VISA QIWI WALLET PULL PAYMENTS REST API

ver. 2.1

USER GUIDE

ver. 2.12

MOSCOW
+7-495-783-5959

RUSSIA
+7-800-200-0059

FAX
+7-495-926-4619

WEB
WWW.QIWI.COM

CONTENT

1. LIST OF CHANGES

Version	Date	Changes
2.12	July 11, 2016	Added "pay_source" parameter to redirection request for invoice payment Redirection URL changed to bill.qiwi.com
2.11	March 28, 2016	Added example of PHP script to check digital signature in merchant notification request
2.10	November 30, 2015	Changed Visa QIWI Wallet API URL from w.qiwi.com to api.qiwi.com
2.9	August 27, 2015	HTTP-protocol must be enabled to use redirection (chapter 4.3) Added new error code
2.8	May 27, 2015	Changed Visa QIWI Wallet URL qiwi.com to w.qiwi.com due to network policy Refund identifier format changed (chapters 4.6, 4.7, 4.8.3)
2.7	November 25, 2014	Document title is changed
2.6	November 12, 2014	Authorization chapter 2.1 moves to chapter 4.1 Authorization chapter 2.2 moves to chapter 5.1
2.5	October 27, 2014	Added new error codes
2.4	August 20, 2014	Changed Visa QIWI Wallet URL w.qiwi.com to qiwi.com
2.3	July 29, 2014	Added a clarification: • HTTP 200 is required for notification response
2.2	May 27, 2014	Added new error code on excessive refund amount
2.1	July 30, 2013	Added some clarifications: • Empty "pay_source" handling • Merchant's server client certificate should be in DER or PEM format
2.0	November 22, 2012	HMAC signature can be used for notification authorization
1.0	October 23, 2012	Initial version

2. INTRODUCTION

2.1. Purpose of the API

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

2.2. Things to Know About Visa QIWI Wallet

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

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

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

Visa 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 encoded by UTF-8. In response, the data is returned in one of two formats in accordance with the value of the "Accept" HTTP 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 notifications to the merchant's server the data is transmitted as "application/x-www-form-urlencoded" content type encoded by UTF-8. Parameters, transmitted in the URL string, must be URL-encoded. Response must be in XML format.

To receive notifications merchant must whitelist following IP subnets connected by 80, 443 ports exclusively:

- 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. Authorization on Visa QIWI Wallet side is performed by login and password for API access (for details see chapter "Authorization on Visa QIWI Wallet Server").

SSL might also be used for encryption of notifications 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 on Merchant's Server").

Merchant must check validity of Visa QIWI Wallet's server certificate using standard algorithm for certificate validation.

3. TYPICAL INTERACTION FLOW

3.1. Invoice Issue

WARNING

All requests to Visa QIWI Wallet server require authorization.

User submits an order on the merchant's website. Then merchant sends invoice request to Visa QIWI Wallet server (Fig. 1).

Merchant is recommended to redirect to checkout page on Visa QIWI Wallet site when request is completed.

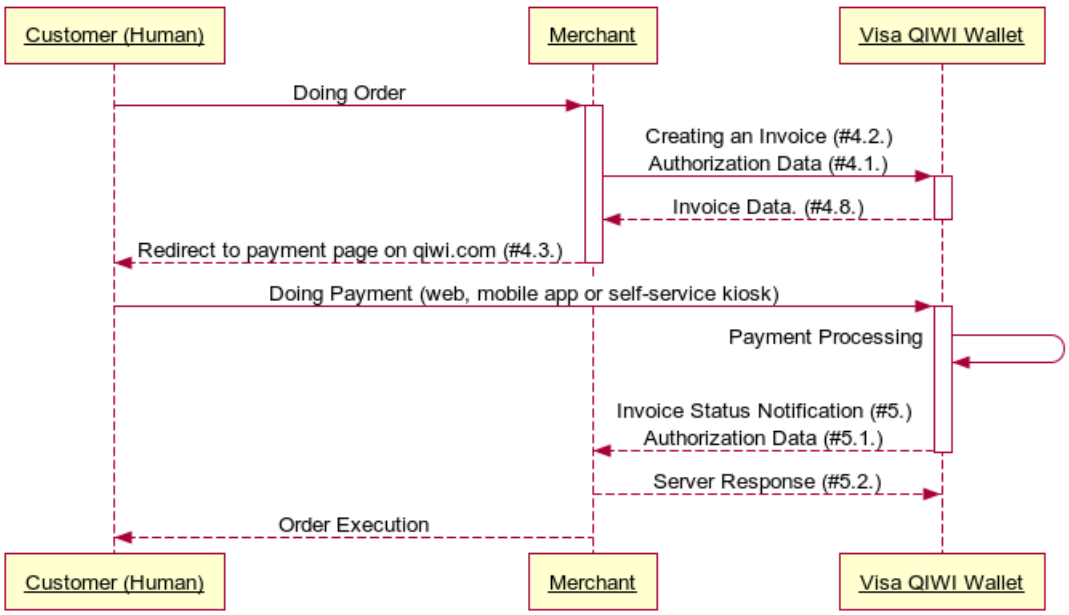
If merchant enables notifications, then Visa QIWI Wallet sends to the merchant's server a notification on the invoice status once invoice is paid or cancelled by the user. Authorization on the merchant's side is required for notifications.

