WebPay Form integration manual

Version 1.2.5.

Content

Requirements

Integration must be done within our test environment first. When this process is finished and approved by our staff, you may go live and start processing with real money.

To start integrating with WebPay service you will need:

- test merchant account
- HTTP client library

If you don't have a test merchant account, please contact us at podrska@webteh.hr and we will open one for you. Then you can login into your account at https://ipg.webteh.hr/en/login with login and password provided.

WebPay Direct is a REST web service and you will need a HTTP client library (like cURL - http://curl.haxx.se). All API calls are XML documents POST-ed over HTTPS to our test server at https://ipg.webteh.hr.

IMPORTANT Parametrize `https://ipg.webteh.hr` URL, in production mode the subdomain will be different.

NOTICE More on REST at http://en.wikipedia.org/wiki/Representational_state_transfer.

An example of such call using cURL from command line may look like this:

```
curl -H "Content-Type: application/xml" -H "Accept: application/xml" -k -i -d
request_xml https://ipg.webteh.hr/action
```

where request_xml is a XML document that contains data necessary for transaction processing and `https://ipg.webteh.hr/action` is an URL that responds to certain API call.

IMPORTANT Accept and Content-Type headers must be set to application/xml for all message types.

Variables - names, lenghts and formats

Here are the variables and their definitions used when generating XML documents for API calls:

Buyer's profile

name	lenght	format	additional info
ch_full_name	3-30	alphanumeric	buyer's full name
ch_address	3-100	alphanumeric	buyer's address
ch_city	3-30	alphanumeric	buyer's city
ch_zip	3-9	alphanumeric	buyer's zip
ch_country	3-30	alphanumeric	buyer's country
ch_phone	3-30	alphanumeric	buyer's phone
ch_email	3-100	alphanumeric	buyer's email

Card details

name	lenght	format	additional info
pan	14-19	integer	valid card number

cvv	3-4	integer	optional for Maestro cards; must be not set for MOTO transactions
expiration_date	4	numeric (YYMM)	this value can not be in the past

Order details

name	lenght	format	additional info
order_info	3-100	alphanumeric	short description of order being processed
order_number	1-40	alphanumeric	unique identifier
amount	3-11	integer	amount is in minor units, ie. 10.24 USD is sent as 1024
currency	predefined	alpha	possible values are `USD`, `EUR`, `BAM` or `HRK`

Processing data

name	lenght	format	additional info
ip	7-15	alphanumeric	valid IPv4 address
language	predefined	alpha	used for errors localization, possible values are `en`, `es`, `ba` or `hr`
transaction_type	predefined	alpha	possible values are `authorize`, `purchase`, `capture`, `refund` or `void`

authenticity_token	40	alphanumeric	autogenerated value for merchant account, can be found under merchant settings
digest	40	alphanumeric	SHA1 hash generated from concatenation of key, order_number, amount and currency as strings; key can be found under merchant settings
number_of_installments	1-2	integer	range 2-12
moto	predefined	alpha	possible value is `true` or `false`; missing variable is equivalent to `false`

Transaction messages

Authorization

Authorization is a preferred transaction type for e-commerce. Merchant must capture these transactions within 28 days in order to transfer the money from buyer's account to his own. This transaction can also be voided if buyer cancel the order. Refund can be done after original authorization is captured.

Below is an XML example of authorization message in which transaction_type tag has a value authorize. This XML document is generated according to variable definitions.

Digest is calculated using following formula:

```
digest = SHA1(key + order_number + amount + currency)
```

With the following example data

- key: qwert123
- order_number: abcdef
- amount: 54321
- currency: EUR

the digest formula gives a result as follows:

```
digest = SHA1("qwert123abcdef54321EUR") =
"16e943d2b84546ce4271de51679abc3bf1eb163b"
```

`key` is a shared secret used to calculate digest value. It is set through merchant interface under API settings of your merchant account.`authenticity_token` is autogenerated value and is copied from merchant account.

NOTICE Client does not send a TID/MID pair in authorization message, those are set in merchant account.

