**数字货币多交易所行情及交易网关使用说明（V.2018-0621）**

**目录**

[1.行情部分 2](#_Toc7460)

[1.1 运行环境 2](#_Toc11944)

[1.2 参数配置 2](#_Toc9234)

[1.2.1 配置TBotApi.json 2](#_Toc8997)

[1.2.2 配置variables.json 3](#_Toc12398)

[1.2.3 配置keywords.json 3](#_Toc12221)

[1.2.4 配置exchange\_symbols.json 4](#_Toc10304)

[1.2.5 配置exchange\_futures.json 4](#_Toc22419)

[1.2.6 配置contract\_type.json 5](#_Toc19061)

[1.3常见问题解答（FAQ） 5](#_Toc19693)

[1.3.1 如何增加交易所 5](#_Toc23327)

[1.3.2 如何指定交易所 5](#_Toc15085)

[1.3.2 如何修改订阅品种 6](#_Toc23236)

[1.3.3 如何将网关运行多个实例 6](#_Toc25131)

[1.3.4 如何修改订阅数据的深度 6](#_Toc26915)

[2.交易部分 7](#_Toc18148)

[2.1 运行环境 7](#_Toc14099)

[2.2 参数配置 7](#_Toc26078)

[2.2.1 配置rest\_urls.json 7](#_Toc18538)

[2.2.2 配置acc\_key.json 7](#_Toc19760)

[2.2.3 配置trade.json 7](#_Toc3598)

[2.2.4 配置代码映射表symbols\_map.json 9](#_Toc8338)

[2.2.5 配置程序运行参数 9](#_Toc24991)

[2.3 交易测试 9](#_Toc3295)

蓝目数据（http://www.blueye.info）

2018-06

## 1.行情部分

### 1.1 运行环境

行情网关环境，如果在windows下，有32位和64位两种环境：

如果运行32位网关程序，则需要安装vs2010环境，以及boost1.59和openssl1.0.0.s这些库。

如果运行64位网关程序，则需要安装VS2017环境，以及boost1.66以上和openssl1.0.0.s以上这些库。

有关boost和openssl的安装，请参阅其官方网站。

注意：不同版本的boost和openssl, 其对应的编译和运行环境都不一样，一定要仔细看安装文档。

(linux环境安装后续再添加)。

行情网关启动程序压缩包为TBotApi\_bin.rar，直接解压缩到一个盘符的目录下，运行程序为TBotApi.exe，创建一个快捷方式到桌面。

### 1.2 参数配置

启动TBotApi.exe之前，需要做如下配置：

#### 1.2.1 配置TBotApi.json

交易所行情及其对应的地址和请求参数，在TBotApi.json文件里面。每个交易所为一个json对象，里面分别有如下4项：  
 "ticker":现货实时行情；"depth"：现货深度数据；"future\_ticker":期货实时行情；"future\_depth":期货深度数据。

每个项目对应有”url” 和”body”，根据交易所的具体API进行填写。

配置举例如下：

"okex":

{

"ticker":{"url":"wss://real.okex.com:10441/websocket","body":{"event":"addChannel","channel":"ok\_sub\_spot\_$symbol\_ticker"}},

"depth":{"url":"wss://real.okex.com:10441/websocket","body":{"event":"addChannel","channel":"ok\_sub\_spot\_$symbol\_depth"}},

"future\_ticker":{"url":"wss://real.okex.com:10441/websocket","body":{"event":"addChannel","channel":"ok\_sub\_futureusd\_@X\_ticker\_@Y"}},

"future\_depth":{"url":"wss://real.okex.com:10441/websocket","body":{"event":"addChannel","channel":"ok\_sub\_futureusd\_@X\_depth\_@Y\_200"}}

},

#### 1.2.2 配置variables.json

每个行情程序对应一个variables.json,启动参数指定对应的文件。如果启动命令不带参数，默认使用variables.json作为配置启动，如果带参数，则为另外一个variable\_xxx.json(xxx为通配符)参数文件被加载。