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

Once the invoice is created, user should log on to Visa QIWI Wallet and pay the invoice.

Finally, merchant delivers ordered services/goods when the invoice gets paid.

Fig. 1. Invoice issuing scenario diagram



3.2. Refund Scenario

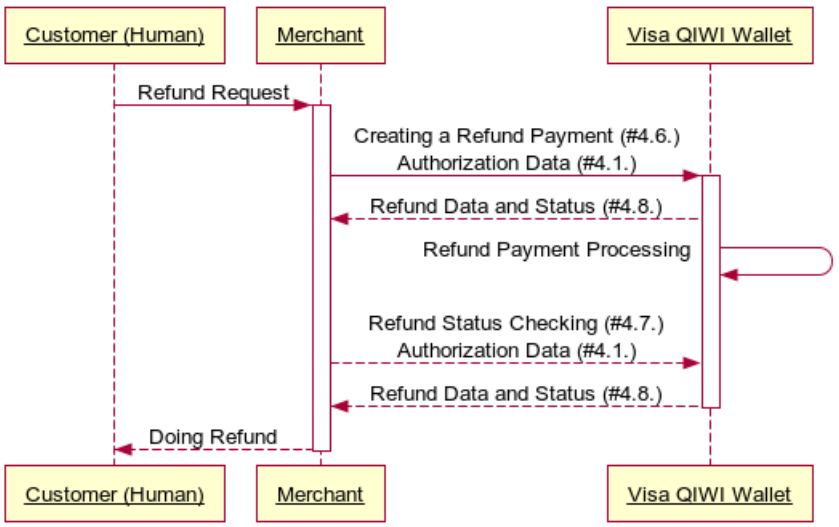
WARNING

All requests to Visa QIWI Wallet server require authorization.

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 Visa QIWI Wallet server (Fig. 2).

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

Fig. 2. Refund scenario diagram



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

4. VISA QIWI WALLET INTERFACE

4.1. Request Authorization

All requests transmitted to Visa QIWI Wallet server should be encrypted using SSL. Authorization on Visa QIWI Wallet side is performed by login and password for API access.

Upon registration the merchant will receive password which should be used for authentication. When generated on ishop.qiwi.com site, the password is linked to specific identifier for Visa QIWI Wallet API (API ID).

The "API ID: API password" couple is used for the merchant's authentication, where:

- API ID – merchant's shop unique identifier for Visa QIWI Wallet API, as displayed in **API ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site;
- API password – Visa QIWI Wallet API password corresponding to this API ID.

NOTE

The merchant can fully control authentication data by generating new password and API ID couples as well as removing old ones. New API password and API ID can be generated in **Protocols details → REST-protocol → Authentication details** section of ishop.qiwi.com web site by clicking the Generate new ID link.

Authentication data are transmitted using the standard rules of basic-authorization for HTTP-requests. The HTTP header "Authorization" is added to the request. The value of this parameter is composed of the word "Basic", blank character and encrypted BASE64 pair "`API ID:API password`":

```
Authorization: Basic ***
```

Where:

```
BASE64("API ID:API password") = "***"
```

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

Example of using authorization data is given in PHP script for invoice issuing.