Authorization request example

```
 <?xml version="1.0" encoding="UTF-8"?>

<transaction>

  <transaction-type>authorize</transaction-type>

  <amount>54321</amount>

  <expiration-date>1707</expiration-date>

  <cvv>265</cvv>

  <pan>4111111111111111</pan>

  <ip>10.1.10.111</ip>

  <order-info>1 x SnowMaster 3000</order-info>

  <ch-address>Elm street 22</ch-address>

  <ch-city>Knoxville</ch-city>

  <ch-country>Tennessee</ch-country>

  <ch-email>email@example.com</ch-email>

  <ch-full-name>John Doe</ch-full-name>

  <ch-phone>phone</ch-phone>

  <ch-zip>123456789</ch-zip>

  <currency>EUR</currency>

  <digest>16e943d2b84546ce4271de51679abc3bf1eb163b</digest>

  <authenticity-token>7db11ea5d4a1af32421b564c79b946d1ead3daf0</authenticity-token>
```

  <order-number>abcdef</order-number>

  <language>en</language>

</transaction>

This XML is now posted to `https://ipg.webteh.hr/api`.

IMPORTANT Parametrize `https://ipg.webteh.hr` URL, in production mode the subdomain will be different.

If all values pass validations at our side, transaction is send to the bank and response is returned. This response may look like this:

- **HTTP status code:** 201 - Created
- **HTTP headers:** {:connection=>"close", :date=>"Tue, 25 Oct 2011 01:18:37 GMT", :location=>"https://ipg.webteh.hr/transactions/845", :content_type=>"application/xml; charset=utf-8", :cache_control=>"no-cache", :x_ua_compatible=>"IE=Edge", :x_runtime=>"1.475305", :transfer_encoding=>"chunked"}
- **HTTP body:**
- <?xml version="1.0" encoding="UTF-8"?>
- <transaction>
-   <id type="integer">845</id>
-   <acquirer>rogach bank</acquirer>
-   <order-number>abcdef</order-number>
-   <amount type="integer">54321</amount>
-   <response-code>000</response-code>
-   <approval-code>38860</approval-code>
-   <response-message>authorization OK</response-message>
-   <reference-number>898951263</reference-number>
-   <systan>83704</systan>
-   <eci>05</eci>
-   <xid>tgC4qopGGk28I15K45m7BgAABAQ=</xid>
-   <acsv>AAABB2kQZQAAAAAAEhBIAAAAAAA=</acsv>
-   <cc-type>visa</cc-type>
-   <status>approved</status>
-   <transaction-type>authorize</transaction-type>
-   <created-at type="datetime">2011-10-25T03:18:38+02:00</created-at>
-   <enrollment>Y</eci>
-   <authentication>A</eci>
- </transaction>

New transaction is generated - `201 Created HTTP status code`, and it's location is set in appropriate HTTP header. A client then must parse a body from HTTP response and extract all values from that XML document. Transaction is approved only and if only status is set to `approved`. All other fields are standard data carried over payment networks. If issuer declines a transaction, status flag is set to `decline`. In a case of an error, the flag will be set to `invalid`.

IMPORTANT Do not rely on any output variable except status to determine successful of autorization.

IMPORTANT authorize messages won't be settled unless they are captured within 28 days. After authorization is captured, it can be refunded within 180 days.

NOTICE We highly recommend to our merchants to keep a whole response (this includes HTTP headers and body) and to save all parsed values for easier troubleshooting during the integration phase and production later on. Even if the body is empty, HTTP response code is valuable information; HTTP headers are in the hearth of REST architecture. The quality of our support depends on availability of these information.

In case of invalid request, service will also return a response with `406 Not Acceptable HTTP status code` and XML document in it's body. Each ofended variable will be printed out along with brief explanation what went wrong. That response may look like this:

```
 <?xml version="1.0" encoding="UTF-8"?>

<errors>

  <error>Ch phone can't be blank</error>

  <error>Ch phone is too short (minimum is 3 characters)</error>

  <error>Ip can't be blank</error>

  <error>Ip is not a valid IPv4 address</error>

  <error>Digest is invalid</error>

</errors>
```

This invalid request is also recorded and errors are visible through merchant account interface.

Purchase

Purchase doesn't need to be approved, funds are transfered in next settlement between issuer and acquirer banks, usually within one business day. These transactions can be refunded within 180 days.

This message has the same structure as authorization request XML document, only difference is in transaction_type tag which has `purchase` value now. Response has identical structure as authorization response and all response fields should be treated in the same way.

NOTICE purchase message can be refunded within 180 days.

Capture

Approved authorization must be captured in order to transfer the funds form buyer's card to merchant. This action can be done through merchant account interface or programatically through an API call. A lesser amount than original authorization amount can be captured, ie. a merchant can make a partial delivery of goods and/or services. Capture must be done within 28 days or will be automatically voided.

After an authorization is successfully made, a capture call must be done to settle that authorization.

Capture XML document is POST-ed to `https://ipg.webteh.hr/transactions/:order_number/capture.xml`, where `:order_number` has a value from original authorization.