Variables.json的各个项目释义如下：

"exchange":"kraken" , //指定当前交易所

"$symbol":"eth\_btc" , //指定需要订阅的品种，如果是”\*”，则从exchange\_symbols里面获取该交易所的所有品种

"action":"depth" , //该项目暂时保留

"$depth\_len":"8" , //该项目暂时保留

"$huobi\_depth\_type":"step4" , //该项目暂时保留

"$huobi\_client\_id":"id1" , //该项目暂时保留

"write\_message\_queue":"1", //是否写入共享内存，1写入，0不写入

"ticker\_message\_queue\_name":"ticker\_data", //ticker共享消息队列名称

"ticker\_max\_message\_queue\_size":"206" , //ticker共享消息队列长度

"ticker\_max\_message\_queue\_number":"1024" , //ticker共享消息队列最大个数

"depth\_message\_queue\_name":"depth\_data", //depth共享消息队列名称

"depth\_max\_message\_queue\_size":"2048" , //depth共享消息队列长度

"depth\_max\_message\_queue\_number":"1024" , //depth共享消息队列最大个数

"channel\_width":"30", //标识字符宽度

"format\_integer\_width":"8", //默认整数部分长度

"format\_decimal\_width":"8", //默认小数部分长度

"bids\_asks\_depth":"50", //深度数据每传输包记录数

"max\_string\_buffer\_size":"10000", //网络传输包字节长度

"max\_message\_count\_in\_queue":"100", //最大消息缓冲

"message\_gap\_in\_milliseconds":"100", //消息发送间隔

"console\_show":"0", //是否控制台显示，‘1’显示；‘0’不显示

"tcp\_port":"1818" //行情服务端TCP端口号

#### 1.2.3 配置keywords.json

每个交易所必须在keywords.json文件里面配置一个对应的对象，内容如下（以okex为例）：

"okex":

{

"$api\_key":"api\_key",

"$sign":"sign",

"$exchange":"exchange",

"$symbol":"symbol",

"$response\_tag":"channel",

"$ticker\_channel":"ok\_sub\_spot\_$symbol\_ticker",

"$depth\_channel":"ok\_sub\_spot\_$symbol\_depth",

"$future\_ticker\_channel":"ok\_sub\_futureusd\_@X\_ticker\_@Y",

"$future\_depth\_channel":"ok\_sub\_futureusd\_@X\_depth\_@Y\_200",

"$ticker\_object":"data",

"$last":"last",

"$open":"open",

"$high":"high",

"$low":"low",

"$close":"close",

"$buy":"buy",

"$vol":"vol",

"$amount":"amount",

"$sell":"sell",

"$bids":"bids",

"$asks":"asks",

"$limitHigh":"limitHigh",

"$limitLow":"limitLow",

"$unitAmount":"unitAmount",

"$hold\_amount":"hold\_amount",

"$contractId":"contractId",

"$timestamp":"timestamp",

"$depth\_timestamp":"timestamp",

"$order\_type":"type",

"$order\_id":"order\_id",

"$trans\_id":"tid"

},

这些项目是交易所API和网关直接的一个关键词映射表，主要是在请求参数和应答数据之间的名称映射。这部分内容建议由开发人员和维护人员一起来配置。除非交易所接口API升级，否则不要轻易修改这部分内容。

#### 1.2.4 配置exchange\_symbols.json

该文件里面保存了每个交易所的货币品种对用来作为行情订阅，如果在variables.json里面把”$symbols”项目配置成”\*”,则会从exchange\_symbols.json文件里面读取对应交易所下面的所有货币对发送到交易所进行订阅。

#### 1.2.5 配置exchange\_futures.json

这个文件里面保存了所有交易所对应的期货订阅品种，如下：

{

"okex":"btc,ltc,eth,etc,bch,xrp,eos,btg" ,

"zb":"" ,

"huobi":"" ,

"binance":"" ,

"bitfinex":"",

"kraken":""

}

#### 1.2.6 配置contract\_type.json

该文件保存了所有交易所的期货对应的合约类别，如下所示：

{

"okex":"this\_week,next\_week,quarter" ,

"zb":"" ,

"huobipro":"" ,

"binance":"" ,

"bitfinex":""

