GOC.IO

ПЛАТЕЖНАЯ СИСТЕМА

API ЭЛЕКТРОННОЙ ТОРГОВОЙ СИСТЕМЫ

ВЕРСИЯ 1.20.00 **25.04.2015**

Предназначение и возможности API

Предназначение API

API служит для автоматизации операций в системе электронных торгов GOC.IO, эквивалентных по возможности операциям, доступным через веб-интерфейс системы в разделе «биржа».

При помощи API возможна автоматизация следующих процессов:

- создание и отмена заявок на покупку или продажу электронных валют в рамках пар BTC/RUR, LTC/RUR, LTC/BTC;
- получение информации о состоянии счета и открытых заявках, получение информации об операциях, связанных с работой в системе GOC.IO, получение информации о торговых операциях, проведенных в системе электронных торгов, просмотр текущих неисполненных (или исполненных частично) заявок, поданных как при помощи API, так и через веб-интерфейс системы.

Доступ к API

Для получения доступа к API пользователь системы GOC.IO должен через веб-интерфейс выполнить следующие действия:

1. Перейти в раздел «биржа».
2. Кликнуть ссылку «ключи API».
3. В открывшемся диалоге ввести условное наименование нового ключа API (используется только для того, чтобы пользователь в дальнейшем мог различать ключи API между собой) и нажать кнопку «добавить ключ API». Если у данного пользователя активизирован платежный пароль, необходимо для создания ключа API ввести также и платежный пароль.
4. После отображения созданного ключа API на экране в списке ключей API сохранить для последующего использования в автоматизированной системе располагающиеся в одной колонке друг под другом *ключ API* (выделен полужирным) и *секретный код ключа API* (обычный шрифт). Секретный код ключа API доступен для просмотра в веб-интерфейсе только в течение 15 мин. с момента создания, в дальнейшем в интерфейсе показываются только первые 4 знака секретного кода ключа API, чтобы предоставить возможность сориентироваться, верный ли секретный код используется.
5. Активировать, как описано далее в тексте, права на получение информации и на торговые операции для вновь созданного ключа API (на усмотрение пользователя – например, можно выделить ключи только для получения информации и ключи только для торговых операций).

API электронной торговой системы принимает запросы по адресу:

```
https://goc.io/api/
```

Параметры запроса могут передаваться интерфейсу API одним из нижеперечисленных способов:

- в переменных запроса POST или GET;
- в виде JSON-объекта, передаваемого непосредственно в теле запроса (при этом содержимое переменных POST/GET игнорируется и исключительными параметрами считаются полученные в JSON-объекте).

Максимально допустимая нагрузка, которую может создавать автоматизированное

клиентское решение при помощи данного API – 5 rps. При превышении данного значения доступ к API может быть заблокирован автоматически.

Тикеры

API поддерживаются следующие тикеры:

1. Информация об открытии торгов, текущем курсе и его последнем изменении, состоянии заявок ASK и BID (в соответствии с парами BTC/RUR, LTC/RUR, LTC/BTC, BTC/USD, LTC/USD, USD/RUR):

```
https://goc.io/api/btc_rur/ticker/
https://goc.io/api/ltc_rur/ticker/
https://goc.io/api/ltc_btc/ticker/
https://goc.io/api/btc_usd/ticker/
https://goc.io/api/ltc_usd/ticker/
https://goc.io/api/usd_rur/ticker/
```

Данные предоставляются по данному URL без авторизации в виде JSON-объекта со следующими полями:

```
{
"ticker":
        {
         "online":true,
         "high":25451.7601,
         "low":24910.1029,
         "avg":25180.9315,
         "vol":4036488.14986,
         "vol_cur":178.52999851,
         "last":25409.1700,
         "last_change":2.7029,
         "buy":25409.1700,
         "sell":25100.9812,
         "vol_24h":1029850.1960,
         "vol_cur_24h":21.6950989,
         "updated":1388266919,
         "server_time":1388309961
        }
}
```

Значимые поля JSON-объекта содержат следующую информацию:

