GOC.IO

PAYMENTS SYSTEM

E-COMMERCE VENTURES XML INTERFACE

VERSION 1.00.40 **JANUARY 17TH, 2014**

Basic aspects of the interface setup

Access Setup

E-commerce ventures interface of GOC.IO provides an XML protocol access to any user of the internet without limitations, however the key required especially for XML interface operations.

The key for XML interface operations is stored in *pwd_bill* field of the user's record, kept plain (unencoded) or hashed. This field may be changed via web-interface of the system by user (via web-interface), or GOC.IO administrator's permissions may be required in order to get that field changed (according to the policy of the system).

E-commerce ventures XML interface is set up on the internet address

```
https://basic_system_hostname/ec_xml/
```

E.g.:

```
https://goc.io /ec_xml/
```

XML requests should be sent to that URL via POST method in requests' body, and the response to each request is provided by the server immediately (synchronously) in the response stream.

Essential Functions

In order to benefit from an ability of accepting payments via GOC.IO system, e-commerce venture requires the following functions: *a)* issuing a bill to an user of the system (*BILL* in terms of GOC.IO system), *b)* checking the status of the issued bill by it's ID (*BILLID*) or it's unique ID in e-commerce system (*UNIQUEID*), and if e-commerce venture accepts Bitcoin payments directly to it's account, *c)* requesting the current relative RUR/BTC rate.

Functions of the Interface

Requesting the Relative RUR/BTC Rate

It is assumed that the main volume of GOC.IO system operations is held in Bitcoin currency, that has currency ID "1". Therefore, before the bill is issued to the user of GOC.IO system, e-commerce venture should request the current relative RUR/BTC rate in order to calculate the price of it's goods or services in Bitcoins, using own protection algorythms preventing loss in case of rapid decline of the rate during the period when payment against the bill from the user is expected.

The following XML request should be sent in order to get the current RUR/BTC exchange rate:

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
        <command>get_btc_rate</command>
</request>
```

In the response GOC.IO system will immediately report the current RUR/BTC rate as well as last recorder change of the rate:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
        <result>0</result>
        <result_code>0</result_code>
        <BTC_rate>19921.069</BTC_rate>
        <BTC_rate_last_change>-59.00000000</BTC_rate_last_change>
</response>
```

The are following variables in the response:

- *result* – processing result code (*0* – success, any other value means error);

- *result_code* – explanation of the processing result code in text (*0* will be returned in case of successful processing, no need to check the value of this variable);
- *BTC_rate* – ralative RUR/BTC rate (amount of RUR per one Bitcoin) with decimal point (if the value is fractional);
- *BTC_rate_last_change* – positive, negative or zero value of the last recorded change of the RUR/BTC rate (generally the last recorded change of the rate will be either positive or negative).

High volatility of the RUR/BTC rate should be taken into consideration, meaning that after the rate was requested its' actual value may drastically change (for more than several percents) to any direction.

Issuing a Bill to the Customer of the System

The following XML request is used in order to issue a bill to the user of GOC.IO:

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
        <command>register_bill</command>
        <from_USERID>TEST_SHOP</from_USERID>
        <to_USERID>PAYEE_USERID</to_USERID>
        <currency>1</currency>
        <value>0.0520</value>
        <bill_details>Bill details</bill_details>
        <technical_details></technical_details>
        <UNIQUEID></UNIQUEID>
        <bill_valid>2013-11-20 07:10:00</bill_valid>
        <md5_hash>2db5b13e846ff69ae60e9ed6e45d6a2c</md5_hash>
</request>
```

There are following variables in the request:

- *command* – command code (register_bill);
- *from_USERID* – the ID of GOC.IO system user who is issuing a bill (e-commerce venture's user ID);
- *to_USERID* – the ID of user being billed (user with such an ID at the moment of issuing a bill may be not yet registered in GOC.IO);
- *currency* – ID of the GOC.IO currency that is used for billing («1» means Bitcoin); following values of this variable are supported:
  - *1, empty value,* or *BTC* – Bitcoin currency ID;
  - *2, RUR,* or *RUB* – Roubles currency ID.
- *value* – value of the bill to be paid in corresponding currency;
- *bill_details* – description of the bill in text providing to user the information on goods or services (description of goods or services, brief information on it's features etc.);
- *technical_details* – a variable for technical information that will be kept together with the basic bill information, e.g. serialized JSON array consisting some additional bill's attributes, in particular GOC.IO itself processes following parameters in case the appear in JSON array of the *technical_details* variable:
  - *success_url* – the user will be prompted to follow the corresponding URL in case the bill will be paid successfully;
  - *error_url* – the user will be prompted to follow the corresponding URL in case some error will occur during the payment attempt.

  If *success_url* and/or *error_url* parameters are used, it should be taken into consideration that the user of GOC.IO may be followed to corresponding URLs unlimited number of times, therefore the e-commerce venture's system should manage itself the unique nature of the actions performed when the URL is followed

(services provision, goods delivery etc.).

- *UNIQUEID* – unique ID of the bill according to e-commerce venture's system;
- *bill_valid* – a variable that may limit the possibility of bill payment by particular date and time in 'YYYY-MM-DD HH:MM:SS' format (local time of GOC.IO); the bill may get paid only before the specified date and time, if it was set in original bill issuing request, otherwise there is no time limit for bill to be paid;
- *md5_hash* – MD5 hash calculated from the string consisting concatenated values of the variables: *from_USERID, to_USERID, currency, value, UNIQUEID, bill_valid*, and e-commerce venture's key (*pwd_bill*).

If values of variables provided in request are checked successfully by GOC.IO system, it will issue a bill and return a response that will be looking like following:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
         <result>0</result>
         <result_code>bill_registered</result_code>
         <BILLID>28023497238</BILLID>
</response>
```