}

### 1.3常见问题解答（FAQ）

#### 1.3.1 如何增加交易所

按照如下步骤，配置文件：

1. 在TBotApi.json里面，参照其它已经配置好的交易所对象，增加一个交易所对象，并对里面的每个地址和请求相关参数做修改。
2. 在keywords.json文件里面增加交易所对象，参照其它交易所对每个新增项目进行赋值。
3. 在exchange\_symbols.json里面增加该交易所的所有现货品种。
4. 在exchange\_futures.json里面增加该交易所的所有期货品种（如果存在的话）。
5. 在contract\_type.json里面增加该交易所的期货合约的类别（如果存在的话）。

#### 1.3.2 如何指定交易所

该网关程序默认只支持一个交易所的行情品种订阅，如果需要改变交易所，只需要在variables.json里面，修改”exchange” 项目，指向新的交易所（该交易所必须按照1.3.1的步骤增加）。

#### 1.3.2 如何修改订阅品种

每个交易所的需要订阅的品种在variables.json文件里面定义，项目为”$symbol”，如果改部分指定单个或者多个货币对（中间用逗号分隔），则从这里进行订阅；如果该值指定为”\*”，则从exchange\_symbols.json里面进行订阅。

#### 1.3.3 如何将网关运行多个实例

由于每个交易所全部订阅数量比较大，所以网关程序默认只能运行一个交易所的数据，包含ticker和depth。但是实际情况，可能会需要在一台主机上配置多个交易所的订阅，那么这时候就需要把网关程序启动运行多份。步骤如下：

1. 创建多个variables.json文件，比如variables\_01.json, variables\_02.json,…
2. 每个不同的variables\_xx.json（这里xx代表通配符）里面，用exchange项目,tcp\_port等等字段进行个性化配置。
3. 启动网关的时候，后面带参数，每个参数执行不同的variables\_xx.json（这里xx代表通配符）文件。例如：

TBotApi variables\_01.json,

TBotApi variables\_02.json, …

4，行情订阅客户端程序只需要在程序中指定不同的配置文件variables\_xx.json，即可实现多交易所的行情订阅和接收。

#### 1.3.4 如何修改订阅数据的深度

大部分交易所的深度数据订阅返回的深度是相对固定的，但是有些交易所可能需要用户在订阅的时候把深度当作参数化参入进去，比如OKXE交易所，订阅的时候需要参数三个标量X,Y,Z,分别代表货币名称、合约类别和深度，举例：

订阅参数 ”ok\_sub\_spot\_$symbol\_depth”代表订阅某品种($symbol)代表某品种，的现货深度数据，深度由交易所决定。如果在订阅参数后面增加“\_Z”（Z代表某个1-500的数字）,则会按照用户的订阅深度返回，并且，这个数字越大，数据返回的频率会越低（间隔时间变长）。

同理，对于期货合约深度订阅参数"ok\_sub\_futureusd\_@X\_depth\_@Y\_Z",其中@X代表品种，@Y代表合约类别，“\_Z”（Z代表某个1-500的数字）。

