FMZ docs Documentation

Release 0.1

xiaocao

Apr 17, 2019

1. Basic Instruction

The main documentation for the site is organized into a couple sections:

CHAPTER 1

1.1 Getting Started

FMZ is an automated trading platform for cryptocurrency traders with support for many bitcoin/eth/altcoin exchange markets. We has a complete tutorial for beginners, such as https://www.fmz.com/bbs-topic/3649 . Only need studying a couple of days, you are ready to code for your own robot.

What can FMZ do for me?

You can learn how to write your bots(strategies) from our strategies' square which contains lots of open source code, share your strategy's code with others, ask for professional help any time, run your strategy on any exchanges, contorl your bot on website with computer or cellphone, sell your strategies if you want, communicate with many other auto-trading lovers in our group. In a word, FMZ is a perfect platform for those who want do automated trading.

Which Cryptocurrency exchanges does FMZ support?

FMZ supports almost all exchanges that are popular, such as Binance, Bitfinex, Bitstamp, OKEX, Huobi, Poloniex,etc. you can also trade futures on OKEX and BitMEX. Check a full support list on *2.2.1 Add Exchange*. You only need to write one strategy and run it on all exchanges without any changes.

What kinds of programming languages can I use to write my strategies?

FMZ supports *JavaScript*, *Python*, *C++* (JavaScript and Python are recommended) for coding your strategies. Benefiting from the completed languages supporting(not a custom language only can be used for one platform), you can improve your programming skills as well as learn to write strategies.

What is the docker?

The docker is a program that runs on your own Internet Server, which in charge of the data request, data reception, network link, Log review etc. You can treat it like your strategy's executor. Even if the FMZ server offline (breakdown, etc.), it will have no influence on your robot that is running. The Manager can run on variety of operating system. Such as *Windows*, *Linux*, *Mac OS*, *Android*, *Raspbian*, etc.

CHAPTER 2

1.2 The Backtest System

What is backtest system? What it uses for?

When you completed a quantitative strategy, how do you know that the logic, amount and direction of profit of this strategy goes? Does it work? Apparently, we wouldn't use the real money to test our strategies on the real market. But we could use the historical data to test it, see how it works out on the time, profit, and asset management in the past. It is winning or losing? How to use the Backtest System?

Does the number from the backtest system accurate? The results can be trusted?

FMZ divides the backtest system into real market level and simulation level. The real market level contains the whole completed historical data back testing.

The simulation level contains the k-lines data at regular intervals. Both level are based on the real market historical data. Only the real market level is more accurate and the results are more reliable.

The simulation level Backtest System explanation.

Notice that backtesting is the strategy only preform in the past. Historical data doesn't represent the future. History may replay, it also may lead to the black swan. Please treat the backtest results reasonable and objectively.

CHAPTER 3

1.3 FMZ Instruction Manual

If you are new to trading, you need to understand a few basic concepts:

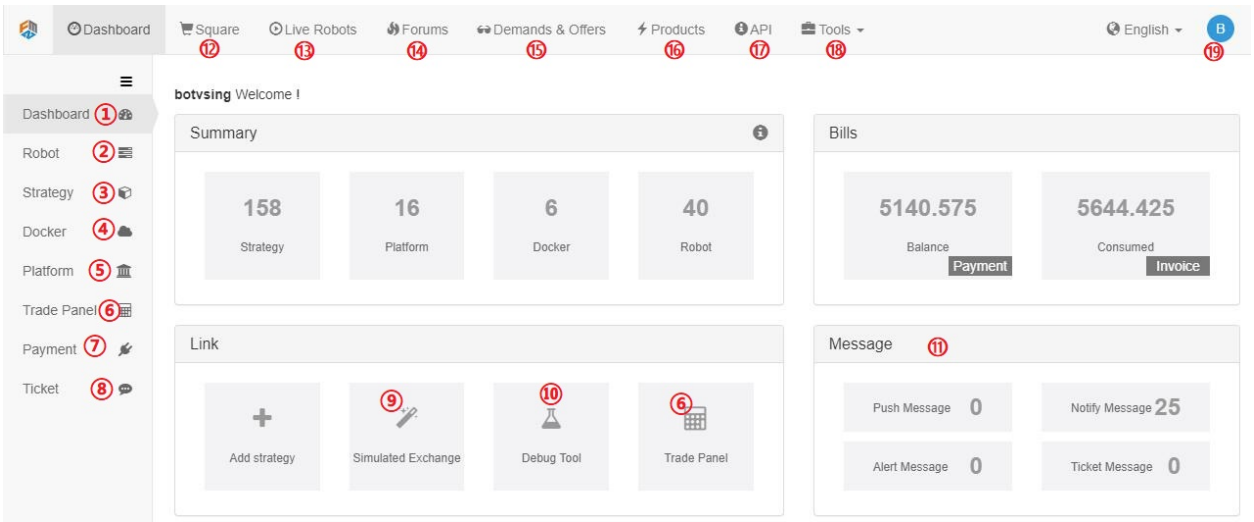
KEY-WORD: “Futures”, “Spot”, “Stock”, “Position”, “Long”, “Short”, “Balance”, “Margin Call”, “Hedge”, “K Line”, “MACD”, “Ask / Bid”.

3.1 1.3.1 A quick look of the main page

After learning the most basic concepts, let’s start using FMZ to explore the quantitative world. (Building your own quantitative trading system is a very large project, you need to have considerable computer knowledge and skills, fortunately, FMZ has done this for you!)

Register your FMZ account,Log in to https://www.fmz.com.

First time to log in the website, it looks like this:



- 1.Your main control page

- 2.Manage all your bots (start,stop,delete,open,etc)
- 3.Manage all your strategies' code
- 4.Deploy and manage your docker
- 5.Add new exchanges
- 6.Manual trading on the exchanges you added
- 7.Pay your bill
- 8.Ask any question here
- 9.FMZ's simulated exchange
- 10.Debug tool where you can run a block of code without start a bot.
- 11.All kinds of message
- 12.Strategy square where open-source and charging strategies are listed
- 13.Live Robots where all live-running bots are listed.
- 14.Forums where you can post a post to discuss any question related.
- 15.Ask for someone to write code for you or provide this service for others.
- 16.Products for exchanges and agencies.
- 17.API documentation.
- 18.Some useful tools, check for yourself.
- 19.Your account information.

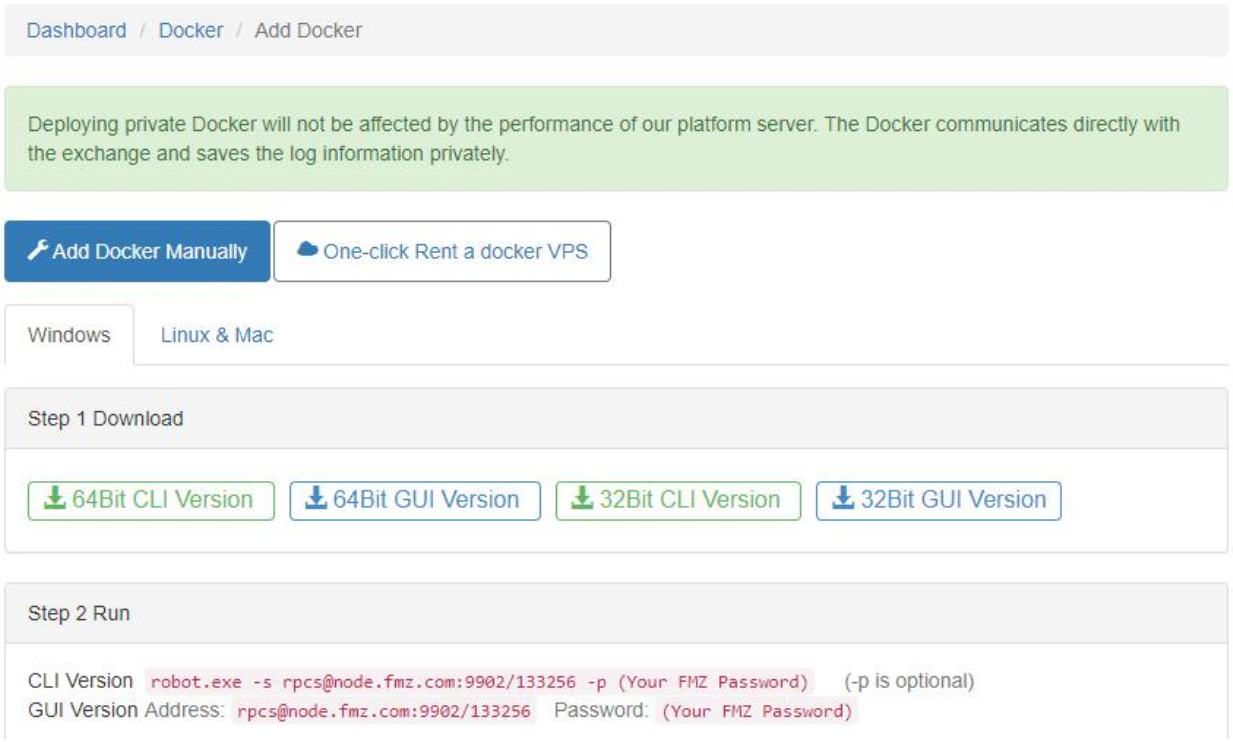
3.2 1.3.2 Deploy the docker

First of all, FMZ's framework is very advanced, the user's robot program (that is, the automated trading program) is running on the user's own computer (of course, it can also be run on the cloud server), So, it's very safe (don't need worry about the FMZ website breakdown etc.), the user has direct control over the program.

Docker is a program that run your robots and communicate with FMZ website. You need to run a Docker first before start a real market robot.

Note: It is highly recommended to use the cloud server for runing program stably, such as Amazon or Google Cloud Server.

In the Dashboard page–`Add docker` button, you can link to the download page https://www.fmz.com/m/add-node.