IMPORTANT Parametrize `https://ipg.webteh.hr` URL, in production mode the subdomain will be different.

Document example for capture message may look like this:

```
 <?xml version="1.0" encoding="UTF-8"?>

<transaction>

  <amount>54321</amount>

  <currency>EUR</currency>

  <digest>e64d4cd99367f0254ed5296d38fad6ce87d3acab</digest>

  <authenticity-token>7db11ea5d4a1af32421b564c79b946d1ead3daf0</authenticity-token>

  <order-number>11qqaazz</order-number>

</transaction>
```

where digest is calculated the same way as for authorize or purchase messages.

Response has identical structure as authorization response and all response fields should be treated in the same way.

NOTICE capture message can be refunded within 180 days.

Refund

Approved purchases and captures can be refunded within 180 days. This action can be done through merchant account interface or programatically through an API call. This is required if merchant cancels the order after transaction is settled, or buyer is not satisfied with delivered goods and/or services. Refunds can be done with lesser amount than original authorization amount.

Request XML for this transaction_type is identical to capture example, but now the document is POST-ed to `https://ipg.webteh.hr/transactions/:order_number/refund.xml`, where `:order_number` has a value from original purchase or capture.

Response has identical structure as authorization response and all response fields should be treated in the same way.

Void

Approved authorization must be voided within 28 days if merchant cancels the order. This action can be done through merchant account interface or programatically through an API call.

Void messages are POST-ed to to `https://ipg.webteh.hr/transactions/:order_number/void.xml`, where `:order_number` has a value from original message (authorization, capture, purchase or refund). They have identical structure as capture or refund.

Response has identical structure as authorization response and all response fields should be treated in the same way.

Transactions with installments

To send a request with installments, `number-of-installments` tag is added to authorize or purchase XML document. This number depends on merchant's setting in acquiring bank, it's value is a number between 2 and 12

Moto transactions

To send a request for moto transaction, `moto` tag is added to authorize or purchase XML with `true` value.

3-D Secure messages

If your merchant account has active 3-DS flag under it's settings, all incoming authorize and purchase requests will be processed as 3-D Secure transactions. For cards not enrolled in 3-DS, a regular authorize or purchase response will be returned.

If the card is enrolled, a 3-DS check will occur and appropriate response is returned which may look like this:

- **HTTP status code:** 201 - Created
- **HTTP headers:** {:connection=>"close", :date=>"Wed, 26 Oct 2011 15:39:18 GMT", :content_type=>"application/xml; charset=utf-8", :cache_control=>"no-cache", :x_ua_compatible=>"IE=Edge", :x_runtime=>"4.147298", :transfer_encoding=>"chunked"}
- **HTTP body:**
- <?xml version="1.0" encoding="UTF-8"?>
- <secure-message>
-   <id type="integer">505</id>
-   <acs-url>https://visatest.3dsecure.com/mdpayacs/pareq</acs-url> <pareq>eJxVUV1TwjAQ/CsMr47NR0s/mCMzRR7EoQ4C+srUcELRtpA2Av/epLa...</pareq>

- <authenticity-token>7465c9ab97defa1501ed0e680b3a0b4b88937c17</authenticity-token>
- </secure-message>

Client should parse a HTTP body from above example response and extracts `acs-url`, `pareq` and `authenticity-token` values. They are POST-ed through buyer's browser to ACS server at `acs-url` as follows (this example use javascript to automatically submit the form):

```
 <!DOCTYPE html>

<html>

  <head>

    <title>3D Secure Verification</title>

    <script language="Javascript">

      function OnLoadEvent() { document.form.submit(); }

    </script>

  </head>

  <body OnLoad="OnLoadEvent();">

    Invoking 3-D secure form, please wait ...

    <form name="form" action="acs-url" method="post">

      <input type="hidden" name="PaReq" value="pareq">

      <input type="hidden" name="TermUrl" value="term-url">

      <input type="hidden" name="MD" value="authenticity-token">

      <noscript>

        <p>Please click</p><input id="to-asc-button" type="submit">

      </noscript>

    </form>
```

  </body>

</html>

where `acs-url`, `pareq` and `authenticity-token` are substituted with appropriate extracted values.

Buyer will POST the result of 3-D secure identity check from ACS server to `term-url` at merchant side through his browser. Following data is captured at merchant's term-url:

- PaRes - eJzNmGuvosyygP/KZPZHMy93lYmzkuaOCnJH+LLDTe6ggoD8+t3qrDXrnUxO3rO/nENigKK6urq76qm2N1Z2TR ...
- MD - 7465c9ab97defa1501ed0e680b3a0b4b88937c17

NOTICE Merchant should implement a listener at `term-url` that captures response from issuer's ACS server.