另外，修改深度的时候，需要分别在TBotApi.json 和 keywords.json里面，对于这个订阅字符串修改一致，这样重启程序才有效果。

## 2.交易部分

### 2.1 运行环境

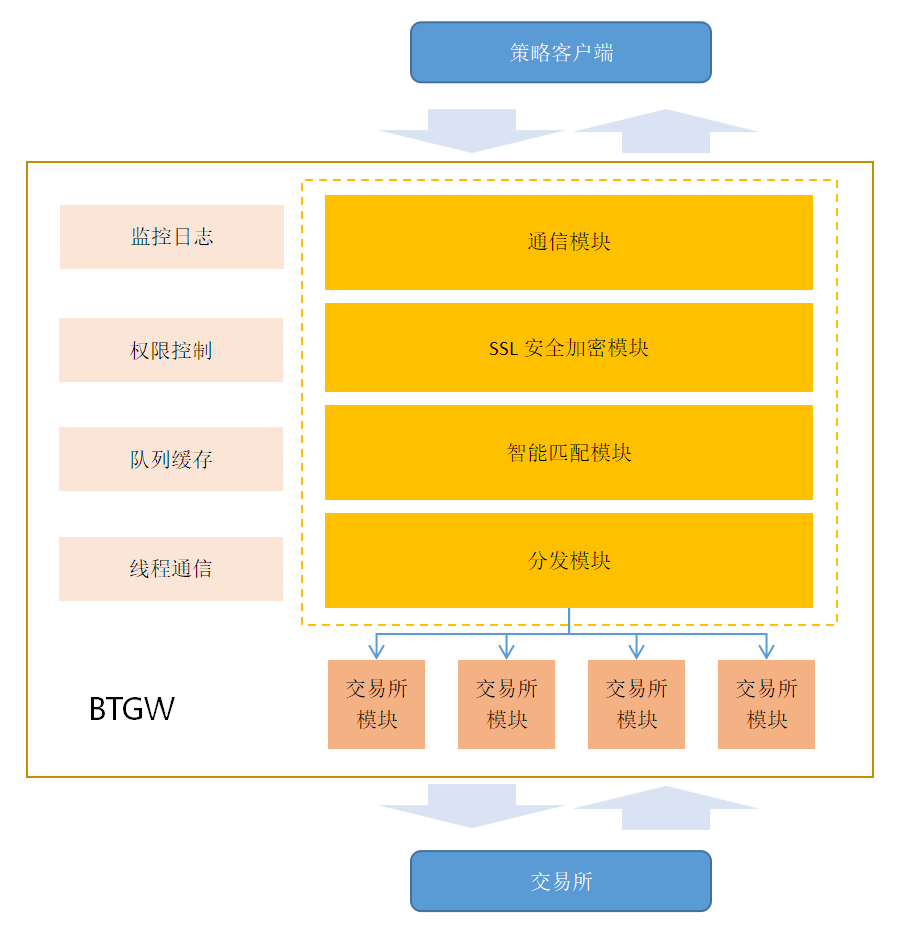
交易网关BTGW，安装包为BTGW\_bin.rar。解压到工作目录下，主程序为BTGW.exe。主目录下有程序运行所需要的动态链接库和主配置文件，有exchanges子目录，下面有各个交易所的配置文件和动态库。

交易网关与上层策略端之间采用https加密协议建立信息安全通道，来保证数据传输的安全性。使用数字证书、数字摘要以及时间戳技术确保交易真实性，防止内容被第三方冒充及篡改，防止重放攻击。

每个策略客户端需要申请一个token，作为连接交易网关的凭据，服务器后台可对其生命周期进行管理。

策略端每次请求的请求ID(req\_id)字段是timestamp，精确到毫秒级，必须确保递增，用来限制重发。

交易网关逻辑结构图



### 2.2 参数配置

启动BTGW.exe之前，需要做如下信息的配置：

#### 2.2.1 配置acc\_key.json

配置账户及api\_key。打开文件acc\_key.json,配置账户及交易 api\_key和secret\_key，数据样本如下：

"0001":"okex,api\_keyapi\_keyapi\_keyapi\_keyapi\_key,secret\_keysecret\_keysecret\_keysecret\_keysecret\_key" ，

其中第一项为虚拟账户，用户自己设定；第二项是由交易所名称、apyi\_key和secret\_key依次组成的字符串，中间用逗号分隔。

#### 2.2.2 配置程序运行参数variables.json

网关的默认连接TCP端口是1819。可以在启动网关时用命令行参数的形式指定网关运行参数文件，例如：BTGW.EXE variables.json。参数文件配置格式即注释如下（实际使用要删掉注释）：

{

"console\_show": "1", #是否在控制台打印日志

"btgw\_port": "18191", #交易网关端口号

"verify\_interval": 360000, #验证token的时间间隔

"sendlog": "0" #是否发送交易日志到后台监控服务器

}

#### 2.2.3 配置代码映射表symbols\_map.csv

代码映射表用于把上层调用中标准的代码名称转换成各交易所指定的格式，同时支持反向的转换，大小写敏感。代码映射表文件配置格式如下，第1列为交易所名称，第2列为交易所的品种代码，第3列为标准品种代码：

......