- *online* – открыты ли торги по данной паре электронных валют;
- *high* – максимальное (обычно с момента открытия торгов) зарегистрированное значение курса;
- *low* – минимальное зарегистрированное значение курса;
- *avg* – среднее значение курса (обычно с момента открытия торгов);
- *vol* – объем торгов в расчетной электронной валюте с момента открытия торгов;
- *vol_cur* – объем торгов в базовой электронной валюте с момента открытия торгов;
- *last* – значение курса, с которым была зарегистрирована последняя сделка в системе электронных торгов;
- *last_change* – изменение курса относительно предыдущей сделки в системе электронных торгов;
- *buy* – минимальное значение ставок продажи (BID);
- *sell* – максимальное значение ставок покупки (ASK);
- *updated* – время последнего обновления информации в формате UNIXTIME;
- *vol_24h* – объем торгов в расчетной электронной валюте за истекшие 24 часа;

- *vol_cur_24h* – объем торгов в базовой электронной валюте за истекшие 24 часа;
- *server_time* – время сервера в формате UNIXTIME.

2. Информация о состоявшихся сделках:

```
https://goc.io/api/btc_rur/trades/
https://goc.io/api/ltc_rur/trades/
https://goc.io/api/ltc_btc/trades/
https://goc.io/api/btc_usd/trades/
https://goc.io/api/ltc_usd/trades/
https://goc.io/api/usd_rur/trades/
```

Данные предоставляются по одному из показанных URL без авторизации в виде JSON-объекта, представляющего собой массив обезличенных (т.е. не содержащих информацию непосредственно об участниках торговых сделок) объектов со следующими полями:

```
[
        {
                "date":1388261273,
                "price":18000,
                "amount":0.17,
                "tid":194,
                "price_currency":"RUR",
                "item":"BTC",
                "trade_type":"bid"
        },
        {
                "date":1388261273,
                "price":18000,
                "amount":0.83,
                "tid":193,
                "price_currency":"RUR",
                "item":"BTC",
                "trade_type":"bid"
        },
        {
                "date":1388261162,
                "price":18000,
                "amount":1,
                "tid":192,
                "price_currency":"RUR",
                "item":"BTC",
                "trade_type":"bid"
        }
]
```

Значимые поля объектов-элементов массива содержат следующую информацию:

- *date* – время регистрации сделки в формате UNIXTIME;
- *price* – курс, по которому зафиксирована данная сделка;
- *amount* – сумма сделки (валюта определяется полем *item*);
- *tid* – идентификатор сделки;
- *price_currency* – валюта котировки данной сделки;
- *item* – базовая валюта данной сделки (определяет сумму сделки в поле *amount*);
- *trade_type* – вид торговой операции, “*bid*” (продажа) или “*ask*” (покупка).

2. Информация об открытых торговых заявках:

```
https://goc.io/api/btc_rur/depth/
https://goc.io/api/ltc_rur/depth/
https://goc.io/api/ltc_btc/depth/
https://goc.io/api/btc_usd/depth/
https://goc.io/api/ltc_usd/depth/
https://goc.io/api/usd_rur/depth/
```

Данные предоставляются по каждому из URL в списке без авторизации в виде JSON-объекта,

представляющего собой массив объектов со следующими полями:

```
{
        "asks":
                [
                        [24100,10,2],
                        [20000,2,1]
                ],
        "bids":
                [
                        [24900,6.43,1],
                        [25100,10,1]
                ]
}
```

JSON-объект состоит из двух полей, *asks* и *bids*, каждое из которых представляет собой массив, состоящий из элементов-массивов с ценовыми предложениями в категориях заявок ASK (заявки на покупку) и BID (заявки на продажу) в формате *[цена,сумма,число_заявок]*.

Права для ключей API

С использованием веб-интерфейса пользователь системы GOC.IO должен назначить права для каждого вновь созданного ключа.

Права разделяются на обеспечивающие информационные функции (“информация”), т.е. на такие, при помощи которых никаких изменений в системе электронных торгов не производится, а также на торговые функции (“торги”), при помощи которых в системе электронных торгов создаются и отзываются заявки.