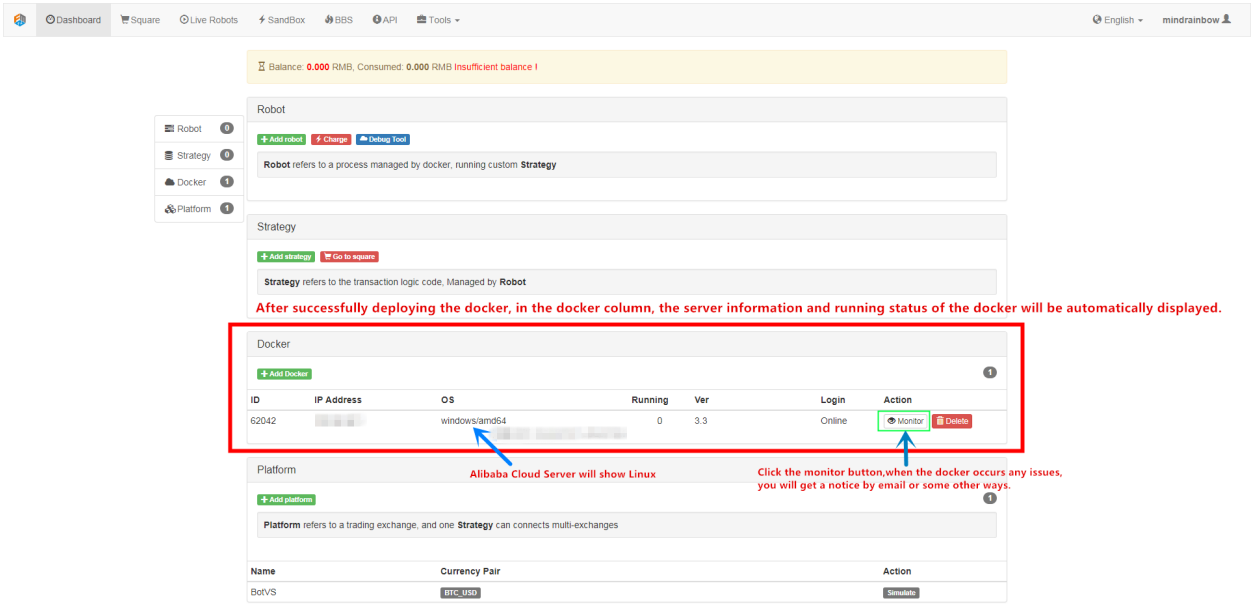
Here are steps to deploy the dockr in a Linux server(centOS 6):

- Buy a cloud server (VPS) from Amazon or Google, the lowest and cheapest configuration is enough. you may often has a free try for a long time.
- Login your server, fellow the instruction from your server provider or Google.
- Chose the docker that statisty your system version, most of the time, it is 64Bit.
- For `centos`, run `wget 'http://q.fmz.net/dist/robot_linux_amd64.tar.gz'`, command not found? install first `yum install wget -y`.
- Run `tar -xzvf robot_linux_amd64.tar.gz` to unzip.
- Run `./robot -s rpcs@node.fmz.com:9902/xxxxxx -p yourFMZpassword`, you should see something like `2018/07/05 05:04:10 Login OK, SID: 62086, PID: 7226, Name: host.localdomain`, which means everything is worked.
- `rpcs@node.fmz.com:9902/xxxxxx` is unique to every users, find your own on https://www.fmz.com/m/add-node.
- Now the docker isn't run in the background, if you close the SHH client, the docker will stop.
- Press `ctrl + C` to stop the docker.
- Run `nohup ./robot -s rpcs@node.fmz.com:9902/xxxxxx -p yourFMZpassword &` to run in the background. this step can also be done by `Screen` command.
- Check on https://www.fmz.com/m/dashboard, if everything is OK , you can find the docker deployed.

Steps to update the docker:

Note: If you want to keep the old docker, one server can run many dockers, just create a new folder and repeate the deploy steps.

- Stop all robots that run on the docker.
- Delete the docker from FMZ website. the docker will stop on your server too(don't have to, you can run two dockers on one server, just create a new folder)
- Run `rm -rf robot_linux_amd64.tar.gz` in your dokcer files to delete the old docker.
- Run `wget http://q.fmz.net/dist/robot_linux_amd64.tar.gz` to download the lastest docker.
- Repeate the steps above.
- Change robot's config to use the new docker, restart robots.



Note: One docker can run many robots, however, you can deploy more than one dockers on different server for speed or request-rate-limit consideration. the docker can be specified or auto-distributed when start a robot.

Warning: There are two public dockers for testing. don't use them to run your robot on real market.

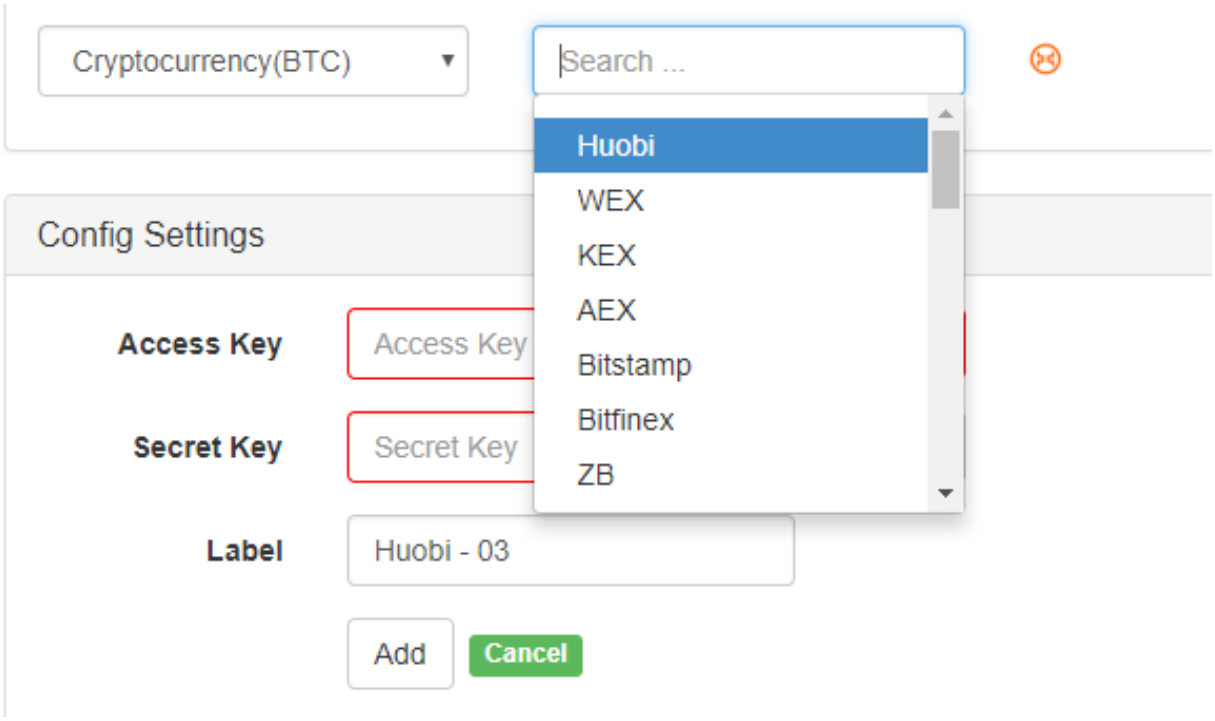
3.3 1.3.3 Add exchanges

Add your exchanges at this page: https://www.fmz.com/m/add-platform.

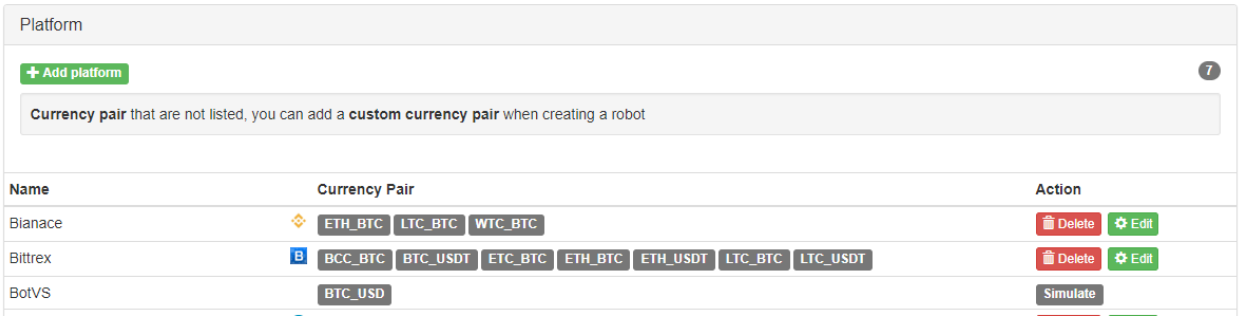
Now support:

```
Binance, Bitfinex, Huobi(huobipro), OKEX, Futures_OKCoin(OKEX), Futures_BitMEX,␣
↪Poloniex, Bitstamp,
Wexapp(FMZ Simulation Exchange),AEX, BigONE, BitFlyer, Bithumb, Bitpie, Bittrex,␣
↪CoinEx, CoinPlus,
Coincheck, Coinone, Futures_CTP, Futures_Deribit, Futures_Esunny, GateIO, HitBTC, KEX,
↪ Korbit,
Kraken, LiveCoin, OKCoin_EN, Quoine, WEX, ZB, Zaif.
```

`Access Key` and `Secret Key` is needed, you should apply on your exchange first.



Once the exchange is added, you can find it on Dashboard https://www.fmz.com/m/dashboard.

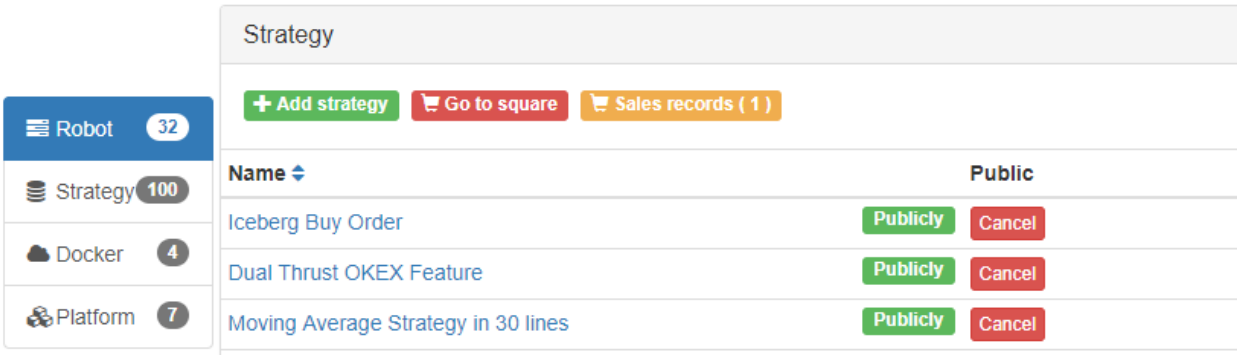


Note: New exchange supported is keep being added. you need to update the lastest docker to support new exchange.