4.2. Creating an Invoice

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

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

where:

- **{prv_id}** – merchant's Shop ID (numeric value, as displayed in **Shop ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site);
- **{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
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. **Required.**
ccy	Three-letter abbreviation	^[a-zA-Z]{3}$	Invoice currency identifier (Alpha-3 ISO 4217 code). **Required.**
comment	Any text up to 255 symbols	^\.{0,255}$	Comment to the invoice. **Required.**
user	String of the form "tel:phone_number", where "phone_number" – wireless phone number in international format	^tel:\+\d{1,15}$	The Visa QIWI Wallet user's ID, to whom the invoice is issued. It is the user's phone number with "tel:" prefix. **Required.**
lifetime	Date/time, up to the seconds, in ISO 8601 format (YYYY-MM-DD'T' hh:mm:ss) **Important!** Time is specified in Moscow time zone	^\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}$	Date and time up to which the invoice is available for payment. If the invoice is not paid by this date it will become void and will be assigned a final status. **Required.** **Important!** Invoice will be automatically expired when 45 days is passed after the invoicing date
pay_source	"mobile", "qw"	^((mobile)|(qw)){1}$	(optional) 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 is used available in Visa QIWI Wallet interface. If parameter isn't present, value "qw" is assumed
prv_name	Any text, not more than 100 symbols	^\.{1,100}$	(optional) Merchant's name.

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

NOTE

When you send two consecutive requests with the same {prv_id}, {bill_id} and "amount" parameters, you will get the same result code.

Sample request (JSON response):

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

user=tel%3A%2B79031234567%26amount=10.0%26ccy=RUB%26comment=Order #1234 at
hosting.com%26lifetime=2012-11-25T09%3A00%3A00
```

Response:

```
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": "Order #1234 at hosting.com"
  }
}}
```

4.3. Redirection for Invoice Payment

Merchant may offer a Visa QIWI Wallet user to pay the invoice immediately by redirecting to the Visa QIWI Wallet checkout page. To redirect, HTTP GET-request is used to the following URL:

https://bill.qiwi.com/order/external/main.action

WARNING

To use redirection, make sure that HTTP-protocol is enabled (**HTTP-protocol settings** flag is On) for the specific merchant (project) on ishop.qiwi.com in **Protocols details → HTTP-protocol** page.

In response, server displays invoice checkout page on Visa QIWI Wallet website. Page contains a number of ways to pay the invoice.

GET-request accepts the following set of parameters:

Name	Type	Description	Example
shop	string	Merchant's ID in Visa QIWI Wallet system, corresponds to {prv_id} parameter used to create the bill. **Required.**	123

Name	Type	Description	Example
transaction	string	Invoice ID generated by the merchant, corresponds to {bill_id} parameter used to create the bill. **Required.**	abcde12345
iframe	string	(optional) This parameter (if true) means that invoice page would be opened in "iframe". The checkout page appears more compact and can be embedded conveniently within the merchant's site.	true
successUrl	URL-encoded string	(optional) The URL to which the payer will be redirected in case of successful creation of Visa QIWI Wallet transaction. **URL must be within merchant's site**	http%3A%2F%2Fmystore.com%2Fsuccess%3Fa%3D1%26b%3D2
failUrl	URL-encoded string	(optional) The URL to which the payer will be redirected when creation of Visa QIWI Wallet transaction is unsuccessful. **URL must be within merchant's site**	http%3A%2F%2Fmystore.com%2Ffail%3Fa%3D1%26b%3D2
target	"iframe" or empty	(optional) This parameter means that hyperlink specified in *successUrl* / *failUrl* parameter opens in "iframe" page	
pay_source	String "mobile", "qw", "card", "wm", "ssk"	(optional) Default payment method to show first for the client. Possible values: • qw – Visa QIWI Wallet account; • mobile – client's cell phone account; • card – a credit/debit card; • wm – linked WebMoney wallet; • ssk – payment by cash in a QIWI Terminal. When specified method is inaccessible, the page contains notice about it and the client can choose another method.	ssk

Optionally, Visa QIWI Wallet will redirect back to URL specified in **successUrl** or **failUrl** parameter (if any) when the user completes payment process.

NOTE

Redirection occurs when the user pays by Visa QIWI Wallet balance on Visa QIWI Wallet web site only.

WARNING

Redirection to **successUrl** does not necessarily mean that invoice is successfully paid. You should wait for the Visa QIWI Wallet's notification with final invoice status to the merchant's server before complete the user order. When notifications are not used, you should request the invoice status.

The URL for redirection supplements **order** parameter with its value as the invoice ID taken from the **transaction** parameter of the initial GET-request. Using this parameter, merchant can render the final page depending on the order details.

Example:

- Merchant redirects to URL:

  https://qiwi.com/order/external/main.action?shop=2042&transaction=123123123&successUrl=http%3A%2F%2Fmystore.com%2Fsuccess%3Fa%3D1%26b%3D2&failUrl=http%3A%2F%2Fmystore.com%2Ffail%3Fa%3D1%26b%3D2&iframe=true&target=iframe&pay_source=qw.
- User pays the invoice by Visa QIWI Wallet balance on Visa QIWI Wallet site and completes payment process.
- If transaction is successfully created, Visa QIWI Wallet redirects user to http://mystore.com/success?a=1&b=2&order=123123123.
- If creation of transaction is unsuccessful, Visa QIWI Wallet redirects user to http://mystore.com/fail?a=1&b=2&order=123123123.

4.4. Requesting Invoice Status

Merchant can request status of the invoice by sending GET-request to the following URL:

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

where:

- **{prv_id}** – merchant's Shop ID (numeric value, as displayed in **Shop ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site);
- **{bill_id}** – unique invoice identifier generated by the merchant (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).

Sample request (JSON response):

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

Response:

```
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": "Order #1234 at hosting.com"
  }
}}
```

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

where:

- **{prv_id}** – merchant's respective Shop ID (numeric value, as displayed in **Shop ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site);
- **{bill_id}** – unique invoice identifier generated by the merchant (non-empty string of up to 200 characters).

Request parameters:

Name	Value format	Regexp	Description
status	"rejected"	^rejected$	New invoice status (cancelled). **Required.**

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).

Sample request (JSON response):

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

status=rejected
```

Response:

```
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"
  }
}}
```

4.6. Refunds

Merchant can process a full or partial refund to user's Visa QIWI Wallet account using the PUT-request below. This request creates a reversed transaction for the initial one.

Merchant can create several refund operations for the same initial invoice provided that:

- Amount of all refund operations does not exceed initial invoice amount.
- Different refund IDs used for different refund operations of the same invoice (see below).

WARNING

When the transmitted amount exceeds the initial invoice amount or the amount left after the previous refunds, error code 242 returns.

To process the request, merchant has to send PUT-request to the following URL:

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

where:

- **{prv_id}** – merchant's Shop ID (numeric value, as displayed in **Shop ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site);
- **{bill_id}** – unique invoice identifier generated by the merchant (non-empty string of up to 200 characters);
- **{refund_id} –** refund identifier, a number specific to a series of refunds for the invoice **{bill_id}** (string of 1 to 9 symbols – any 0-9 digits and upper/lower Latin letters).

Request parameters:

Name	Value format	Regexp	Description
amount	A positive number rounded up to 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. **Required.**

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

Sample request (JSON response):

```
PUT /api/v2/prv/2042/bills/BILL-1/refund/A1
Accept: text/json
Authorization: Basic ***
Host: api.qiwi.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8

amount=5.0
```

Response:

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

{"response": {
  "result_code": 0,
```

```
  "refund": {
    "refund_id": A1,
    "amount": "5.00",
    "status": "success",
    "error": 0
  }
}}
```

4.7. Refund Status Verification

Merchant can verify status of the refund by sending GET-request to the following URL:

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

Where:

- **{prv_id}** – merchant's Shop ID (numeric value, as displayed in **Shop ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site);
- **{bill_id}** – unique invoice identifier generated by the merchant (non-empty string of up to 200 characters);
- **{refund_id} –** refund identifier, unique number from a series of refunds processed for the invoice **{bill_id}** (see section 4.6).

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 refund of the invoice (for more details see server response).

Sample request (JSON response):

```
GET /api/v2/prv/2042/bills/BILL-1/refund/A1
Accept: text/json
Authorization: Basic ***
Host: api.qiwi.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
```

Response:

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

{"response": {
  "result_code": 0,
  "refund": {
    "refund_id": A1,
    "amount": "5.00",
    "status": "success",
    "error": 0
  }
}}
```

4.8. Server Response

The server response is serialized in XML or JSON (depends on "Accept" header in a request). The response data represents a "Response" object with the following elements:

- "Result code" element.
- One of the following objects depending on the request type:
  - "Invoice",

– "Refund".