Для изменения прав для ключей API пользователь системы GOC.IO должен через веб-интерфейс выполнить следующие действия:

1. Перейти в раздел «биржа».
2. Кликнуть ссылку «ключи API».
3. В открывшемся списке напротив наименования нужного ключа можно увидеть, предоставлены ли данному ключу права на использование информационных и торговых методов API. Если ключу API предоставлены права на использование информационных методов, в колонке «информация» в строке напротив наименования данного ключа будет отображен символ «✅» – или символ «❌», если такие права пользователем не предоставлены. Аналогично символ в колонке «торги» отображает возможность ключа на использование торговых методов.
4. Изменение прав осуществляется путем клика на соответствующий запрещающий или разрешающий символ. При этом его значение меняется на противоположное («разрешено» при клике на «запрещено», и наоборот). Если у данного пользователя активизирован платежный пароль, для изменения прав ключей API необходимо также ввести платежный пароль в поле для создания новых ключей, в противном случае изменение не регистрируется. В системе API изменения вступают в силу немедленно.

Деактивация ключей API

Если по каким-либо причинам ключ API должен быть выведен из эксплуатации, пользователь системы должен предпринять следующие действия:

1. Перейти в раздел «биржа».
2. Кликнуть ссылку «ключи API».
3. В открывшемся списке кликнуть символ «❌» в колонке «деактивировать» в строке напротив наименования нужного ключа API, при этом соответствующий ключ будет деактивирован немедленно и навсегда пропадет из списка ключей. Ключ может быть

деактивирован и без использования платежного пароля, даже если для данного пользователя платежный пароль активизирован. В системе API изменения вступают в силу немедленно.

Деактивированные ключи в списке ключей не отображаются, и их действительность в дальнейшем не возобновляется. Система предотвращает возможность создания в дальнейшем новых ключей API, соответствующих ранее деактивированным.

Совместимость

Методы API в целом совместимы с используемыми в основных системах электронной торговли, оперирующими криптовалютами, такими, как btc-e.com.

Общие сведения об API

Подпись при вызове методов API

Для формирования подписи запроса из параметров, входящих в запрос (исходя из особенностей конкретного метода), и их значений должна быть составлена строка, аналогичная строке параметров в POST-запросе протокола HTTP.

Например, если в запросе используются параметры *method=TradeHistory* и *nonce= 1388317502*, строка для формирования подписи должна выглядеть следующим образом:

```
method=TradeHistory&nonce=1388317502
```

Подпись формируется при помощи алгоритма SHA-512 функцией прикладной системы разработки *hash_hmac()* или аналогичной с использованием ранее полученного для данного ключа API секретного кода, например, так:

```
sign=hash_hmac('sha512',request_data,api_secret)
```

В данном вызове функции *'sha512'* – наименование используемого алгоритма, *request_data* – условная переменная, в которой хранится строка для формирования подписи, и *api_secret* – секретный код ключа API.

После того, как подпись сформирована, заголовок HTTP-запроса к API дополняется двумя параметрами с данным ключом API и вычисленной подписью:

```
Key: ключ API
Sign: сформированная подпись для данного запроса
```

Запрос обрабатывается API немедленно в синхронном режиме, и запрашивающая система сразу же после завершения получает JSON-объект с результатом обработки в ответе сервера.

Параметр *nonce*

В каждый запрос должен быть включен цифровой параметр *nonce* (целое число), самостоятельно увеличиваемый запрашивающей системой не менее чем на единицу для каждого нового запроса, передаваемого с использованием того же ключа API (т.е. инкрементный ключ).

API не проверяет значение параметра *nonce* при первом запросе для данного ключа API (считая, что для API данный параметр отсутствует), однако после того, как параметр *nonce* был передан API впервые, в случае, если в следующем запросе будет передано такое же или меньшее значение параметра *nonce*, это вызовет сообщение об ошибке.

Ошибки при исполнении запросов API

В общем случае если при исполнении запросов API возникает какая-либо ошибка, информация об этом передается в ответе в формате:

```
{
"success":0,
"error":"описание ошибки"
}
```

Как правило, причиной возникновения ошибки является недостаток прав данного ключа API, некорректное использование каких-либо параметров запроса, некорректное наименование методов или параметров запроса, неверное вычисление подписи запроса, неверное значение параметра *nonce* и пр.

Методы API

Метод *getInfo*

Тип метода: *информационный* (права «информация»)

Предназначение: получение информации о состоянии торгового счета и ключей API.