3.4 1.3.4 Write or copy a strategy

Note: There are lots of details this docs doesn't cover, you can explore by yourself, most of them are simple and clear. You can always post on our forum if you have any question.

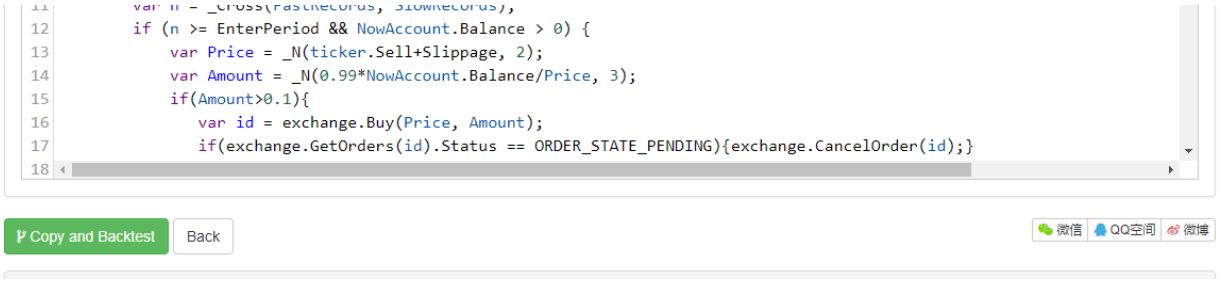
Write your own strategy by clicking `Add Strategy`.



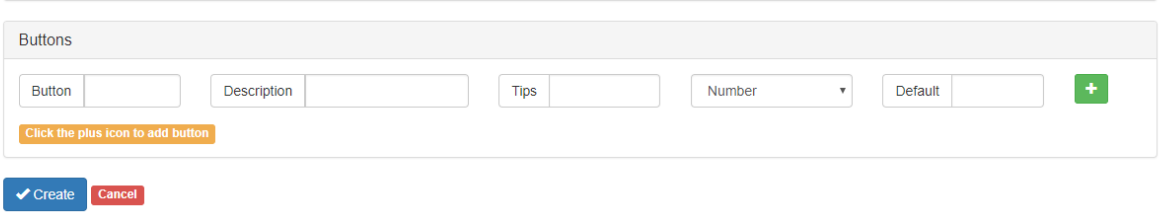
You can choose different code languages and backtesting

For beginners, copy this strategy to begain: https://www.fmz.com/strategy/103070, which can be found on https://www.fmz.com/square.

Click `Copy and backtest`:

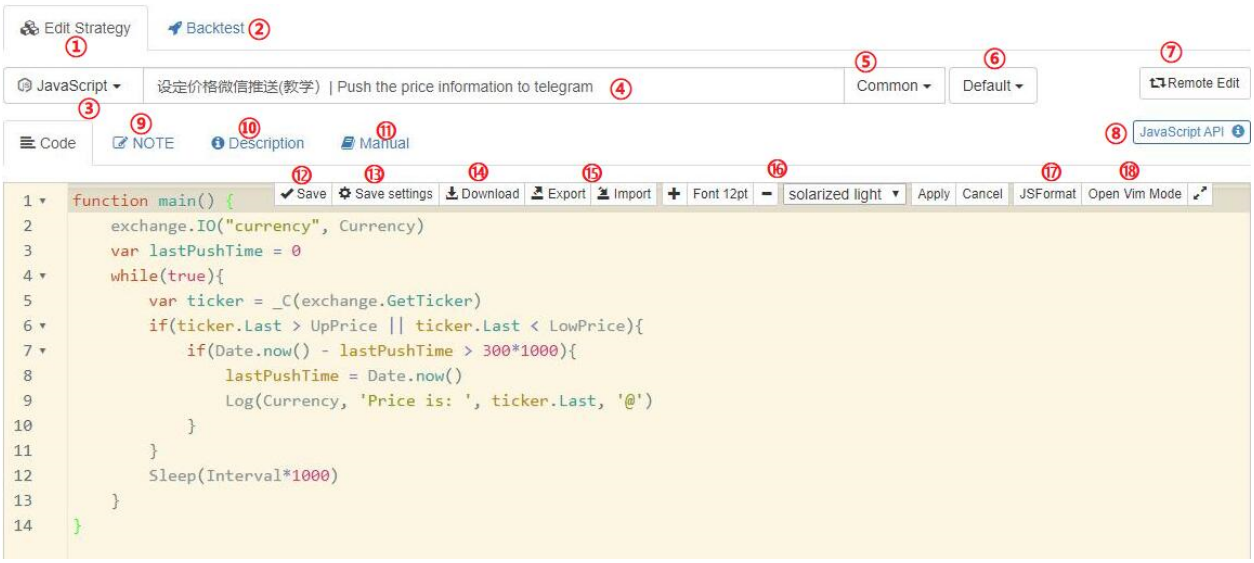


Click `Creat`:



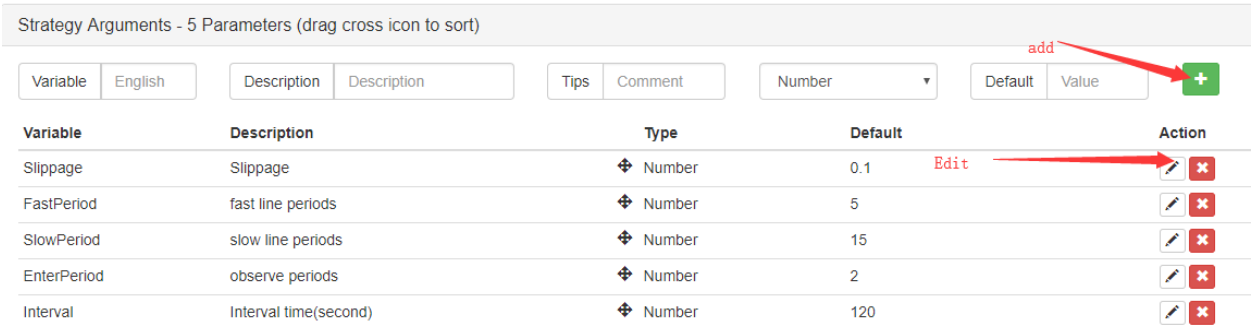
Now your can find this strategy on your dashboard strategies list. https://www.fmz.com/m/dashboard

Edit your code here, don't forget to save your code:



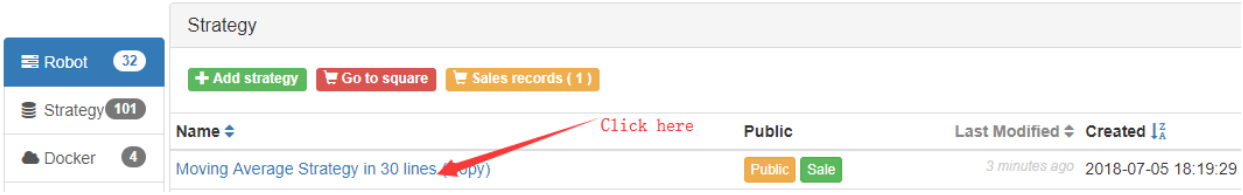
- 1.Edit your code
- 2.Backtesting, we will cover this part on an intermediate tutorial
- 3.The programer language of your code, JavaScript was used in this demo
- 4.The title, "|" splits Chinese and English title, which one will be showed is decided by the language of FMZ website
- 5.The type of your strategy, the default is common
- 6.The category of your strategy. You can divide your strategies into different categories if you have too many
- 7.Remote editing your code from your own IDE instead of our website
- 8.A link to the API doc
- 9.Notes of the strategy(only be seen by yourself). you can record the thoughts here.
- 10.Descriptions of the strategy. Others will see the descriptions if you share or sell your strategy on Square.
- 11.Manual of the strategy, can only be seen when someone bought your strategy.
- 12.Save your code, or `Ctrl+S` on edit mode.
- 13.Save the backtesting config on the code.
- 14.Download the strategy file
- 15.Export and import the strategy while keeping all the parameters
- 16.Change the font size and edit the theme
- 17.Format the code automatically
- 18.Use VIM mode to edit.