The are following variables in the response:

- *result* – processing result code (0 – success, any other values means error);
- *result_code* – explanation of the processing result code in text (*bill_registered* means that the bill has been registered successfully and is ready to be paid by the user);
- *BILLID* – the ID of the registered bill in GOC.IO system.

If there are no user with specified ID in GOC.IO system at the moment the bill is being issued, GOC.IO will prepear user's record for following registration (i.e. it will register new user's record automatically and will keep it in "dormant" mode), and it will return in a response a special security code necessary for new user to complete the registration:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
         <result>0</result>
         <result_code>bill_registered</result_code>
         <BILLID>28023497238</BILLID>
         <to_pwd_renew_code>992849</to_pwd_renew_code>
</response>
```

The following variable will be added to the response:

- *to_pwd_renew_code* – alpha-numeric code that e-commerce venture should obligatory provide to the user who's receiving a new bill (it will be requested at the moment of registration in GOC.IO).

In case of an error (e.g. *md5_hash* has been calculated with a mistake) the following response may be received:

```
</response>
<?xml version="1.0" encoding="UTF-8"?>
<response>
         <result>-1</result>
         <result_code>md5_hash_wrong</result_code>
</response>
```

The are following variables in the response:

- *result* – processing result code (*-1* means error);
- *result_code* – explanation of the processing result code in text (*md5_hash_wrong* – the md5_hash variable's value is incorrect);

Customer's Payment Against Issued Bill

As soon as the bill will be issued to the user of the system via XML interface, this bill will be available immediately for payment in the system's web-interface. According to the current policy of GOC.IO, all of the bills should be issued in Bitcoins, or there is a possibility of issuing bills in Roubles.
In first case the payment against the bill is performed immediately when the user invokes payment procedure, and then the Bitcoin value is transferred from internal account of the user to internal account of the e-commerce venture.
If the bill is issued in Roubles, when the payment against the bill is invoked the bill gets blocked, and GOC.IO tries to sell the corresponding value of Bitcoins from user's internal account via exchange, then (if Bitcoins are sold successfully and the expected amount of Roubles has been reserved) e-commerce venture's internal Roubles account is being credited by the corresponding Roubles value.

Requesting the Information on Payment Status of the Bill

After the bill has been issued and during some period of time before the maximum time when the bill may get paid is reached, the e-commerce venture's system should check repeatedly if the bill has been paid or not, to provide to the user corresponding goods or services when the payment has been made.
It is recommended to check the status of the bill not more often than once in a minute during first hour after the bill is issued, then the delay between check requests should be increased so that the check will be performed once per hour, and then the bill status should be checked last time immediatelly after maximum time then payment is allowed has been reached.
There is no possibility for a bill to be paid after the maximum time set is reached.
The following request should be sent in order to check the status of the bill:

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
          <command>check_payment</command>
          <from_USERID>TEST_SHOP</from_USERID>
          <BILLID>28023497238</BILLID>
          <UNIQUEID></UNIQUEID>
          <payment_details></payment_details>
          <md5_hash>a658049e67333b216355ab936070cb64</md5_hash>
</request>
```

There are following variables in the request:

- *command* – command code (*check_payment*);
- *from_USERID* – ID of the user in GOC.IO system who has issued the bill;
- *BILLID* – ID of a bill in GOC.IO;
- *UNIQUEID* – unique ID of a bill in e-commerce venture's system (this variable may be used separately from *BILLID* or together with it);
- *payment_details* – if non-empty value has been provided (such as *«1»*), the payment information will be also included in the response in case the bill has been paid (see below).
- *md5_hash* – MD5 hash calculated from a string consisting the concatenated values of the following variables: *from_USERID, BILLID, UNIQUEID,* and e-commerce venture's key (*pwd_bill*).