Здесь и далее параметры запроса будут показаны в виде JSON-объекта. При необходимости аналогичные параметры могут передаваться пользовательским приложением в виде параметров POST или GET-запроса.

Параметры запроса:

```
{
"method":"getInfo",
"nonce":"1"
}
```

Описание полей запроса:

- *method* – наименование вызываемого метода (“getInfo”);
- *nonce* – параметр *nonce*.

Пример ответа:

```
{
"success":1,
"return":
        {
                "funds":
                        {
                        "btc":38.29193648,
                        "ltc":490.1209,
                        "rur":285581.62154,
                        "usd":60.2917,
                        },
                "rights":
                        {
                        "info":1,
                        "trade":1,
                        "withdraw":0
                        },
                "transaction_count":185,
                "open_orders":2,
                "server_time":1388319654
        }
}
```

Описание полей ответа:

- *success* – “1” означает «успешное исполнение запроса»;
- *return* – объект, возвращаемый пользователю API, содержащий следующие

включенные объекты:

- *funds* – информация о наличии средств на электронных счетах пользователя системы, выпустившего данный ключ API, в виде объекта со значениями остатков в формате *“код валюты”:остаток*;
- *rights* – информация о правах, предоставленных данному ключу API, в виде объекта со значениями прав «info» (информация), «trade» (торги) и «withdraw» (возможность вывода средств), значение «0» означает отсутствие прав, значение «1» – наличие;
- *transaction_count* – число торговых операций, выполненных данным пользователем (с использованием всех ключей API);
- *open_orders* – число торговых заявок, открытых для данного пользователя (с использованием всех ключей API);
- *server_time* – время сервера в формате UNIXTIME.

Метод *TransHistory*

Тип метода: *информационный* (права «информация»)

Предназначение: получение списка переводов данного пользователя в системе, имеющих отношение к торговым операциям.

Параметры запроса:

```
{
"method":”TransHistory”,
“from”:”10”,
“count”:”25”,
“from_id”:”10”,
“end_id”:”20”,
“order”:”ASC”,
“since”:”1388319654”,
“end”:”1388319654”,
"nonce":”1”
}
```

Описание полей запроса:

- *method* – наименование вызываемого метода (“TransHistory”);
- *from* – номер строки ответа, с которой должен быть начат список (необязательный параметр);
- *count* – максимальное число платежей в ответе (необязательный параметр, по умолчанию число платежей в ответе ограничено числом 1000);
- *from_id* - идентификатор платежа, с которого должен быть начат список (необязательный параметр);
- *end_id* – идентификатор платежа, на котором может быть закончен список (необязательный параметр);
- *order* – последовательность идентификаторов платежей в ответе (*ASC* по возрастанию, *DESC* по убыванию, необязательный параметр, значение по умолчанию *DESC*);
- *since* – дата, начиная с которой должны выводиться платежи в списке (необязательный параметр, формат UNIXTIME);
- *end* – дата, до которой должны выводиться платежи в списке (необязательный параметр, формат UNIXTIME);

- *nonce* – параметр *nonce*.

Пример ответа:


```
{
"success":1,
"return":
        {
        "1649":
                {
                "type":4,
                "amount":18000,
                "currency":"RUR",
                "desc":"ORDER 323 RETURN: RUR 18000.00000000",
                "status":2,
                "timestamp":1388266919
                },
        "1645":
                {
                "type":5,
                "amount":1,
                "currency":"BTC",
                "desc":"BTC 1 -> RUR 50000",
                "status":2,
                "timestamp":1388263814
                }
        }
}
```


Описание полей ответа:

- *success* – “1” означает «успешное исполнение запроса»;
- *return* – объект, возвращаемый пользователю API, содержащий включенные объекты, ключом каждого из которых является идентификатор платежа, а значением – объект, состоящий из параметров платежа:
    - *type* – код типа платежа, который может принимать следующие значения:
        - *1* – операция зачисления средств на счет пользователя,
        - *2* – операция перевода средств со счета пользователя,
        - *4* – торговая операция, связанная с зачислением средств на счет пользователя (например, возврат или перевод средств на счет пользователя в результате завершения сделки),
        - *5* – торговая операция, связанная со списанием средств со счета пользователя (перевод обеспечения);
    - *amount* – сумма платежа;
    - *currency* – код электронной валюты платежа (“BTC”, “LTC“, “RUR” или “USD”);
    - *desc* – текстовое описание платежа;
    - *status* – статус платежа, который может принимать следующие значения:
        - *0* – платеж отменен (не активен и не ожидает подтверждения);
        - *1* – ожидание завершения платежа (платеж активен и ожидает подтверждения);
        - *2* – платеж завершен (платеж активен и не ожидает подтверждения);
    - *timestamp* – время завершения платежа в формате UNIXTIME.