BINANCE,ETHBTC,eth\_btc

BINANCE,LTCBTC,ltc\_btc

BINANCE,BNBBTC,bnb\_btc

BINANCE,NEOBTC,neo\_btc

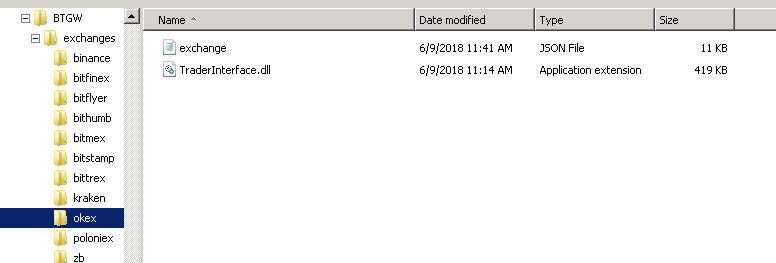
BINANCE,QTUMETH,qtum\_eth

......

上层客户端调用时请求参数里面填入标准品种代码，程序按照symbols\_map.csv的对应关系获取到交易所的品种代码，发送给交易所；与此类似，交易所返回时，再将交易所品种代码转换为标准品种代码，返回给三层调用客户端。

#### 2.2.4 配置exchange.json

exchange.json是交易所动态库配置文件，位于exchanges子目录下交易所同名目录下，如okex的配置文件位于.\exchanges\okex目录下。



该配置文件定义了标准请求数据包头、交易所请求相关信息和4种主要的请求类型应答数据格式，由交易所动态库TraderInterface.dll加载解析。4种主要请求类型是下单、撤单、单条查询和多条查询，分别是add\_order、cancel\_order、order\_info、order\_info\_multi。每种业务包括4种格式的配置：标准请求格式standard\_req、交易所请求格式exchange\_req、交易所应答格式exchange\_resp和标准应答格式standard\_resp。交易所动态库识别配置好的格式及其转换关系，实现从标准请求 -> 交易所请求 -> 交易所应答 -> 标准应答整个消息流程的自动转换和发送。其中变量以@开头的为标准变量，如@symbol\_name是面向上层策略客户端的品种代码，由策略端传进来，或者送往策略端；以$开头的为交易所变量，如$symbol是送往实际交易所的，或者从交易所返回的。

参数文件配置格式示例如下：

"exchange\_req": {

"$symbol": "pair",

"$buy\_sell": "type",

"$order\_type": "ordertype",

"$price": "price",

"$amount": "volume"

},

"exchange\_resp": {

"$result": "error",

"$order\_id": "result:txid[]",

"$error\_message": "error"}

从标准请求数据转换到交易所请求数据的转换，是由exchange\_req、value\_maping\_request和其他相关参数完成。exchange\_req 设定交易所变量的字段名称，即key，value\_maping\_request 设定交易所变量的字段值，即value。value\_maping\_request中可以设置多个输入变量合并起来确定一个交易所变量，如$buy\_sell是由@buy\_sell和@order\_type决定的，也可以一对一的确定一个交易所变量，如"$price": "@price"，还可以配置特殊对应关系，如$symbol，它是由symbols\_map.csv对照表决定的。

"exchange\_req": {

"$symbol": "pair",

"$buy\_sell": "type",

"$order\_type": "ordertype",

"$price": "price",

"$amount": "volume"

},

"value\_maping\_request": {

"$buy\_sell": {

"input\_fields": {

"@buy\_sell": "",

"@order\_type": ""

},

"value": {

"0,0": "buy",

"0,1": "buy\_market",

"1,0": "sell",

"1,1": "sell\_market",

"\*": ""

}

},

"$symbol": {

"input\_fields": {

"@symbol\_name": ""

},

"value": {

"&symbol\_map": "@symbol\_name"

}

},

"$price": "@price",

"$amount": "@amount"

},

### 2.3 交易测试

客户端测试发单的模拟程序是TEmu.exe,连接到网关程序BTGW.EXE进行请求应答的处理。

启动TEmu.exe,可以后跟参数为网关的IP地址和端口号。比如192.168.0.123 1819。 如果不带参数启动，则默IP地址和端口号分别为： 127.0.0.1 1819。

发单指令可以在控制台下输入或者粘贴交易指令，指令及参照emu\_msg.txt里面的数据样本进行发送（可以根据实际情况自行修改），发送完网关处理完可以把执行结果显示到该客户端，以便用户检查跟踪。

交易发送的请求和应答消息格式，参见文件《数字货币网关消息定义》。