The 3-D secure processing is done and merchant now issue a new request to `https://ipg.webteh.hr/pares` to finish the transaction. Example of such request may look like this:

```
 <?xml version="1.0" encoding="UTF-8"?>

<secure-message>

  <MD>7465c9ab97defa1501ed0e680b3a0b4b88937c17</MD>


<PaRes>eJzNmGuvosyygP/KZPZHMy93lYmzkuaOCnJH+LLDTe6ggoD8+t3qrDXrnUxO3rO/nENigKK6urq76qm2N1Z2TR ...</PaRes>

</secure-message>
```

NOTICE authenticity-token from WbPay is submitted to ACS server in variable MD; then is sent back again to WebPay as MD variable.

WebPay will return response as would for a regular authorize or purchase request messages. Only difference is that `eci`, `xid`, `acsv`, `enrollment` and `authentication` fields are now populated in response XML according to 3-DS rules.

List of response codes

Here is the list of response codes and their description:

- 0000 - Approved
- 1001 - Card expired
- 1002 - Card suspicious
- 1003 - Card suspended
- 1004 - Card stolen
- 1005 - Card lost

- 1011 - Card not found
- 1012 - Cardholder not found
- 1014 - Account not found
- 1015 - Invalid request
- 1016 - Not sufficient funds
- 1017 - Previously reversed
- 1018 - Previously reversed
- 1019 - Further activity prevents reversal
- 1020 - Further activity prevents void
- 1021 - Original transaction has been voided
- 1022 - Preauthorization is not allowed for this card
- 1023 - Only full 3D authentication is allowed for this card
- 1024 - Installments are not allowed for this card
- 1025 - Transaction with installments can not be send as preauthorization
- 1026 - Installments are not allowed for non ZABA cards
- 1050 - Transaction declined
- 1802 - Missing fields
- 1803 - Extra fields exist
- 1804 - Invalid card number
- 1806 - Card not active
- 1808 - Card not configured
- 1810 - Invalid amount
- 1811 - System error - database
- 1812 - System error - transaction
- 1813 - Cardholder not active
- 1814 - Cardholder not configured
- 1815 - Cardholder expired
- 1816 - Original not found
- 1817 - Usage limit reached
- 1818 - Configuration error
- 1819 - Invalid terminal
- 1820 - Inactive terminal
- 1821 - Invalid merchant
- 1822 - Duplicate entity
- 1823 - Invalid acquirer
- 2000 - Internal error - host down
- 2001 - Internal error - host timeout
- 2002 - Internal error - invalid message
- 2003 - Internal error - message format error
- 2013 - 3D Secure error - invalid request
- 3000 - Time expired
- 3100 - Function not supported
- 3200 - Timeout
- 3201 - Authorization host not active
- 3202 - System not ready
- 4001 - 3D Secure error - ECI 7
- 4002 - 3D Secure error - not 3D Secure, store policy
- 4003 - 3D secure error - not authenticated
- 5000 - Request in progress
- 5018 - RISK: Minimum amount per transaction
- 5019 - RISK: Maximum amount per transaction

- 5001 - RISK: Number of repeats per PAN
- 5020 - RISK: Number of approved transactions per PAN
- 5003 - RISK: Number of repeats per BIN
- 5016 - RISK: Total sum on amount
- 5021 - RISK: Sum on amount of approved transactions per PAN
- 5022 - RISK: Sum on amount of approved transactions per BIN
- 5005 - RISK: Percentage of declined transactions
- 5009 - RISK: Number of chargebacks
- 5010 - RISK: Sum on amount of chargebacks
- 5006 - RISK: Number of refunded transactions
- 5007 - RISK: Percentage increment of sum on amount of refunded transactions
- 5023 - RISK: Number of approved transactions per PAN and MCC on amount
- 5011 - RISK: Number of retrieval requests
- 5012 - RISK: Sum on amount of retrieval requests
- 5013 - RISK: Average amount per transaction
- 5014 - RISK: Percentage increment of average amount per transaction
- 5015 - RISK: Percentage increment of number of transactions
- 5017 - RISK: Percentage increment of total sum on amount
- 5050 - RISK: Number of repeats per IP
- 5051 - RISK: Number of repeats per cardholder name
- 5052 - RISK: Number of repeats per cardholder e-mail
- 6000 - Systan mismatch