Метод *TradeHistory*

Тип метода: *информационный* (права «информация»)

Предназначение: получение истории торговых операций (сделок), выполненных в системе данным пользователем с использованием любых ключей API.

Параметры запроса:

```
{
"method":”TradeHistory”,
“from”:”10”,
“count”:”25”,
“from_id”:”10”,
“end_id”:”20”,
“from_order_id”:”1000”,
“end_order_id”:”2000”,
“order”:”ASC”,
“since”:”1388319654”,
“end”:”1388319654”,
“pair”:”btc_rur”,
"nonce":”1”
}
```

Описание полей запроса:

- *method* – наименование вызываемого метода (“TradeHistory”);
- *from* – номер строки ответа, с которой должен быть начат список (необязательный параметр);
- *count* – максимальное число сделок в ответе (необязательный параметр, по умолчанию число платежей в ответе ограничено числом 1000);
- *from_id* - идентификатор сделки, с которой должен быть начат список (необязательный параметр);
- *end_id* – идентификатор сделки, на которой должен быть закончен список (необязательный параметр);
- *from_order_id* - идентификатор заявки, участвовавшей в сделке, с которой должен быть начат список (необязательный параметр, но в случае применения должен также использоваться параметр *end_order_id*);
- *end_order_id* – идентификатор заявки, участвовавшей в сделке, на которой должен быть закончен список (необязательный параметр, но применяется только совместно с параметром *from_order_id*);
- *order* – последовательность идентификаторов сделок в ответе (*ASC* по возрастанию, *DESC* по убыванию, необязательный параметр, значение по умолчанию *DESC*);
- *since* – дата, начиная с которой должны выводиться сделки в списке (необязательный параметр, формат UNIXTIME);
- *end* – дата, до которой должны выводиться сделки в списке (необязательный параметр, формат UNIXTIME);
- *pair* – код пары электронных валют, для которой следует передать историю сделок (“*btc_rur*” по умолчанию);
- *nonce* – параметр *nonce*.

Пример ответа:

```
{
"success":1,
"return":
          {
          "194":
                    {
                    "pair":"btc_rur",
                    "type":"sell",
```

```
                "amount":"0.17000000",
                "rate":18000,
                "order_id":"317",
                "is_your_order":1,
                "timestamp":1388261273
                },
        "193":
                {
                "pair":"btc_rur",
                "type":"sell",
                "amount":"0.83000000",
                "rate":18000,
                "order_id":"317",
                "is_your_order":1,
                "timestamp":1388261273
                },
        "192":
                {
                "pair":"btc_rur",
                "type":"sell",
                "amount":"1.00000000",
                "rate":18000,
                "order_id":"315",
                "is_your_order":1,
                "timestamp":1388261162
                }
        }
}
```


Описание полей ответа:

- *success* – “1” означает «успешное исполнение запроса»;
- *return* – объект, возвращаемый пользователю API, содержащий включенные объекты, ключом каждого из которых является идентификатор сделки, а значением – объект, состоящий из параметров сделки:
    - *pair* – код пары электронных валют, для которых зарегистрирована сделка (“btc_rur”, “ltc_rur” или “ltc_btc”);
    - *type* – тип сделки, “sell” продажа или “buy” покупка;
    - *amount* – сумма базовой электронной валюты сделки (следует первой в параметре *pair*);
    - *rate* – курс, по которому была осуществлена сделка;
    - *order_id* – идентификатор заявки, по которой зарегистрирована сделка;
    - *is_your_order* – показывает, была ли заявка, идентификатор которой передан в поле *order_id*, подана пользователем с данным ключом API (значение “1”), либо другим пользователем (значение “0”);
    - *timestamp* – время регистрации сделки (формат UNIXTIME).