Change and add global variables here:

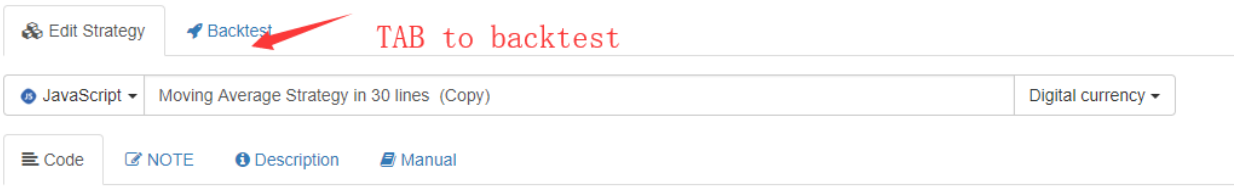


3.5 1.3.5 Backtest your strategy

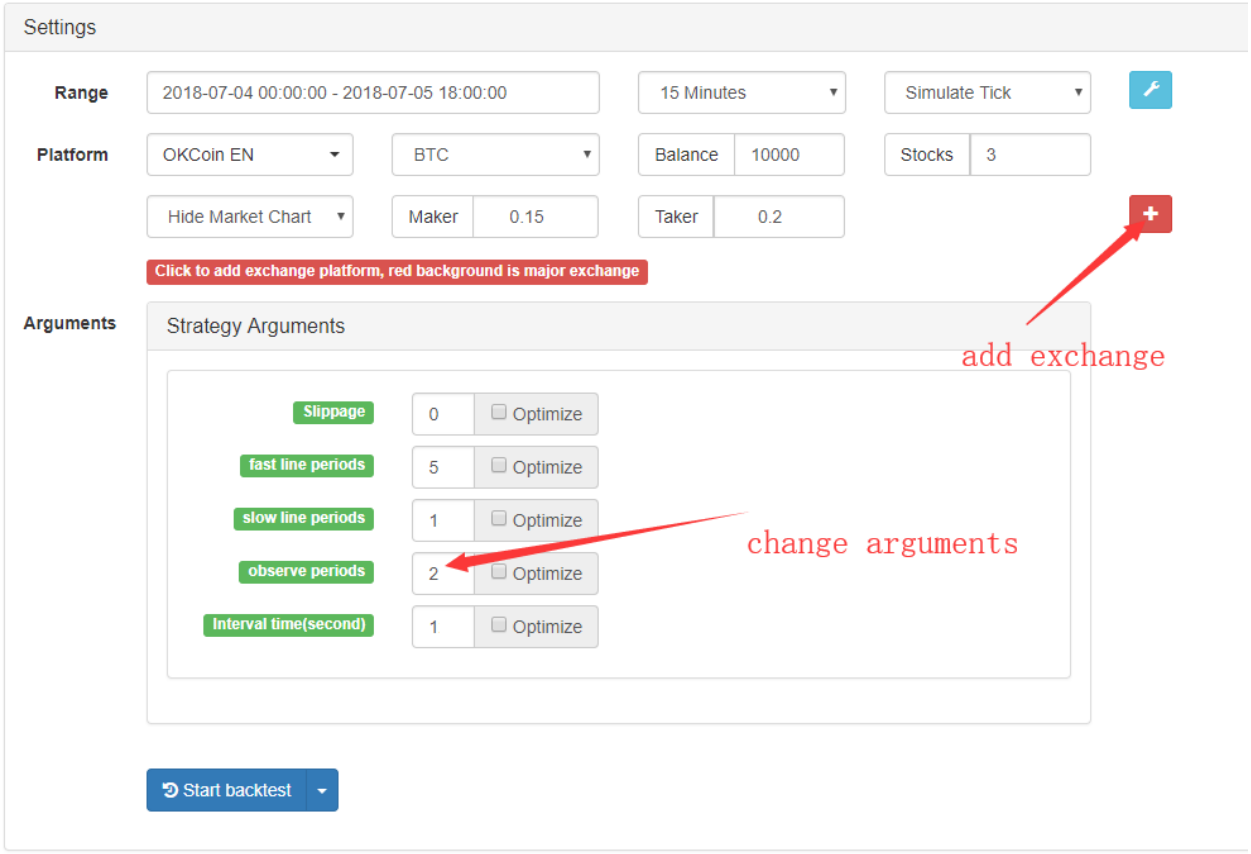
Click Stragegy name to strategy page.



Go to backtest page:



Add exchange and config your strategy:

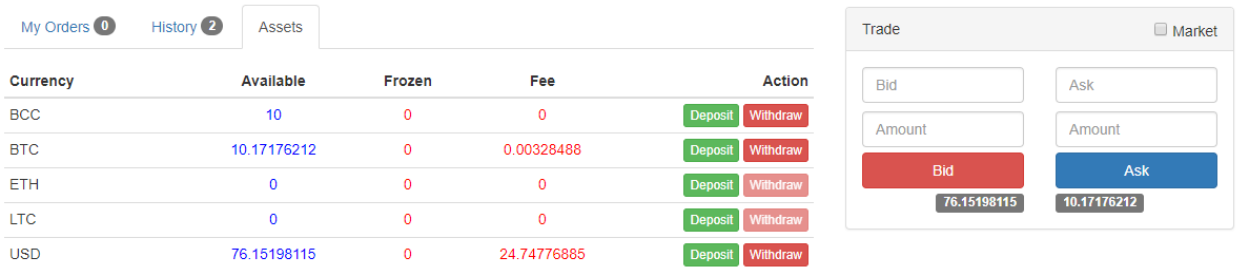


Click `Start Backtest` to start.

3.6 1.3.6 Run a robot on Wexapp

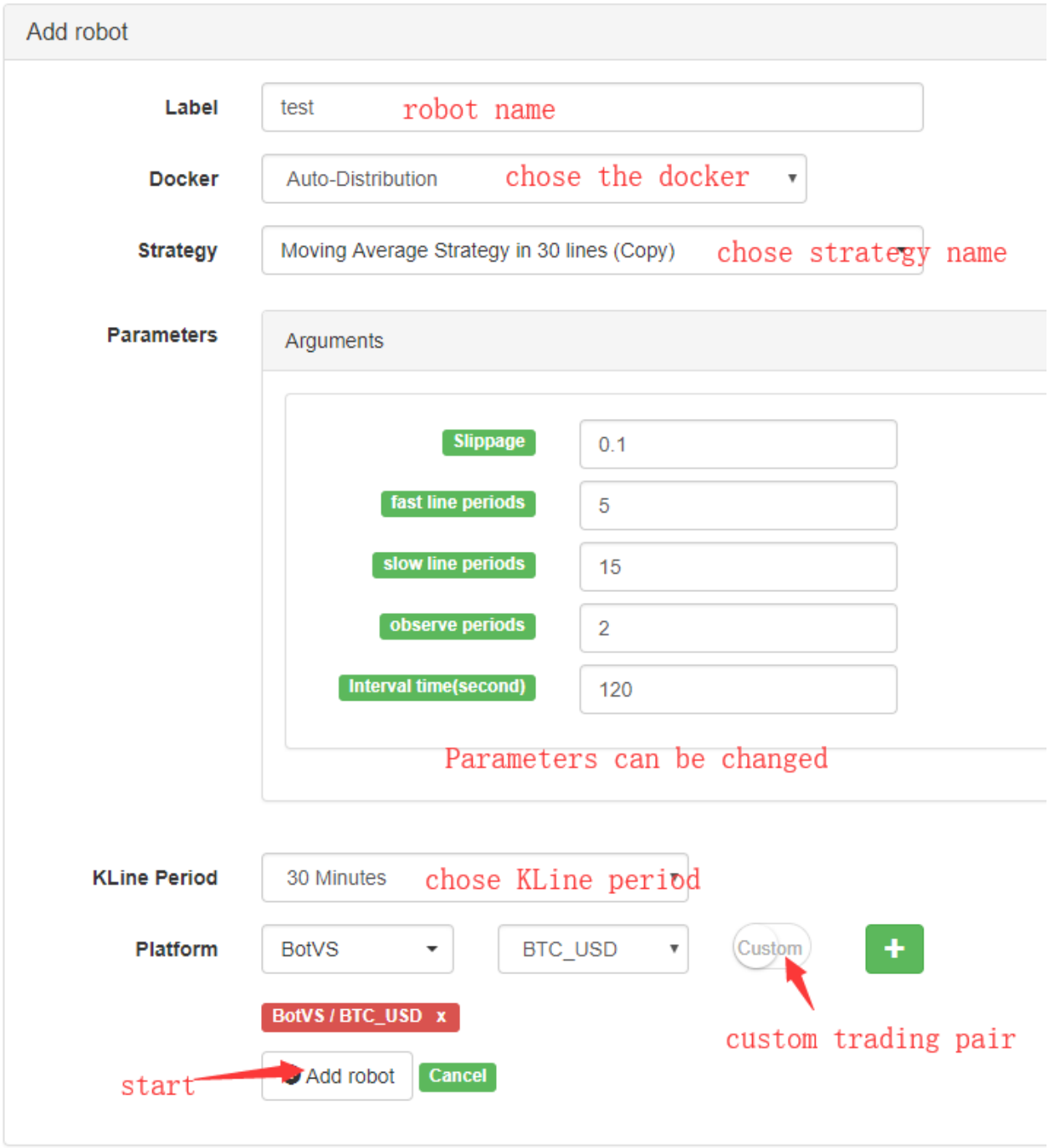
Wexapp is FMZ Simulation Exchange, which is basically the same as a real exchange but free of charge, you can run your robot on FMZ Simulation Exchange for testing your strategy.

First, you need to deposit assets on your simulation account on https://www.fmz.com/m/sandbox.

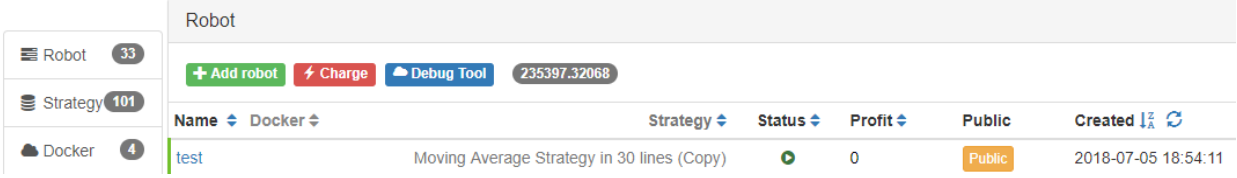


Click `Add Robot` or https://www.fmz.com/m/add-robot to run a robot.

