HOW TO TRADE ON AN EXTERNAL CRYPTOCURRENCY EXCHANGE VIA METATRADER 5

psonk (25) (/@psonk) in **steemit** (/trending/steemit) • 2 months ago

Not so long ago, the MQL5 language developers have introduced the updated functionality featuring the ability to develop custom symbols and charts. The broad traders' community has not yet had time to appreciate the power of this innovation, but even an easy and unobtrusive brainstorm shows an enormous potential hidden in the custom symbols. Together with other MQL tools, they allow you to implement many of the most daring and interesting ideas.
From now on, MetaTrader 5 is not just a terminal that interacts with one DC. Instead, it is a self-sufficient analytical platform able to connect to various exchanges via the API, as well as visualize price movements and trade flows. A small set of new features turns the terminal into an open system rather than a toolbox containing a limited number of trading tools. In my opinion, custom tools can also become powerful analytical capabilities.

Let's illustrate the new language features using the popular subject of cryptocurrencies as an example. I believe, it will further strengthen the community's interest in custom symbols.
Who might benefit from the article:

cryptocurrency exchange traders;
investors familiar with MetaTrader 5 and portfolio investments;
freelance programmers who can now execute the customers' orders related to cryptocurrency trading in a simpler (and cheaper) way;
everyone who follows the new MetaTrader 5 and MQL5 language features.
First, we need to choose a cryptocurrency exchange providing a web API.

When developing one of my CodeBase products, I used BTC-e, which is no longer relevant. Therefore, I decided to switch to an alternative — . Its API features are sufficient to decently demonstrate both the new and already existing MQL functionality, including downloading bars, price flow, market depth, viewing current account orders and positions as well as order and trade history.

Let's stick to the following plan.

Describing all data structures returned by web requests.
Developing the classes for connecting to the exchange. These classes should implement WebRequest to their access points.
Developing an Expert Advisor that receives bars and updates cryptocurrency prices in MetaTrader 5.
Developing scripts and an EA working with orders and positions.

1. API structures of the exchange data

Like some other cryptocurrency exchanges, BITFINEX has two access points.

The first one (which is a set and format of requests) gives out the price data of the exchange — bars, ticks, market depth. The data are generalized and anonymous, so they are requested without any signatures and authorizations.
The second access point provides data on your account — account status, open positions, pending orders, trades history. These data are private, so you should send the SHA384-HMAC data signature in the request header

using the API key received in your personal trader's room of the exchange. This guarantees that the request is made by you and not by an attacker under your name.
Both access points work in REST style (note: WebSockets are not considered here) using JSON format. This is convenient both for reading data and writing API.

The exchange has two protocol versions. The second version is currently in beta mode. There are two reasons for its appearance. First, there is a new functionality (requesting bars, working with alerts, orders' magic numbers, etc.). The second reason is also clearly visible in the format of JSON structures - this is traffic saving. Let's compare how the response to the Ticker request (current symbol data) looks like in the first and second protocol versions:

Version 1
Version 2
{

"mid":"244.755",
"bid":"244.75",
"ask":"244.76",
"last_price":"244.82",
"low":"244.2",
"high":"248.19",
"volume":"7842.11542563",
"timestamp":"1444253422.348340958"

}
[
BID,
BID_SIZE,
ASK,
ASK_SIZE,
DAILY_CHANGE,

DAILY_CHANGE_PERC,
LAST_PRICE,
VOLUME,
HIGH,
LOW
]
As you can see, the second version avoids naming the fields, but their position is rigidly fixed meaning that arrays are used here instead of objects. In this article, I will show examples for both API versions of the exchange.

First, let's see what requests and structures of the open access point we should use for each protocol version.
For version 1:

Symbols
Symbol Details
OrderBook

For version 2:
Ticker
Candles
Trades
Let me first explain how these structures are made using the SymbolDetails request as an example. Other data structures are presented in the BFxDefine.mqh file. If you need other API requests not included in the article, you can check them out in a similar way.

How it looks in the documentation
How it looks in BFxDefine.mqh
[
{
"pair":"btcusd",
"price_precision":5,
"initial_margin":"30.0",

"minimum_margin":"15.0",
"maximum_order_size":"2000.0",
"minimum_order_size":"0.01",
"expiration":"NA"
},

...

]
struct bfxSymbolDetails
{
string pair;
int price_precision;
double initial_margin;
double minimum_margin;
double maximum_order_size;
double minimum_order_size;
string expiration;
};
struct bfxSymbolsDetails
{
bfxSymbolDetails symbols[];
};
The documentation shows that the request returns the object array. So, I have created two structures. The first one (bfxSymbolDetails) parses and stores data on a single array object. The second one (bfxSymbolsDetails) stores the received bfxSymbolDetails object array and is directly used in a web request. In other words, the JSON <-> MQL matching format is simple, mirror-like and extends to all documentation.