Метод *ActiveOrders*

Тип метода: *информационный* (права «информация»)

Предназначение: получение списка активных (не закрытых) торговых заявок, созданных в системе при помощи API либо веб-интерфейса.

Параметры запроса:


```
{
"method":”ActiveOrders”,
“pair”:”btc_rur”,
"nonce":”1”
}
```


Описание полей запроса:

- *method* – наименование вызываемого метода (“ActiveOrders”);
- *pair* – код пары электронных валют, для которой следует передать действующие заявки;
- *nonce* – параметр *nonce*.

Пример ответа:

```
{
"success":1,
"return":
        {
        "288":
                {
                "pair":"btc_rur",
                "type":"buy",
                "amount":10,
                "remains":9.2094,
                "rate":24100,
                "timestamp_created":1388168475,
                "status":0
                }
        }
}
```

Описание полей ответа:

- *success* – “1” означает «успешное исполнение запроса»;
- *return* – объект, возвращаемый пользователю API, содержащий включенные объекты, ключом каждого из которых является идентификатор заявки (*order_id*), а значением – объект, состоящий из параметров заявки:
    - *pair* – код пары электронных валют, для которых зарегистрирована заявка;
    - *type* – тип заявки, “buy” продажа или “sell” покупка;
    - *amount* – сумма заявки в базовой валюте (первая в списке в коде *pair*);
    - *remains* – сумма остатка по данной заявке в базовой (остаток может отличаться от суммы заявки, если заявка была частично удовлетворена встречными заявками);
    - *rate* – курс заявки;
    - *timestamp_created* – время создания заявки в формате UNIXTIME;
    - *status* – статус данной заявки (0 – ожидает исполнения).

Метод *OrderInfo*

Тип метода: *информационный* (права «информация»)

Предназначение: получение информации по активной (не закрытой) торговой заявке, идентифицируемой по номеру.

Параметры запроса:

```
{
"method":”OrderInfo”,
“order_id”:”12345”,
"nonce":”1”
}
```

Описание полей запроса:

- *method* – наименование вызываемого метода (“OrderInfo”);
- *order_id* – идентификатор заявки.

- *nonce* – параметр *nonce*.

Описание полей ответа: соответствует методу *ActiveOrders*.

Метод *Trade*

Тип метода: *торговый* (права «торги»)

Предназначение: создание в системе торговых заявок.

Параметры запроса:

```
{
"method":”Trade”,
“pair”:”btc_rur”,
“type”:”buy”,
“amount”:1,
“rate”:20000,
“fok”:false,
“ioc”:false,
"nonce":”1”
}
```

Описание полей запроса:

- *method* – наименование вызываемого метода (“Trade”);
- *pair* – код пары электронных валют, для которой создается заявка;
- *type* – тип заявки, “buy” для продажи, “sell” для покупки;
- *amount* – сумма заявки в базовой валюте заявки (первая в списке из кода *pair*);
- *rate* – курс, по которой регистрируется заявка;
- *fok* – признак, определяющий, что заявка должна быть исполнена с политикой FOK (*Fill Or Kill*, другое название *All Or None*): заявка может быть исполнена исключительно в указанном объеме, причем присутствующем в одной встречной заявке. Если на рынке в данный момент не присутствует достаточного объема финансового инструмента, заявка будет отменена.
- *ioc* – признак, определяющий, что заявка должна быть исполнена с политикой IOC (*Immediate Or Cancel*): в данном случае трейдер соглашается совершить сделку по максимально доступному на рынке объему в пределах указанного в заявке. В случае невозможности полного исполнения заявка будет исполнена в пределах доступного объема, а неисполненный объем заявки будет отменен.
- *nonce* – параметр *nonce*.

Пример ответа:

```
{
"success":1,
"return":
        {
        "received":1,
        "remains":1,
        "order_id":327,
        "funds":
                {
                "btc":38.29193648,
                "ltc":0.12948928,
                "rur":507411.62154
                }
        }
}
```

Описание полей ответа:

- *success* – “1” означает «успешное исполнение запроса»;

- *return* – объект, возвращаемый пользователю API, содержащий следующие включенные объекты:
    - *received* – сумма, предъявленная в запросе на регистрацию заявки;
    - *remains* – остаток от суммы, предъявленной в запросе на регистрацию заявки;
    - *order_id* – идентификатор зарегистрированной заявки;
    - *funds* – информация о наличии средств на электронных счетах пользователя системы, выпустившего данный ключ API, в виде объекта со значениями остатков в формате *“код валюты”:остаток*;

Метод *CancelOrder*

Тип метода: *торговый* (права «торги»)

Предназначение: отмена торговых заявок, созданной с использованием API либо веб-интерфейса.