Config page as below:

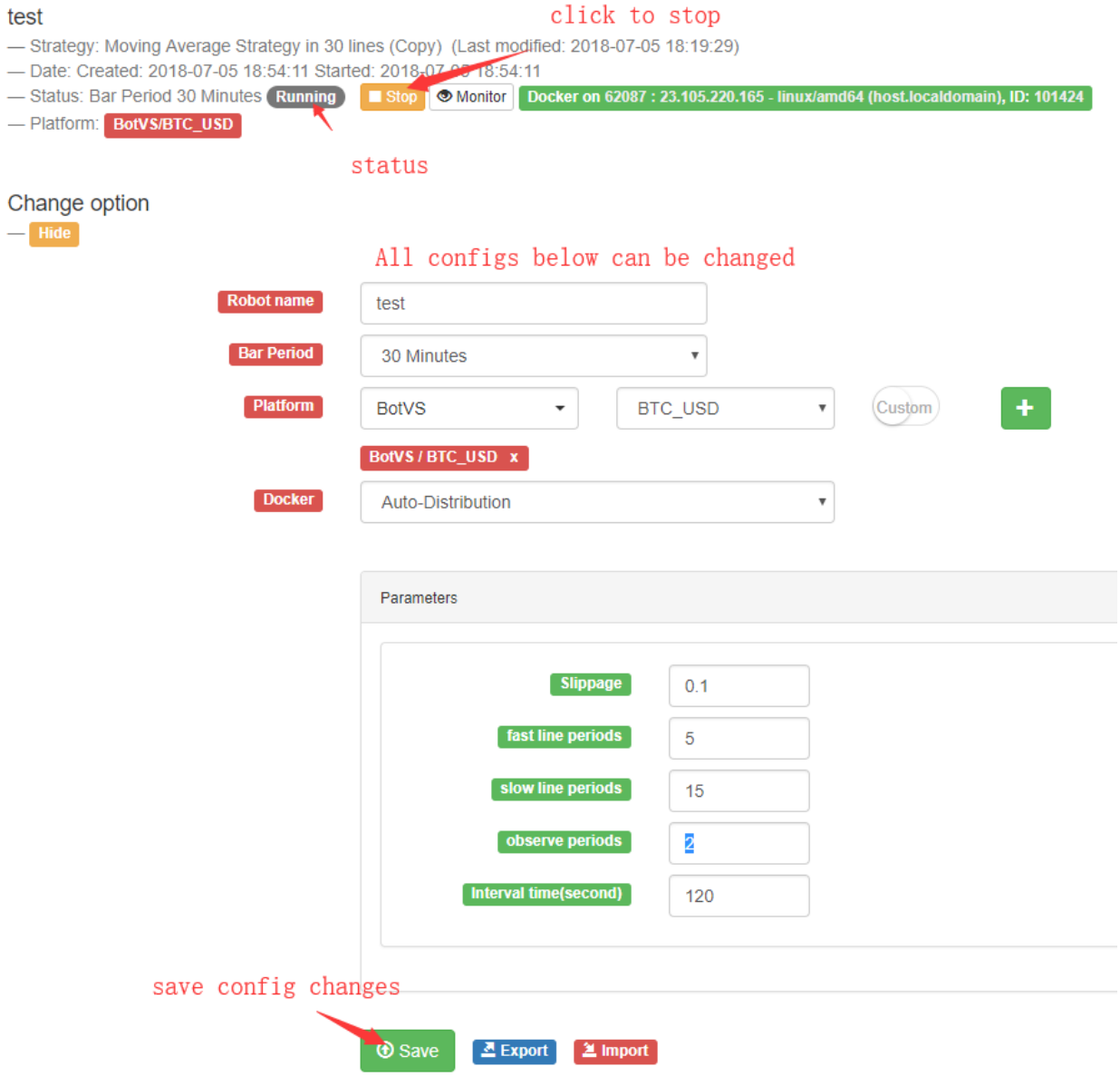


You can find your robot is running on dashboard Now.



Go to robot page:

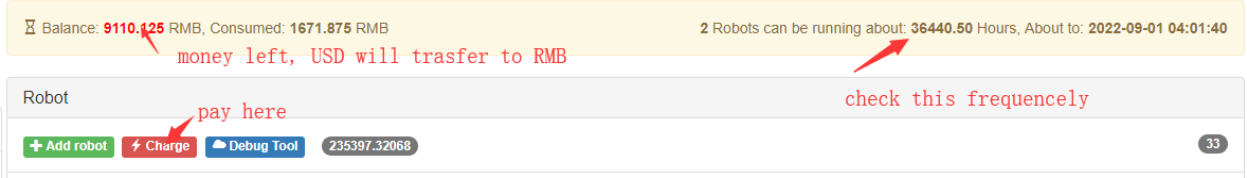
You can check the robot's status and Logs, change the configs(need to stop robot first),



3.7 1.3.7 Charges Notes

0.125 RMB per robot per hour(around 0.018 USD).

robot run on FMZ Simulation Exchange(Wexapp) is free.



CHAPTER 4

2.1 Typical Strategy structure

Strategy Coding currently support *JavaScript*, *Python*, *C++*, more programming tools supports under developing...

- `main()` as the entry function.
- `onexit()` as the normal exit function, 5 minutes top run time. It can undeclared.
- `onerror()` as the abnormal exit function, 5 minutes top run time. It can undeclared.
- `init()` as the Initialization function, the strategy program will run it automatically at the beginning . It can undeclared.

Example of a basic strategy fame:

```
function onTick(){
//write your strategy here, it will repeat itself
}
function main(){
    while(true){
        onTick();
        Sleep(60000);
        }
    }
}
```

For example, if You want buy one amount of order at the price 100 every second. It can be written like this:

```
function onTick(){
    exchange.Buy(100,1);
}
function main(){
    while(true){
        onTick();
        Sleep(1000);//Pause time is optional, Units are Millisecond1 second = 1000ms)
    }
}
```

CHAPTER 5

2.2 Exchange

Every API method is called by exchange object, such as `exchange.GetTicker()`, `exchange.Buy()`.

Exchange and trade pair is set when you start a robot. The following market api and trade api is based on this setting. If you set your exchange is "Binance", trading pair is "BTC_USDT", then call `exchange.GetTicker()`, you will get the ticker of "BTCUSDT" of Binance. trading pair is also called `symbol` or `currency` in some exchenge's API docs,

5.1 2.2.1 Add Exchange

Check on how to add exchange when start a robot: *1.3.3 Add exchanges*

Now support:

```
AEX
BigONE
Binance
BitFlyer
Bitfinex
Bithumb
Bitpie
Bitstamp
Bittrex
BotVS
CoinEx
CoinPlus
Coincheck
Coinone
Exchange
Futures_BitMEX
Futures_CTP
Futures_Deribit
Futures_Esunny
Futures_OKCoin
```

(continues on next page)

(continued from previous page)

```
GateIO
HitBTC
Huobi
KEX
Korbit
Kraken
LiveCoin
OKCoin_EN
OKEX
Poloniex
Quoine
STEX
WEX
ZB
Zaif
```

5.2 2.2.2 Exchange Variable

exchange

It can be imaged as a real exchange; default setting is the first exchange in your strategy's parameter. all of data interactive with the exchange which achieve through this object function.

exchanges

The exchange's array, it contains multiple exchange objects which are sorted by order of addition . used as `exchanges[0]` , `exchange[1]` etc...

Note: If you use `exchange` directly, which equals to `exchanges[0]`.

When you add an exchange, it's always fllowed by a trade pair, such as `BTC_USDT`, so `exchanges[0]` and `exchanges[1]` can be the same "actually exchange" but with different trade pair.

5.3 2.2.3 Exchange Information

GetName

`exchange.GetName()` Return the name of the exchange, string type

GetLabel

`exchange.GetLabel()` Return the exchange's custom label, string type

5.4 2.2.4 FMZ Simulation Exchange

FMZ Simulation Exchange is basically the same as a real exchange, you can run your robot on FMZ Simulation Exchange for testing your strategy, which is totally free. check it on https://wex.app, and deposit some money or bitcoin to start

CHAPTER 6

2.3 Market API

The following functions is used for get market inforamtion of exchange, which are called from object `exchange` or `exchanges[x]`,for example`exchange.GetTicker()` or `exchanges[0].GetTicker()` means return the market quotations.

Warning: When calling any API function that accesses the exchange API (such as `GetTicker()`, `Buy()`, `CancelOrder()`, etc...), it may get access failure due to exchange server problem, the network transmission problem, and so on. In this case, `GetTicker()` will return `null`, which may cause the stop of your programe. a JavaScript example to do fault tolerance as below.

```
var ticker = exchange.GetTicker()
if(ticker == null){
    // Retry, or other processing logic.
}
```

A deflaut retry function is `_C()`

6.1 2.3.1 GetTicker

```
exchange.GetTicker()
```

Get the current market quotations.

Return value: Ticker structure

The Ticker structure contains the following variables:

Field	Type	Description
Info	Object	the original data returned by the exchange
High	Number	Highest price
Low	Number	lowest price
Sell	Number	the latest selling price, also called bidPrice
Buy	Number	the latest buying price, also called askPrice
Last	Number	last traded price
Volume	Number	most recent trading volume
OpenInterest	Number	net position(only for features)

Example ticker from binance:

```
{
    "Info":{
            "highPrice":"6173.01000000",
            "openTime":1530152780326,
            "firstId":53572675,
            "lastId":53735989,
            "count":163315,
            "priceChange":"-266.96000000",
            "weightedAvgPrice":"6035.81831943",
            "bidPrice":"5871.63000000",
            "openPrice":"6139.00000000",
            "closeTime":1530239180326,
            "lowPrice":"5827.00000000",
            "quoteVolume":"197096315.23211791",
            "priceChangePercent":"-4.349",
            "prevClosePrice":"6139.00000000",
            "lastQty":"0.25866000",
            "bidQty":"0.00300000",
            "askPrice":"5872.05000000",
            "symbol":"BTCUSDT",
            "lastPrice":"5872.04000000",
            "askQty":"0.07344000",
            "volume":"32654.44796400"
        },
    "High":6173.01,
    "Low":5827,
    "Sell":5872.05,
    "Buy":5871.63,
    "Last":5872.04,
    "Volume":32654.447964,
    "OpenInterest":0,
    "Time":1530239180443
}
```

A JavaScript example using the variables in the Ticker structure:

```
function main(){
    var ticker = exchange.GetTicker();
    Log("High:", ticker.High, "Low:", ticker.Low, "Sell:", ticker.Sell)
}
```

For Python the code is basically the same:

```
def main():
    ticker = exchange.GetTicker()
    Log("High:", ticker.High, "Low:", ticker.Low, "Sell:", ticker.Sell)
```

Note: If you use the a number in `Info` directly, make sure the data type is float.

- For JavaScript: `var priceChange = praseFloat(ticker.Info.priceChange);`
- For Python: `priceChange = float(ticker.Info["priceChange"])`.

6.2 2.3.2 GetDepth

```
exchange.GetDepth()
```

Get the exchange order book.