In a response to the request GOC.IO will immediately provide the current status of the bill. For example, if the bill hasn't been paid yet, the response will look like following:

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

```
<response>
         <result>-1</result>
         <error>bill_isnt_paid</error>
         <error_details>28023497238;</error_details>
</response>
```

The are following variables in the response:

- *result* – processing result code (*-1* means error but in fact it's not a technical error that the user didn't pay against the bill yet);
- *error* – description of the processing result code in text (*bill_isnt_paid* – the bill hasn't been paid yet, this value will be returned as well in case the maximum payment time is reached and the bill hasn't been paid before it has occured);
- *error_details* – this variable consists of ID of the bill (*BILLID*) and unique ID according to e-commerce venture's system (*UNIQUEID*) separated by semicolon, enabling the user of the XML interface to check how exactly those IDs were seen by GOC.IO.

If the bill has been paid by the user, the reponse will contain the following variables:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
         <result>0</result>
         <result_code>check_payment</result_code>
         <payment_REFID>120230901748</payment_REFID>
</response>
```

The are following variables in the response:

- *result* – processing result code (0 – success);
- *result_code* – description of the processing result code in text (*check_payment* – command code from the request);
- *payment_REFID* – ID of the payment against corresponding bill in GOC.IO.

If the *payment_details* variable was set to non-empty value the response will look like following:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
         <result>0</result>
         <result_code>get_transfers_list</result_code>
         <payment_REFID>120230901748</payment_REFID>
         <payment_complete>1</payment_complete>
         <from_USERID>PAYEE_USERID</from_USERID>
         <currency>2</currency>
         <value>1200.00000000</value>
         <commission_value>12.00</commission_value>
         <commission_on_us>1</commission_on_us>
         <payment_complete_on>2013-11-12 15:09:33</payment_complete_on>
</response>
```

Additionally to the variables from the basic response there will be following variables:

- *payment_complete* – this variable keeps the value *1* if the payment has been completed successfully, it means that the account of e-commerce venture in GOC.IO has been credited by the corresponding sum of the payment (debited from user's account);
- *from_USERID* – ID of the user of GOC.IO who has paid the bill;
- *currency* – ID of the currency that was used during the payment (is equal to the currency of the bill), *1* for Bitcoins and *2* for Roubles;
- *value* – payment sum in the corresponding currency;
- *commission_value* – GOC.IO system's commission in the corresponsing currency; the sum received by the e-commerce venture is equal to *value–commission_value*;
- *commission_on_us* – indicates who was responsible for paying the GOC.IO' commission during the payment (*1* – commission is paid by e-commerce venture, *0*

or an empty value – commission is paid by the user paying the bill or not paid at all, the value of the *commission* variable in this case will be zero and it won't be available to the e-commerce venture).

- *payment_complete_on* – date and time of the completion of the bill payment in 'YYYY-MM-DD HH:MM:SS' format (local time of GOC.IO).

Issuing a Bill to the Customer of the System Using POST-Form

No key (*pwd_bill*) is required if this method is used, but in order to check the status of the bill the e-commerce venture may use the same features as described above. POST-form usage may be restricted by the user of the GOC.IO.

In order to issue a bill using this approach the following POST-form should be placed at the web-site of an e-commerce venture:

```
<FORM method=POST action='https://basic_system_hostname/'>
         GOC.io user ID:
         <INPUT type='text' name='USERID' value=''>
         <INPUT type='hidden' name='currency' value='RUR'>
         <INPUT type='hidden' name='price' value='500'>
         <INPUT type='hidden' name='seller' value='TEST_SHOP'>
         <INPUT type='hidden' name='product' value='Description of goods or services'>
         <INPUT type='hidden' name='success_url' value='http://my_shop.domain.com/success.html'>
         <INPUT type='hidden' name='error_url' value=' http://my_shop.domain.com/error.html '>
         <INPUT type='submit' name='buy' value='pay'>
</FORM>
```

There are following variables in the form:

- *USERID* – ID of the user in GOC.IO system (not a mandatory variable);
- *currency* – currency code that should be used in the bill (BTC or RUR);
- *price* – value of the bill in the corresponding currency;
- *seller* – ID of the user in GOC.IO issuing a bill (e-commerce venture's ID);
- *product* – description of goods and/or services exactly the same way as it will be shown to the user of GOC.IO paying a bill;
- *success_url* – URL that will be presented to user to be followed in case a bill will be paid successfully;
- *error_url* – URL that will be presented to user to be followed in case there will an error occur during the payment;
- *buy* – arbitrary variable used to submit a POST-form, not processed by GOC.IO (any variable is allowed or the form may be submitted without such a variable).

As soon as the user will be redirected to GOC.IO system in order to pay a bill, he or she will be prompted to authorize or to register, and the bill will be first thing that the user will see immediatelly after authorization.
The bill may be paid immediatelly (if there are funds available on user's account), or the user may decide to keep the bill for the later payment.

Numeric Format of GOC.io System

Decimal point («.») is used by GOC.IO as a delimeter of the integer and the fractional part of numbers, no special delimeters (such as spaces or apostrophes) should be used in order to split up particular digits of the numbers.
If the format of the numbers differs from described above, GOC.IO may response with an error message, considering incorrectly provided number as variable in 'string' format.