Параметры запроса:

```
{
"method":”CancelOrder”,
“order_id”:”111”,
"nonce":”1”
}
```

Описание полей запроса:

- *method* – наименование вызываемого метода (“CancelOrder”);
- *order_id* – идентификатор отменяемой заявки (обязательный параметр);
- *nonce* – параметр *nonce*.

Пример ответа:

```
{
"success":1,
"return":
        {
        "order_id":"111",
        "funds":
                {
                "btc":38.29193648,
                "ltc":0.12948928,
                "rur":527521.62154
                }
        }
}
```

Описание полей ответа:

- *success* – “1” означает «успешное исполнение запроса»;
- *return* – объект, возвращаемый пользователю API, содержащий следующие включенные объекты:
    - *order_id* – идентификатор отмененной заявки;
    - *funds* – информация о наличии средств на электронных счетах пользователя системы, выпустившего данный ключ API, в виде объекта со значениями остатков в формате *“код валюты”:остаток*;

Исполнение заявок

Система пытается исполнить заявку (т.е. найти “встречную” заявку и провести сделку)

немедленно в момент регистрации, однако в зависимости от текущей нагрузки исполнение может быть как немедленным, так и занять некоторое время (от нескольких секунд до минуты).

Заявка может быть исполнена как полностью, так и частично, при наличии (регистрации) в электронной торговой системе “встречной” заявки (“buy” для “sell” или “sell” для “buy”) по соответствующему курсу (курс покупки выше или равен курсу продажи для той же самой пары электронных валют).

Полное закрытие заявки происходит, если объем встречной заявки достаточен для исполнения данной (например, при подаче заявки на покупку 1 BTC по курсу 20000 RUR присутствует заявка на продажу 10 BTC по курсу 20000 RUR или ниже). При этом заявка пропадает из информации в ответе на запрос *ActiveOrders*, но сохраняется в виде сделки в ответе на запрос метода *TradeHistory*.

Частичное закрытие заявки происходит, если объем встречной заявки недостаточен, при этом происходит закрытие по меньшей сумме из остатков данной заявки и встречной заявки (например, при подаче заявки пользователем *A* на покупку 1 BTC по курсу 20000 RUR в системе присутствует заявка пользователя *B* на продажу 0.3 BTC по курсу 20000 RUR, при этом поданная пользователем *A* заявка закрывается в объеме 0.3 BTC и образуется остаток 0.7 BTC, а поданная пользователем *B* заявка закрывается полностью). При этом в информации в ответе на запрос метода *ActiveOrders* поданная заявка сохраняется, и она может быть отменена при помощи метода *CancelOrder* – в этом случае на счет пользователя *A* будет возвращен остаток средств, зарезервированных для покупки. В информации в ответе на запрос метода *TradeHistory* пользователя *A* (и пользователя *B*) при этом будет отражена сделка на продажу/покупку 0.3 BTC по курсу 20000 RUR.

В любом случае признаком однозначным признаком исполнения заявки является отсутствие информации о ней в ответе на запрос метода *ActiveOrders* при наличии информации в ответе на запрос метода *TradeHistory*.

Расчетные операции

Расчеты по торговым операциям, т.е. зачисления на счета пользователей в результате регистрации сделок по ранее поданным заявкам, производятся в течение максимум одной минуты с момента регистрации сделки.

Числовой формат системы GOC.io

В системе GOC.IO, в т.ч. в API, в качестве разделителя целой и дробной частей действительных (вещественных) чисел используется десятичная точка («.»), отдельные разряды целой части действительных чисел не разделяются какими-либо знаками («пробел», «апостроф» и т.п.).
В случае использования формата действительных чисел, отличающихся от описанного, система GOC.IO вернет сообщение об ошибке, распознав число в некорректном формате как строку.