Return value: Depth structure

The Depth structure contains the following variables:

Field	Type	Description
Asks	Array	the array of asks,from low to high by price
Bids	Array	the array of bids,from high to low by price
Time	Number	the timestamp of request

The Asks and Bids structure contains the following variables:

Field	Type	Description
Price	Number	the pirce of ask or bid
Amount	Number	the amount of ask or bid

Example depth from binance:

```
{
    "Info":null,
    "Asks":[
        {"Price":5866.38,"Amount":0.068644},
        {"Price":5866.39,"Amount":0.263985},
        {"Price":5866.73,"Amount":0.05},
        {"Price":5866.77,"Amount":0.05},
        {"Price":5867.01,"Amount":0.15},
        {"Price":5875.89,"Amount":0.05},
        ......
        ]
    "Bids":[
        {"Price":5865.13,"Amount":0.001898},
        {"Price":5865,"Amount":0.085575},
        {"Price":5864.15,"Amount":0.013053},
        {"Price":5863.65,"Amount":0.016727},
        {"Price":5863.51,"Amount":0.128906},
        {"Price":5863.15,"Amount":0.2}
        ......
```

(continues on next page)

(continued from previous page)

```
        ],
    "Time":1530241857399
}
```

A useful JavaScript example using depth:

```
function main(){
    var depth = exchange.GetDepth();
    var price = depth.Asks[0].Price;
    var amount = depth.Asks[0].Amount;
    if(amount > 10){
        exchange.Buy(price, 10);
    }
}
```

Note: `GetDepth()` doesn't return real depth at backtesting.

6.3 2.3.3 GetTrades

```
exchange.GetTrades()
```

Get Exchange Trading History.(not your trading history)

Return value: Array of Trade Structure

Note: Some exchanges do not support this method, the number of return data depends on exchanges.

The Trade structure contains the following variables:

Field	Type	Description
Time	Number	Unix timestamp of the trade time
Price	Number	price of the trade
Amount	Number	amount of the trade
Type	Order Type	Order Type Constant

Order Type is global constant, you can take `ORDER_TYPE_BUY` as `0` :

Global constant	Meaning	Value
ORDER_TYPE_BUY	buy order	0
ORDER_TYPE_SELL	sell order	1

Example trades from binance:

```
[
    {"Id":47317269,"Time":1530244709886,"Amount":0.002902,"Price":5884.38,"Type":1},
    {"Id":47317270,"Time":1530244709886,"Amount":0.082102,"Price":5884.78,"Type":1},
    {"Id":47317271,"Time":1530244713111,"Amount":0.122439,"Price":5884,"Type":0},
    .....
```

(continues on next page)

(continued from previous page)

```
    {"Id":47317278,"Time":1530244717131,"Amount":0.000029,"Price":5884,"Type":0},
]
```

A useful JavaScript example using trades:

```
function main(){
    while(true){
        var trades = exchange.GetTrades();
        for(var i=0;i<trades.length;i++){
            if(trades[i].Type == ORDER_TYPE_BUY && trades[i].Amount > 100){
                Log("Big amount buy order","time:", trades[0].Time, "Price:",␣
↪trades[0].Price, "Amount:", trades[0].Amount);
            }
        }
        Sleep(3000)//sleep 3 seconds
    }
}
```

Warning: The trades in simulation backtesting is empty.

6.4 2.3.4 GetRecords

```
exchange.GetRecords(period)
exchange.GetRecords()
```

Get Exchange's history K lines/Candlesticks data.

Parameter `period` : K lines cycle, Optional Parameters, default K line cycle is set when start the robot.

All available values:

```
PERIOD_M1  : 1 minute,
PERIOD_M5  : 5 minutes,
PERIOD_M15 : 15 minutes,
PERIOD_M30 : 30 minutes,
PERIOD_H1  : 1 hour,
PERIOD_D1  : one day.
```

Return value: Record structure array. from old to recent by time.

The Record structure contains the following variables:

Field	Type	Description
Time	Number	Unix timestamp of the kline
Open	Number	open price of the kline
High	Number	highest price of the kline
Low	Number	lowest price of the kline
Close	Number	close price of the kline
Volume	Number	trading volume

Example Records from binance:

```
[
    {"Time":1526616000000,"Open":7995,"High":8067.65,"Low":7986.6,"Close":8027.22,
↪"Volume":9444676.27669432},
    {"Time":1526619600000,"Open":8019.03,"High":8049.99,"Low":7982.78,"Close":8027,
↪"Volume":5354251.80804935},
    {"Time":1526623200000,"Open":8027.01,"High":8036.41,"Low":7955.24,"Close":7955.39,
↪"Volume":6659842.42025361},
    ......
]
```

A useful JavaScript example using Records to get a close array:

```
function main(){
    var close = [];
    var records = exchange.GetRecords(PERIOD_H1);
    for(var i=0;i<records.length;i++){
        close.push(records[i].Close);
    }
}
```

Note:

- The K-lines data will accumulate over time, accumulating up to 2000, then will update one record at one K-line cycle, and delete the earliest one at the same time.
- If the exchange provides a K-line API. In this case, the data is obtained directly from the exchange.
- If the exchange does not provide a K-line API. your robot will using `GetTrades()` function to generate K-line each time the user calls GetRecords.In this case,Records length will be one when first start.

CHAPTER 7

2.4 Trade API

The following functions is used for trading, which are called from `exchange` or `exchanges[x]` object, for example: `exchange.Sell(100, 1)`; Send a buy order to the exchange with the price is 100, and the quantity is 1.

If you want change the trading pair before trade, check on `exchange.IO("currency", symbol)`

7.1 2.4.1 GetAccount

```
exchange.GetAccount()
```

Get exchange account information

Return value: Account structure

The Account structure contains the following variables:

Field	Type	Description
Info	Object	The original data returned by the exchange
Balance	Number	Balance (Pricing currency balance, BTC if your trading pair is `ETH_BTC`)
FrozenBalance	Number	Frozen balance in your pending buy orders
Stocks	Number	The available quantity of trading currency, ETH if your trading pair is `ETH_BTC`
FrozenStocks	Number	Frozen Stocks in your pending sell orders

Example of GetAccount from binance, trading pair is `BTC_USDT`:

```
{
    "Stocks":0.38594816,
    "FrozenStocks":0,
    "Balance":542.858308,
    "FrozenBalance":0
    "Info":{
```

(continues on next page)

(continued from previous page)

```
            "takerCommission":10,
            "canTrade":true,
            "canDeposit":true,
            "updateTime":1530330645991,
            "balances":[
                        {"asset":"BTC","free":"0.38594816","locked":"0.00000000"},
                        {"asset":"LTC","free":"0.00736000","locked":"0.00000000"},
                        {"asset":"ETH","free":"2.44434439","locked":"0.00000000"},
                        {"asset":"BNC","free":"0.00000000","locked":"0.00000000"},
                        {"asset":"ICO","free":"0.00000000","locked":"0.00000000"},
                        ......
                        ]
            "makerCommission":10,
            "buyerCommission":0,
            "sellerCommission":0,
            "canWithdraw":true
        },
}
```

A useful JavaScript example of Log your account value for a certain trading pair:

```
function main(){
    while(true){
        var ticker = exchange.GetTicker();
        var account = exchange.GetAccount();
        var price = ticker.Buy;
        var stocks = account.Stocks + account.FrozenStocks;
        var balance = account.Balance + account.FrozenBalance;
        var value = stocks*price + balance;
        Log('Account value is: ', value);
        Sleep(3000);
    }
}
```

7.2 2.4.2 Buy

```
exchange.Buy(Price, Amount)
```

Send a buy order, return an order ID

Parameter

```
Price  : order price, number type
Amount : order quantity, number type
```

Return

```
order id, number type
```

Tip: If the exchange's order API support market orders, use `exchange.Buy(-1, 0.1)`, if your trading pair is `ETH_BTC`, which means buy 0.1 ETH at market price. Be carefully when using market order.

A useful JavaScript example of Buy for buy certain amount of bitcoin at a certain price:

```
function main(){
    while(true){
        var ticker = exchange.GetTicker();
        var price = ticker.Buy;
        if(price >= 7000){
            exchange.Buy(price, 10);
        }
        Sleep(3000);
    }
}
```

7.3 2.4.3 Sell

```
exchange.Sell(Price, Amount)
```

Send a sell order, return an order ID

Parameter

```
Price  : order price, number type
Amount : order quantity, number type
```

Return

```
order id, number type
```

Tip: If the exchange's order API support market orders, use `exchange.Sell(-1, 0.1)`, if your trading pair is `ETH_BTC`, which means sell 0.1 ETH at market price.

7.4 2.4.4 CancelOrder

```
exchange.CancelOrder(orderId)
```

Cancel an order by order id.

Parameter

```
orderId  : order id, returned by Buy or Sell API.
```

Return value: bool type

`true` means that the cancellation of the order request was successful. `false` means cancellation of the order request failed. (It is only a successful request. Whether the exchange cancels the order, it is best to call `exchange.GetOrders()`.)

A JavaScript example of cancel an order after some time:

```
function main(){
    var id = exchange.Sell(99999, 1);
    Sleep(3000);
    exchange.CancelOrder(id);
}
```

7.5 2.4.5 GetOrder

```
exchange.GetOrder(orderId)
```

Get order details by order id.