The object may be empty if the result is unsuccessful.

4.8.1. Result code

The result code of an operation is in the following parameter of the response:

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,
…
```

4.8.2. Information on invoice

Invoice object is named as "bill" and has the following parameters:

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 2 or 3 decimal places after the comma	^\d+(\.\d{0,3})?$	The invoice 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 – phone number in international format	^tel:\+\d{1,15}$	The Visa QIWI Wallet 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 from ShopName"
}
…
```

4.8.3. Information on refund

The refund object names as "refund" and has the following parameters:

Name	Value format	Regexp	Description
refund_id	String containing digits from 0 to 9 and upper/lower Latin letters up to 9 symbols	^[A-Za-z0-9]{1,9}$	The refund identifier, unique number in a series of refunds processed for a particular invoice.
amount	A positive number rounded up to 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 Refund status)
error	Integer from 0 to 5000	^\d{1,4}$	Error code (see error codes). **Important!** When the amount of refund exceeds the initial invoice amount or the amount left after the previous refunds, error code 242 is returned.
user	String of the form "tel:phone_number", where phone_number – phone number in international format	^tel:\+\d{1,15}$	The Visa QIWI Wallet 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"
}
…
```

5. MERCHANT NOTIFICATION INTERFACE

This API makes it possible for merchant to receive the notifications on invoice status change.

Notifications are POST-requests from Visa QIWI Wallet server containing all relevant data of the invoice serialized as HTTP-request parameters (encoded by UTF-8) plus parameter "command", which always has "bill" value.

WARNING

As new parameters of the invoice might be introduced on Visa QIWI Wallet side, a list of invoice parameters in HTTP-request should not be fixed on the merchant's side and should be taken from the request itself.

To enable notifications, use **Turn on notifications** flag in **Settings → Protocols details → REST-protocol** section of ishop.qiwi.com web site. To receive these notifications merchant's server should be able to accept specific HTTP-requests on ports 80, 443.

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

Section 5.1 describes authorization of notification requests on merchant's side.

Merchant response to the notification should be in accordance with requirements specified in section 5.2.

5.1. Authorization on Merchant's Server

The merchant's service has to accept notifications from Visa QIWI Wallet with HTTP or HTTPS protocol.

To use HTTPS protocol, merchant's server should support SSL-encryption and client SSL certificate verification.

If the client SSL-certificate is self-generated and is not issued by one of the standard certification centers, this certificate should be uploaded to the Visa QIWI Wallet server via the merchant's console (**Certificate** field in **Settings → Protocols details → REST-protocol** section of ishop.qiwi.com web site). Certificate must be in one of the following formats:

- PEM (text file with .pem extension) – (Privacy-enhanced Electronic Mail) BASE64 encoded DER certificate placed between "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" strings.
- DER (binary file with .cer, .crt, .der extensions) – usually in binary DER format, though PEM certificates are also accepted with this extensions.

Then the merchant's certificate becomes trusted after the upload.

Merchant's server should perform authorization of the notifications by one of the following ways:

1. Plain text password basic-authorization. It is recommended to use HTTPS protocol in this case to make request well protected.

   The "Shop ID: Notification password" couple is used where:

   – Shop ID – merchant's Shop ID, as displayed in **Shop ID** parameter of **Protocols details → REST-protocol** section of ishop.qiwi.com web site.

   – Notification password – password for notifications issued on merchant's registration. Merchant should change it immediately in **Protocols details → REST-protocol** section of ishop.qiwi.com web site (**Change password** button). If necessary, merchant may reset password for notifications at any time.

The authorization data are transmitted according to the rules of basic-authorization. The HTTP header "Authorization" is added to the request. The value of this parameter is composed of the word "Basic", blank character and encrypted BASE64 pair "Shop ID: Notification password":

```
Authorization: Basic ***
```

Where:

```
BASE64("Shop ID:Notification password") = "***"
```

Data should be verified on the merchant's server.

WARNING

If the request is not SSL-encrypted, the transmission is not secure and the data becomes vulnerable to interception.

2. Simplified signature. Merchant should enable **Sign** flag in **Protocols details → REST-protocol** section of ishop.qiwi.com web site.

   The HTTP header "X-Api-Signature" is added to the POST-request. The value of the header (signature) is formulated as follows:

   2.1. Prepare a string of all parameters values from the POST-request sorted in alphabetical order and separated by "|":