When working with the exchange API, we will use two classes having the common CBFx parent. Its objective is to encapsulate common data fields and the general web request function.

```
//------------------------------------------------------------------ class CBFx
class CBFx
{
protected:
string m_answer; // result of request (before JSON deserialization)
enBFxRequestResult m_lastrr; // code of requests result
CJAVal m_lastjs; // deserialized answer from request
public:
CBFx() { }

protected:
string URL() { return "https://api.bitfinex.com/"; }
string Answer() { return m_answer; }
enBFxRequestResult Request(string mode,string url_request,string
head,string body,char &result[])
{
char data[]; int n=StringLen(body); if(n>0)
StringToCharArray(body,data,0,n); string res_header="";
int r=WebRequest(mode, url_request, head, 10000, data, result,
res_header);
if(r<=0) return rrErrorWebRequest;
return r==200?rrOK:rrBadWebRequest;
}
};
```

The CBFx class has two descendants:

CBFxPublic — requests to public API (bars, quotes, depth of market);
CBFxTrade — requests to private API (trading, account data).
These descendants are described in the next two sections.

All preparatory work with the exchange API has been done. Now, let's implement it in the MetaTrader 5 platform.

1. CustomSymbol

   What is the situation with custom symbols at the moment? Working

with them is almost completely similar to working with ordinary symbols even considering that the entire management is conducted by a small set of functions.

The purpose of our work is to create an exchange symbol, upload the bars history and send prices regularly. Thus, we need the following MQL functions:

CustomSymbolCreate
CustomSymbolSetInteger / CustomSymbolSetDouble / CustomSymbolSetString
CustomRatesUpdate
CustomTicksAdd
Unfortunately, WebRequest cannot be called from the indicator. Therefore, let's develop an EA to regularly request symbol bars and current prices, and then update them in MetaTrader 5.

The interaction between the exchange and MetaTrader 5 is as follows:
OnInit

Check the presence of a symbol in MetaTrader 5 SymbolSelect.
If no symbol found -> Request SymbolDetails of the specified symbol.
Create the CustomSymbolCreate symbol and fill in its properties CustomSymbolSetXXX.
Check M1 history.
If download is required -> Request CandleHist with the necessary number of last bars
Add CustomRatesUpdate bars
Start the timer
OnTimer

Request the Ticker price -> Update the CustomTicksAdd tick
Request the last bar CandleLast -> Update CustomRatesUpdate
This is the entire work cycle.
Let's complicate the work to get more working examples. To do this, make a

request and display the market depth (OrderBook) and time and sales (TradesHist). Time and sales is displayed as a comment on a chart, while the market depth is displayed as segments at the corresponding price levels.

The public API is implemented in the CBFxPublic class.

```
#include (/trending/include) "BFxDefine.mqh"
//------------------------------------------------------------------ class CBFxPublic
class CBFxPublic: public CBFx
{
public:
CBFxPublic() { }
protected:
virtual enBFxRequestResult Request(string url_request);
public:
enBFxRequestResult Symbols(bfxSymbols &ret);
enBFxRequestResult SymbolsDetails(bfxSymbolsDetails &ret);
enBFxRequestResult OrderBook(string pair,int depth,bool
group,bfxOrderBook &ret);

public:
enBFxRequestResult CandleHist(string pair,enbfcTimeFrame tf,int
limit,bfxCandles &ret);
enBFxRequestResult CandleLast(string pair,enbfcTimeFrame tf,MqlRates
&ret);
enBFxRequestResult TradesHist(string pair,int limit,bfxTrades &ret);
enBFxRequestResult Ticker(string pair,bfxTicker2 &ret);
};
```

Let's have a look at CandleHist to explain its work.

Documentation description:
General request form:
https://api.bitfinex.com/v2/candles/trade::TimeFrame::Symbol/Section

Request response:

[
[MTS, OPEN, CLOSE, HIGH, LOW, VOLUME],
...
]
Find the request parameters here:
https://docs.bitfinex.com/v2/reference#rest-public-candles

Sample request:
https://api.bitfinex.com/v2/candles/trade:1m:tBTCUSD/hist?limit=50

CandleHist MQL implementation:

```
//---------------------------------------------------------- CandleHist
enBFxRequestResult CBFxPublic::CandleHist(string pair,enbfcTimeFrame
tf,int limit,bfxCandles &ret)
{
Request(URL()+"v2/candles/trade:"+bfcTimeFrame[tf]+":t"+STU(pair)+"/hist
"+(limit>0?"?limit="+string(limit):""));
if(m_lastrr==rrOK) m_lastrr=ret.FromJson(m_lastjs)?
rrSuccess:rrErrorStruct;
return m_lastrr;
}
```

CandleHist function forms the GET request. Since we are interested in minute bars, the request line eventually looks as follows: https://api.bitfinex.com/v2/candles/trade:1m:tXXXXXX/hist. As a response, we get 1000 last bars from history.