Parameter

```
orderId  : order id, returned by Buy or Sell API.
```

Return value: Order structure

The Order structure contains the following variables:

Field	Type	Description
Info	Object	The original data returned by the exchange
Id	Number	Unique ticket identifier
Price	Number	Order price
Amount	Number	Order quantity
DealAmount	Number	The deal amount of this order
AvgPrice	Number	Average transaction price (0 means the exchange do not return this field)
Status	Const	Order Status
Type	Const	Order Type, `ORDER_TYPE_BUY` : Buy Order, `ORDER_TYPE_SELL` : Sell Order

Order Status is global constant:

Global constant	Meaning	value
ORDER_STATE_PENDING	Incomplete	0
ORDER_STATE_CLOSED	Completed	1
ORDER_STATE_CANCELED	Canceled	2

Order Type is global constant:

Global constant	Meaning	value
ORDER_TYPE_BUY	BUY	0
ORDER_TYPE_SELL	SELL	1

Example of GetOrder from binance:

```
{
    "Id":125723661,
    "Amount":0.01,
    "Price":7000,
    "DealAmount":0,
    "AvgPrice":0,
    "Status":0,
    "Type":1,
    "ContractType":"",
    "Info":{
        "side":"SELL",
        "stopPrice":"0.00000000",
        "timeInForce":"GTC",
        "type":"LIMIT",
```

(continues on next page)

(continued from previous page)

```
        "time":1530325939498,
        "orderId":125723661,
        "clientOrderId":"H3R333f47MsFrahQUsa8egU",
        "origQty":"0.01000000",
        "status":"NEW",
        "executedQty":"0.00000000",
        "isWorking":true,
        "symbol":"BTCUSDT",
        "price":"7000.00000000",
        "icebergQty":"0.00000000"
    }
}
```

A JavaScript example of using this API, which will buy until your account has 10 coins:

```
function main(){
    while(true){
        var amount = exchange.GetAccount().Stocks;
        var ticker = exchange.GetTicker();
        if(10-amount>0.01){
            var id = exchange.Buy(ticker.Sell, Math.min(10-amount,1));
        }else{
            return;
        }
        var status = exchange.GetOrder(id).Status;
        if(Status == ORDER_STATE_PENDING){
            exchange.CancelOrder(id);
        }
        Sleep(3000);
    }
}
```

Note: `Buy`, `Sell`, `CancelOrder`, can be followed by some additional output parameters, such as: `exchange.Buy(price, amount, 'BTC_USDT')`, `exchange.CancelOrder(orderId, 'BTC_USDT')`, Which will give you extra information in robot Logs.

7.6 2.4.6 GetOrders

```
exchange.GetOrders()
```

Get all Current open orders for your trading pair.

Return value: Order structure array

Example of GetOrders from binance, the trading pair is `YOYOETH`:

```
[
    {
        "Info":{
                "executedQty":"0.00000000",
                "type":"LIMIT",
                "isWorking":true,
```

(continues on next page)

(continued from previous page)

```
                "price":"0.00012826",
                "status":"NEW",
                "timeInForce":"GTC",
                "symbol":"YOYOETH",
                "side":"SELL",
                "stopPrice":"0.00000000",
                "icebergQty":"0.00000000",
                "time":1530331666316,
                "orderId":16387538,
                "origQty":"1123.00000000",
                "clientOrderId":"TrKOsaHcqc667tjZQtg09b"
                },
        "Id":16387538,
        "Amount":1123,
        "Price":0.00012826,
        "DealAmount":0,
        "AvgPrice":0,
        "Status":0,
        "Type":1,
        "ContractType":""
    }
]
```

A JavaScript example of using this API, which will cancel all open orders for set trading pair:

```
fuction CancelAll(){
    var orders = exchange.GetOrders();
    for(var i=0;i<orders.length,i++){
        exchange.CancelOrder(orders[[i].Id);
    }
}
function main(){
    CancelAll();
    while(true){
        //do something
        Sleep(10000);
    }
}
```

7.7 2.4.7 SetContractType

```
exchange.SetContractType(ContractType)
```

Set contract type for futures trade. must be set first before using other API.

Parameter value: string type

OKEX futures have “this_week”, “next_week”, “quarter” three parameters

```
exchange.SetContractType("this_week"); // Set to Weekly Contract
```

7.8 2.4.8 GetPosition

```
exchange.GetPosition()
```

Get the current position information, only for Futures trade. OKEX can pass a parameter, specify the type of contract to get.

Return value: position array

BTC Futures support: `OKEX`, `BitMEX`.

The position structure contains the following variables:

Field	Type	Description
Info	Object	The original data returned by the exchange
MarginLevel	Number	Leverage size, OKEX is 10 or 20, full amount of okex futures margin mode returns a fixed 10, because the original API does not support
Amount	Number	ositions, OKEX indicates the number of contracts (integer and greater than 1)
FrozenAmount	Number	Position freeze
Price	Number	Average price of positions
Profit	Number	Order Status
Type	Const	`PD_LONG` is a long position, `PD_SHORT` is a short position.
ContractType	String	Contract name

Postion Type is global constant:

Global constant	Meaning	value
PD_LONG	long	0
PD_SHORT	short	1

A JavaScript example:

```
function main(){
    exchange.SetContractType("this_week") //for OKEX future
    var position = exchange.GetPosition()
    if(position.length>0){
        Log("Amount:", position[0].Amount, "FrozenAmount:", position[0].FrozenAmount,
↪"Price:",
            position[0].Price, "Profit:", position[0].Profit, "Type:", position[0].
↪Type, "ContractType:", position[0].ContractType)
    }
}
```

7.9 2.4.9 SetMarginLevel

```
exchange.SetMarginLevel(MarginLevel)
```

Set the leverage size, only for Futures trade.

Parameter value: number integer

Set the leverage size of Buy or Sell. MarginLevel has 5, 10, 20 optional parameters. OKEX supports 10 times and 20 times. For example: `exchange.SetMarginLevel(10)`

7.10 2.4.10 SetDirection

```
exchange.SetDirection(Direction)
```

Set Buy or Sell Order Types, only for Futures trade.

Parameter value: string type, can be `buy`, `closebuy`, `sell`, `closesell`.

A JavaScript example:

```
function main(){
    exchange.SetContractType("this_week");
    exchange.SetMarginLevel(5); // Set the leverage to 5 times
    exchange.SetDirection("buy"); // Set the order type to buy long
    exchange.Buy(1000, 2); //buy long at the price 1000, quantity of 2
    exchange.SetDirection("closebuy");
    exchange.Sell(1000, 2); //close long position
}
```

CHAPTER 8

2.5 Extended API

The following functions is used for extending your API. Most of the time, your code can be wrote by the API above.

8.1 2.5.1 IO

exchange.IO() has different usages depends on parameters.

Change trading pair

```
exchange.IO("currency", symbol)
```

Parameter value: `symbol` , trading pair to switch.

Example:`exchange.IO("currency", "LTC_USDT")`, which will switch the default trading pairs configured by the robot when it was created to `LTC_USDT`.

A JavaScript example of using `IO` to trade several trading pairs:

```
var symbols = ["BTC_USDT", "LTC_USDT", "EOS_USDT", "ETH_USDT", "BCC_USDT"];
var buyValue = 1000;
function main(){
    for(var i=0;i<symbols.length;i++){
        exchange.IO("currency", symbols[i]);
        var ticker = exchange.GetTicker();
        var amount = _N(buyValue/ticker.Sell, 3);
        exchange.Buy(ticker.Sell, amount);
        Sleep(1000);
    }
}
```

Change exchange base API address

```
exchangs.IO("base", baseAddress)
```

Parameter value: `baseAddress` , exchange base API address to switch.

If an exchange change it's base API address, you can use this function to change the base address temporarily till FMZ change it.

In some case, changing the address can acess another part of a exchange, such as huobipro and hadax, their API is the same, just switching the base address you can trade on hadax. `exchange.IO("base", "https://api.hadax.com")`

Use exchange's orgin API

```
exchange.IO("api", httpMethod, resource, params)
```

Acesss to exchange's other API.

Note: Only used for those API methods needed secret key to sign, other methods can be acessed by `HttpQuery(url)` function.

Using this function requires the understanding of exchange's orgin API, it extend the functionality that the FMZ does not add (to submit a POST request without having to worry about the parameter encryption process, the FMZ has completed the encryption already, just need fill in the corresponding parameters). For example, the FMZ platform does not currently support margin leverage trading on bitfinex exchanges. We can implement this function by using the IO function.

- First find bitfinex API description webpage: bitfinex.
- Then we know that the order is interacting with a POST request, so we pass the parameter httpMethod to the order address of the "POST" margin transaction: ' https://api.bitfinex.com/v1/order/new '. Because the FMZ has internally specified the root address, we only need to pass the value of the parameter resource to "/v1/order/new".
- Then the params parameter is not filled in. The params variable represents the information to be exchanged. We can send all kinds of information with the "&" symbol to send them. We first go to bitfinex to see that the next buy or sell order requires 5 parameters., they are: symbol, amount, price, side, type. We assign these five parameters respectively. If we want to buy Litecoin LTC, the quantity is 1, the price is 10, and the margin trading mode, then we can construct such a string: "symbol=ltc&amount=1&price=10&side=buy&type= Limit".

The final JavaScript code:

```
function main(){
    exchange.IO("api","POST","/v1/order/new","symbol=ltc&amount=1&price=10&side=buy&
↪type=limit");
}
```

An OKEX Example

```
function main(){
    var ret = exchange.IO("api", "POST", "/api/v1/future_position.do", "symbol=eth_
↪usd&contract_type=this_week");
        Log(ret);
}
```

8.2 2.5.2 Go

```
exchange.Go(Method, Args)
```

Multi-threaded asynchronous support functions that can convert the operations of all supported functions into asynchronous concurrency.

Parameter value:

```
Method : a function name.
Args   : the args of method.
```

Supported Functions: `GetTicker`, `GetDepth`, `GetTrades`, `GetRecords`, `GetAccount`, `GetOrders`, `GetOrder`, `CancelOrder`, `Buy`, `Sell`, `GetPosition`

robot thread must obtain the result from the wait function, the docker automatically releases the thread resource requested through the Go function. If the return result of the wait function is not obtained, the thread resource will not be automatically released, which will cause threads to accumulate, and more than 2000 will report an error. "too many routine wait, max is 2000"

A JavaScript example

```
function main(){
    var a = exchange.Go("GetTicker"); //GetTicker Asynchronous multithreaded execution
    var b = exchange.Go("GetDepth");
    var c = exchange.Go("Buy", 1000, 0.1);
    var d = exchange.Go("GetRecords", PERIOD_H1);
    // The above four operations are concurrent multi-threaded asynchronous execution,
↪ will not be time-consuming and immediately return
    var ticker = a.wait(); // Call wait method wait for return to asynchronous get␣
↪ticker result
    var depth = b.wait(); // Return depth, it is also possible to return null if it␣
↪fails
    var orderId = c.wait(1000); // Return the order number, limit 1 second timeout,␣
↪timeout returns undefined, this object can continue to call wait until the last␣
↪wait timeout
    var records = d.wait(); // Wait for K-line result
    var ret = d.wait();  // Here waits for an asynchronous operation that has waited␣
↪and ended, returns null, and logs an error message.
}
```

The difference between Python and JavaScript, Python's wait returns two parameters, the first is the result of the asynchronous API, and the second is whether the asynchronous call is completed.

```
ret, ok = d.wait(); // Ok is bound to return true unless the strategy is stopped
ret, ok = d.wait(100); // Ok returns False, waits for a timeout, or waits for an␣
↪instance that has ended
```

Note: This function only creates multi-threaded execution tasks when it runs on a real market. Backtesting does not support multithreaded concurrent execution of tasks (backtesting is available, but it is also performed sequentially).