```
Invoice_parameters = {parameter1}|{parameter2}|..
```

   Where **{parameter1}** – value of the respective parameter of the invoice taken from the notification. All parameter values included into the sign should be treated as strings.

   2.2. Transform obtained string and notification password (see Method 1) into bytes encoded in UTF-8.

   2.3. Apply HMAC-SHA1 function:

```
sign = HMAC-SHA1(Notification_password_byte, Invoice_parameters_byte)
```

   Where:

   - **Notification_password_byte** – transformed merchant's password for notifications (from step 2.2) as a key for signature;
   - **Invoice_parameters_byte** – transformed string from step 2.2;
   - **sign** – result string.

   2.4. Put byte result of HMAC-SHA1 function encoded in BASE64 string into the HTTP header.

NOTE

Examples of Method 2 implementation are in Examples of Signed Notification Authorization section (6.6).

5.2. Requirements to the Response

- The response should be in XML.
- "Content-type" HTTP header must be "text/xml" otherwise notification is treated as unsuccessful on Visa QIWI Wallet side.
- In response to the request, the result code must be returned in `result_code` tag within `result` tag.

  In order to help in identifying the reasons of notification errors we recommend that the result codes returned by the merchant be in accordance with Notification codes table.
- Any response with result code other than 0 ("Success") and/or HTTP status code other than 200 (OK) will be treated by Visa QIWI Wallet server as a temporary error. Thus the server will continue repeating requests (notifications) with increased time intervals within next 24 hours (50 attempts in total) until it gets a response with result code 0 ("Success") and HTTP status code 200 (OK).

  If the response with result code 0 ("Success") and HTTP status code 200 has not been received within 24 hours, Visa QIWI Wallet 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.

5.3. Notification Examples

Sample request to merchant's server:

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

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

Sample request to merchant's server with digital signature:

```
POST /qiwi-notify.php HTTP/1.1
Accept: text/xml
Content-type: application/x-www-form-urlencoded
X-Api-Signature: J4WNfNZd***V5mv2w=

command=bill&bill_id=orderIdLocalTest17&status=paid&error=0&amount=0.01&
user=tel%3A%2B78000005122&prv_name=Test&ccy=RUB&
comment=Some+Descriptor%7C112981674186701448882638413096 64
```

The following response is expected (result code may vary):

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

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

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. Refund Status

Status code	Description	Final status?
Processing	Payment refund is pending	No
success	Payment refund is successful	Yes
fail	Payment refund is unsuccessful	Yes

6.3. Error Codes

Error code	Description	Fatal? *
0	Success	
5	Incorrect data in the request parameters	Yes
13	Server is busy, try again later	No
78	Operation is forbidden	Yes
150	Authorization error (e.g. invalid login/password)	Yes
152	Protocol is not enabled or protocol is disabled	No
155	This merchant's identifier (API ID) is blocked	Yes
210	Invoice not found	Yes
215	Invoice with this *bill_id* already exists	Yes
241	Invoice amount is less than allowed	Yes
242	Invoice amount is greater than allowed. Also returns to refund request when the amount of refund exceeds the initial invoice amount or the amount left after the previous refunds	Yes
298	User not registered	Yes

Error code	Description	Fatal? *
300	Technical error	No
303	Wrong phone number	Yes
316	Authorization from the blocked merchant	No
319	No rights for the operation	No
339	IP-addresses blocked	Yes
341	Required parameter is incorrectly specified or absent in the request	Yes
700	Monthly limit on operations is exceeded	Yes
774	Visa QIWI Wallet user account temporarily blocked	Yes
1001	Currency is not allowed for the merchant	Yes
1003	No convert rate for these currencies	No
1019	Unable to determine wireless operator for mobile commerce	Yes
1419	Bill was already payed	Yes

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

6.4. Notification Codes

Result code	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

6.5. Example of Web-interface for Invoice

This example written on PHP is given to demonstrate using merchant's authorization parameters, i.e. shop ID, API ID and password for the API ID.