The result of all these manipulations with API requests is a simple EA that is to construct bars and move the price along the chart.

```
#include (/trending/include) "..\BFxAPI\BFxPublicAPI.mqh"

input string Pair="btcusd";
input int ShowBookDepth=0;
input bool ShowTimeSales=false;
```

```
CBFxPublic bfx;
string mtPair;
//------------------------------------------------------------------ OnInit
int OnInit()
{
// create name for MT symbol
#ifdef (/trending/ifdef) MQL4
mtPair=StringToUpper(Pair);
#endif (/trending/endif)
#ifdef (/trending/ifdef) MQL5
mtPair=Pair; StringToUpper(mtPair);
#endif (/trending/endif)
mtPair+=".bfx";

// select symbol in MarketWatch
bool bnew=false;
if(!SymbolSelect(mtPair,true))
{
bfxSymbolsDetails sd;
enBFxRequestResult brr=bfx.SymbolsDetails(sd);
if(brr!=rrSuccess) return INIT_FAILED;
for(int i=0; i<ArraySize(sd.symbols);++i)
{
bfxSymbolDetails ss=sd.symbols[i];
if(ss.pair==Pair)
{
bnew=true;
CustomSymbolCreate(mtPair,"BITFINEX");
CustomSymbolSetString(mtPair,SYMBOL_ISIN,Pair);
CustomSymbolSetInteger(mtPair,SYMBOL_DIGITS,ss.price_precision);
CustomSymbolSetDouble(mtPair,SYMBOL_MARGIN_INITIAL,ss.initial_ma
rgin);
CustomSymbolSetDouble(mtPair, SYMBOL_VOLUME_MAX,
ss.maximum_order_size);
CustomSymbolSetDouble(mtPair, SYMBOL_VOLUME_MIN,
```

```
ss.minimum_order_size);
CustomSymbolSetDouble(mtPair,SYMBOL_VOLUME_STEP,
ss.minimum_order_size);
}
}
if(!SymbolSelect(mtPair, true)) return INIT_FAILED;
}
if(Symbol()!=mtPair) ChartSetSymbolPeriod(0,mtPair,bnew?
PERIOD_M1:Period());

// load some history
datetime adt[]; ArraySetAsSeries(adt,true);
int limit=1000;
if(CopyTime(mtPair,PERIOD_M1,0,1,adt)==1)
limit=(int)fmax(2,fmin((TimeCurrent()-adt[0])/60,1000));

bfxCandles bars;
enBFxRequestResult brr=bfx.CandleHist(Pair,tf1m,limit,bars);
if(brr==rrSuccess) CustomRatesUpdate(mtPair,bars.rates);

// start timer
EventSetTimer(3);
return INIT_SUCCEEDED;
}
//------------------------------------------------------------------ OnDeinit
void OnDeinit(const int reason) { EventKillTimer();
ObjectsDeleteAll(0,Pair,0); if(ShowTimeSales) Comment(""); }
//------------------------------------------------------------------ OnTimer
void OnTimer()
{
// get last tick
bfxTicker2 bt;
MqlTick tick[1]={0};
tick[0].time=TimeCurrent();
tick[0].time_msc=TimeCurrent();
```

```
if(bfx.Ticker(Pair,bt)==rrSuccess)
{
tick[0].flags=TICK_FLAG_BID|TICK_FLAG_ASK|TICK_FLAG_VOLUME|TICK
_FLAG_LAST;
tick[0].bid=bt.bid; tick[0].ask=bt.ask; tick[0].last=bt.last; tick[0].volume=
(long)bt.day_vol;
CustomTicksAdd(mtPair,tick); ChartRedraw();
}

// get last bar
MqlRates rate[1];
if(bfx.CandleLast(Pair,tf1m,rate[0])==rrSuccess)
{
rate[0].spread=int((tick[0].ask-tick[0].bid)*MathPow(10,_Digits));
CustomRatesUpdate(mtPair,rate);
if(tick[0].flags>0) { tick[0].last=rate[0].close; CustomTicksAdd(mtPair,tick); }
ChartRedraw();
}

if(ShowBookDepth>0) ShowBook(ShowBookDepth);
if(ShowTimeSales) TimeSales();
}
//------------------------------------------------------------------ ShowBook
void ShowBook(int depth)
{
bfxOrderBook bk;
if(bfx.OrderBook(Pair, depth, true, bk)!=rrSuccess) return;
if(ArraySize(bk.asks)>0)
{
for(int i=0; i<ArraySize(bk.asks);++i)
SetLine(Pair+"Asks"+IntegerToString(i),TimeCurrent(),bk.asks[i].price,Time
Current()+20*PeriodSeconds(),
bk.asks[i].price,clrDodgerBlue,1,STYLE_DOT,i==0?
DoubleToString(bk.asks[i].volume,1):"");
}
```

```
if(ArraySize(bk.bids)>0)
{
for(int i=0; i<ArraySize(bk.bids);++i)
SetLine(Pair+"Bids"+IntegerToString(i),TimeCurrent(),bk.bids[i].price,Time
Current()+20PeriodSeconds(),
bk.bids[i].price,clrRed,1,STYLE_DOT,i==0?
DoubleToString(bk.bids[i].volume,1):"");
}
}
//------------------------------------------------------------------ TimeSales
void TimeSales()
{
bfxTrades tr;
string inf="";
if(bfx.TradesHist(Pair,10,tr)==rrSuccess)
{
for(int i=0; i<ArraySize(tr.trades);++i)
{
bfxTrade t=tr.trades[i];
inf+="\n "+IntegerToString(t.id)+" |
"+TimeToString(t.mts,TIME_DATE|TIME_MINUTES|TIME_SECONDS)+" |
"+DoubleToString(t.price,_Digits)+
" | "+DoubleToString(fabs(t.amount),2)+" | "+(t.amount>0?"Buy ":"Sell");
}
}
Comment(inf);
}
//------------------------------------------------------------------ SetLine
void SetLine(string name,datetime dt1,double pr1,datetime dt2,double
pr2,color clr,int width,int style,string st)
{
ObjectCreate(0,name,OBJ_TREND,0,0,0);
ObjectSetInteger(0,name,OBJPROP_RAY,false);
ObjectSetInteger(0,name,OBJPROP_TIME,0,dt1);
```

```
ObjectSetDouble(0,name,OBJPROP_PRICE,0,pr1);
ObjectSetInteger(0,name,OBJPROP_TIME,1,dt2);
ObjectSetDouble(0,name,OBJPROP_PRICE,1,pr2);
ObjectSetInteger(0,name,OBJPROP_WIDTH,width);
ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
ObjectSetString(0,name,OBJPROP_TEXT,st);
ObjectSetInteger(0,name,OBJPROP_STYLE,style);
}
//------------------------------------------------------------------ SetArrow
void SetArrow(string name,datetime dt,double pr,color clr,int width,string
st)
{
ObjectCreate(0,name,OBJ_ARROW_LEFT_PRICE,0,dt,pr);
ObjectSetInteger(0,name,OBJPROP_TIME,0,dt);
ObjectSetDouble(0,name,OBJPROP_PRICE,0,pr);
ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
ObjectSetString(0,name,OBJPROP_TEXT,st);
ObjectSetInteger(0,name,OBJPROP_WIDTH,width);
}
//+------------------------------------------------------------------+
```

