unsuccessful | Yes |

### 6.3. Error codes

| Error code | Description | Fatal? * |
|---|---|---|
| **0** | Success | No |
| **5** | The format of the request parameters is incorrect | Yes |
| **13** | Server is busy, try again later | No |
| **150** | Authorization error (e.g. invalid login/password) | |
| **210** | Invoice not found | Yes |
| **215** | Invoice with this bill_id already exists | Yes |
| **241** | Invoice amount less than minimum | Yes |
| **242** | Invoice amount greater than maximum (15 000 RUB) | Yes |
| **298** | User not registered | Yes |
| **300** | Technical error | No |

* Fatal means the result will not change with the second and subsequent requests (error is not temporary)

### *6.4. Notification codes*

| Код завершения | Description |
|---|---|
| **0** | Success |
| **5** | The format of the request parameters is incorrect |
| **13** | Database connection error |
| **150** | Incorrect password |
| **151** | Signature authorization failed |
| **300** | Server connection error |