```
<?php
//Shop identifier from Merchant details page
//https://ishop.qiwi.com/options/http.action
$SHOP_ID = "21379721";
//API ID from Merchant details page
//https://ishop.qiwi.com/options/rest.action
$REST_ID = "62573819";
//API password from Merchant details page
//https://ishop.qiwi.com/options/rest.action
$PWD = "**********";
```

```
//Invoice ID
$BILL_ID = "99111-ABCD-1-2-1";
$PHONE = "79191234567";

$data = array(
    "user" => "tel:+" . $PHONE,
    "amount" => "1000.00",
    "ccy" => "RUB",
    "comment" => "Good choice",
    "lifetime" => "2015-01-30T15:35:00",
    "pay_source" => "qw",
    "prv_name" => "Special packages"
);

$ch = curl_init('https://api.qiwi.com/api/v2/prv/'.$SHOP_ID.'/bills/'.$BILL_ID);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, 'PUT');
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
curl_setopt($ch, CURLOPT_USERPWD, $REST_ID.":".$PWD);
curl_setopt($ch, CURLOPT_HTTPHEADER,array (
    "Accept: application/json"
));
$results = curl_exec ($ch) or die(curl_error($ch));
echo $results;
echo curl_error($ch);
curl_close ($ch);
//Optional user redirect
$url = 'https://qiwi.com/order/external/main.action?shop='.$SHOP_ID.'&
transaction='.$BILL_ID.'&successUrl=http%3A%2F%2Fieast.ru%2Findex.php%3Froute%3D
payment%2Fqiwi%2Fsuccess&failUrl=http%3A%2F%2Fieast.ru%2Findex.php%3Froute%3D
payment%2Fqiwi%2Ffail&qiwi_phone='.$PHONE;
echo '<br><br><b><a href="'.$url.'">Redirect link to pay for invoice</a></b>';
  ?>
```

6.6. Examples of Signed Notification Authorization

6.6.1. Java

```
import java.util.Base64;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class Hmac {

/* String of the notification parameters
   Values correspond to parameters' sequence:
   amount, bill_id, ccy, command, comment, error, prv_name, status, user  */

    private static String data = "2.00|5101603|RUB|bill|test-checking-one-way-
response-from-processing|0|simple test|paid|tel:+79167421378";

/* Notification password */
    private static String key = "123456789";

    public static void main(String[] args){
        System.out.println(getSignedBody(data, key));
    }
```

```
    private static String getSignedBody(String data, String key){
        String result = "";
        try {
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes("UTF-8"),
"HmacSHA1");
            Mac mac = Mac.getInstance("HmacSHA1");
            mac.init(signingKey);
            byte[] rawHmac = mac.doFinal(data.getBytes("UTF-8"));
            result = Base64.getEncoder().encodeToString(rawHmac);
        } catch (Exception e) {
            System.out.println("Failed to generate HMAC: " + e.getMessage());
        }
        return result;
    }

}
```

6.6.2. PHP

```
<?php

function hexToStr($hex){
    $string='';
    for ($i=0; $i < strlen($hex)-1; $i+=2){
        $string .= chr(hexdec($hex[$i].$hex[$i+1]));
    }
    return $string;
}

//Signature generation by key and string
function checkSign($key, $req){
    $sign_hash = hash_hmac("sha1", $req, $key);
    $sign_tr = hexToStr($sign_hash);
    $sign = base64_encode($sign_tr);
    return $sign;
}

//Sort POST-request parameters and return values
function getReqParams(){
    $reqparams = "";
    ksort($_POST);
    foreach ($_POST as $param => $valuep) {
        $reqparams = "$reqparams|$valuep";
    }
    return substr($reqparams,1);
}

//Take signature from the request
function getSign(){
    $HEADERS = getallheaders();
    foreach ($HEADERS as $header => $value) {
       if ($header == 'X-Api-Signature') {
            $SIGN_REQ = $value;
       }
    }
    return $SIGN_REQ;
}

// Sort request parameters
$Request = getReqParams();
// Notificatoin password for the shop
$NOTIFY_PWD = "***";
```

```
//Prepare signature
$reqres = checkSign($NOTIFY_PWD, $Request);

//Signature from request
$SIGN_REQ = getSign();

//Compare results
if ($reqres == $SIGN_REQ) {
    $error = 0;
}
else $error = 150;

//Response
header('Content-Type: text/xml');
$xmlres = <<<XML
<?xml version="1.0"?>
<result>
<result_code>$error</result_code>
</result>
XML;
echo $xmlres;
?>
```