First, be sure to launch it in debug mode and move along all the calls step by step. Check out what exactly is returned by the API, how responses are parsed and data are displayed

steemit (/trending/steemit) psonk (/trending/psonk)
forestraders (/trending/forestraders) all (/trending/all)
cryptocurrencytraders (/trending/cryptocurrencytraders)

Sort Order: **Trending**

cheetah (73) **(/@cheetah)** · 2 months ago (/steemit/@psonk/how-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5#@cheetah/cheetah-re-psonkhow-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5) [-]

Hi! I am a robot. I just upvoted you! I found similar content that readers might be interested in:
https://www.mql5.com/en/articles/4160

$0.00 Reply

psonk (25) **(/@psonk)** · 2 months ago (/steemit/@psonk/how-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5#@psonk/re-cheetah-cheetah-re-psonkhow-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5-20180208t231021790z) [-]

Ok. It means we can always share idea

$0.00 Reply

lalu24 (44) **(/@lalu24)** · 2 months ago (/steemit/@psonk/how-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5#@lalu24/re-psonk-how-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5-20180209t004019798z) [-]

hey bro don't copy the content
cheetah is a bot which recognize the copied content
now as a beginner you wont realize it but when you got a high reputation it will frustrate you
there is another bot too so called steem cleaner and there are many too

$0.00 Reply

orelmely (60) **(/@orelmely)** · last month (/steemit/@psonk/how-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5#@orelmely/re-psonk-how-to-trade-on-an-external-cryptocurrency-exchange-via-metatrader-5-20180324t115733100z) [-]

nice..
thankss it is your code?

$0.00 Reply