8.3 2.5.2 GetRawJSON

```
exchange.GetRawJSON()
```

Returning of the original content (string) that returned by the last REST API request, which can be used to resolve extension information on its own.

There are a lot of inforamtions in the raw data returned from exchange, part of them are in `Info` Field, if not, you can use this API.

Return value : string type

A JavaScript example of using GetRawJSON and parse the raw data:

```
function main(){
    exchange.GetAccount();
    var obj = JSON.parse(exchange.GetRawJSON());
    Log(obj);
}
```

8.4 2.5.3 GetName

```
exchange.GetName()
```

Returns the name of the exchange.

Return value: string type.

8.5 2.5.4 GetLabel

```
exchange.GetLabel()
```

Return the exchange's custom label.

Return value: string type

8.6 2.5.4 GetCurrency

```
exchange.GetCurrency()
```

Returns the name of the currency pair operated by the exchange.

Return value: string type

8.7 2.5.5 GetQuoteCurrency

```
exchange.GetQuoteCurrency()
```

Returns the base currency name of the exchange operation, eg BTC_CNY returns CNY, ETH_BTC returns BTC.

Return value: string type

8.8 2.5.6 SetPrecision

```
exchange.SetPrecision(PricePrecision, AmountPrecision)
```

Set the decimal precision of price and type order quantity, and will automatically truncate after setting.

If you use `exchange.Sell(7000.1225,1.223123)` on Bianace directly, which will return error `{"code":-1013,"msg":"Filter failure: PRICE_FILTER"}`. that's why you should care about precision.

You can also use `_N()` function as `exchange.Sell(_N(price,2), _N(amount,5))`

You can find the demands of precision and others in exchange's docs, for example: https://api.binance.com/api/v1/exchangeInfo

Parameter value:

`PricePrecision` , number type, used to control the decimal point of the price.

`AmountPrecision` , number type, used to control the decimal point of the amount.

```
exchange.SetPrecision(2, 3);
function main(){
    exchange.Sell(7000.1225,1.223123)   //which will be the same as exchange.
↪Sell(7000.12,1.223)
}
```

Note: SetPrecision doesn't work in backtesting

8.9 2.5.7 GetRate

```
exchange.GetRate()
```

Returning of the exchange rate that between the exchange currency and the current display currency. Returning 1 means currency conversion is not allowed.

Return value: number type

Note: If you do not call exchange.SetRate() to set the conversion rate, GetRate defaults to the exchange rate value of 1, ie, the current displayed denomination currency has not been converted.

8.10 2.5.8 SetRate

```
exchange.SetRate(scale)
```

Parameter value: `scale` , number type

Return value: number type

If you use exchange.SetRate() to set an exchange rate value, such as 0.85(the rate of EUR and USD), then all exchange prices, depth, order price and all other price information in the current exchange currency represented by the exchange object will be multiplied by the setting.

8.11 2.5.8 SetProxy

```
exchange.SetProxy()
```

Switch to Proxy Server to Access Exchange

Each exchange object (exchanges[n]) can set up an agent. After setting up the agent, the access exchange API will be accessed through the agent.

```
// Take the exchange of the main exchange object as the first added exchange object,␣
↪ie: exchanges[0] as an example.
exchange.SetProxy("socks5://127.0.0.1:8889")                          // Set proxy, no␣
↪username, no password.
exchange.SetProxy("socks5://username:password@127.0.0.1:8889")     // Set up the proxy,
↪ enter the username and password
exchange.SetProxy("")                                                // Switch to normal␣
↪mode without using a proxy.
```

8.12 2.5.9 SetTimeout

```
exchange.SetTimeout(time)
```

Set timeout for exchange's rest request.

Only the REST request is used to set the timeout time.

For example: `exchange.SetTimeout(3000)`, set the exchange exchange object, send a rest request starts timing, exceeds 3 seconds, timeout returns null.

```
exchange.SetTimeout()
```

8.13 2.5.10 Log

```
exchange.Log(logType, orderId, price, amount)
```

Doesn't actually sent the order, just record order information for testing your strategy.

Parameter values:

```
logType : LOG_TYPE_BUY, LOG_TYPE_SELL, LOG_TYPE_CANCEL
orderId : order id, customizable an incremental value
price   : price
amount  : quantity
```

Return value: number type

Note: This function is a function of the exchange exchange object, which is different from the global function Log().

CHAPTER 9

2.6 Global Function

9.1 2.6.1 Log

```
Log(message)
```

Save a message to the robot logs.

Parameter value: message can be any type.

```
Log(123);
Log('hello', 'world', 123);
Log("red color message", "#FF0000");
Log("blue color message#FF0000");
```

Log supports pushing message to your WeChat account.

adding the @ character to the string, the message will go to the push queue and be pushed to the WeChat account that uses the binding (bind in account security) (50/hour, 1/5 second limit)

```
Log("hello Wechat!@");
```

Log supports base64-encoded pictures print:

```
Log("``");
```

Log supports Python matplotlib.pyplot object print directly. as long as the object contains `savefig` method, such as:

```
import matplotlib.pyplot as plt

def main():
    plt.plot([3,6,2,4,7,1])
    Log(plt)
```

Log supports present message's character or background with custom color:

Change message's character color: message need end with RGB color code such as `##FF0000`, for more RGB color, check on https://www.w3schools.com/colors/colors_picker.asp.

Change message's background color: message need end with like `#FF0000CC3299` ,the last six number `CC3299` represnt the RGB color code.

```
Log("red color message", "#FF0000");
Log("blue color message#FF0000CC3299");
```

9.2 2.6.2 LogStatus

```
LogStatus(Msg)
```

This kind of Log information is displayed above the log and update every time when it is called.

Parameter value: Msg can be any type.

```
LogStatus(" This is a normal status prompt");
LogStatus(" This is a red font status prompt #ff0000");
LogStatus(" This is a multi-line status message\n I'm the second line");
```

Like `Log()` function, `LogStatus` supports base64-encoded images and Python matplotlib.pyplot object.

LogStatus can Log tables on your robot page.

Log a table example, add ` characters to both sides and treat it as a complex message format (currently supported table).

```
var table = {type: 'table', title: ' Account information support color #ff0000',␣
↪cols: ['BTC', 'ETH', 'USDT'], rows: [ ['free', 1, 2000], ['frozen', 0, 3000]]};
LogStatus('`' + JSON.stringify(table)+'`');
```

Another example, information can also appear in multiple lines:

```
LogStatus("First line message\n" + JSON.stringify(table)+"`\n third line message");`
```

Log multiple tables in a group, switching by TAB:

```
var table1 = {type: 'table', title: ' Account information 1', cols: ['BTC', 'ETH',
↪'USDT'], rows: [ ['free', 1, 2000], ['frozen', 0, 3000]]};
var table2 = {type: 'table', title: ' Account information 2', cols: ['BTC', 'ETH',
↪'USDT'], rows: [ ['free', 1, 2000], ['frozen', 0, 3000]]};
LogStatus('`' + JSON.stringify([table1, table2])+'`'); // Supports multiple tables to␣
↪be displayed at the same time and will be displayed in a group with TAB
```

Log multiple tables in one page:

```
function main(){
    var tab1 = {type : "table",title : "Table 1",cols : ["1", "2"],rows : []};
    var tab2 = {type : "table",title : "Table 2",cols : ["1", "2", "3"],rows : []};
    tab1.rows.push(["jack", "lucy"]);
    tab2.rows.push(["apple", "pen", "apple pen"]);
    LogStatus('`' + JSON.stringify(tab1) + '`\n' + '`' + JSON.stringify(tab2) + '`');
}
```

Log table with a button in the table. The strategy uses `GetCommand` to receive the contents of the cmd property.

```
var table = {
    type: 'table',
    title: 'Positioning operations',
    cols: ['Column 1', 'Column 2', 'Action'],
    rows: [
        ['abc', 'def', {'type':'button', 'cmd': 'coverAll', 'name': 'Close the␣
↪position'}],
    ]
};
LogStatus('`' + JSON.stringify(table) + '`')
// Or construct a separate button
LogStatus('`' + JSON.stringify({'type':'button', 'cmd': 'coverAll', 'name': ' Close␣
↪the position'}) + '`')
// Can customize button styles (bootstrap button properties)
LogStatus('`' + JSON.stringify({'type':'button', 'class': 'btn btn-xs btn-danger',
↪'cmd': 'coverAll', 'name': 'close the position'}) + '`')
```

9.3 2.6.3 LogProfit

```
LogProfit(Profit)
```

Record profit value, draw a line chart in your robot page, will remain after you restart your robot.

Parameter value: profit , number type

A useful JavaScript example of Log Profit for a certain trading pair:

```
function GetValue(){
        var ticker = exchange.GetTicker();
        var account = exchange.GetAccount();
        var price = ticker.Buy;
        var stocks = account.Stocks + account.FrozenStocks;
        var balance = account.Balance + account.FrozenBalance;
        var value = stocks*price + balance;
        return value;
}
function main(){
    var initValue = GetValue();
    var profit = 0;
    while(true){
        profit = GetValue() - initValue;
        LogProfit(profit);
        Sleep(60000);//sleep one minute
    }
}
```

Note: `LogProfit` doesn't have to be recording the profit , it can be any number you like to present, such as total account value, free USDT amount. `Profit` is calculated by your own.

9.4 2.6.4 SetErrorFilter

```
SetErrorFilter(RegEx)
```

Error message filtering

Parameter value: string type

Errors that are matched by this regular expression will not be uploaded to the log system. Multiple set (filtered logs, database files corresponding to robot IDs in the logs/robot under the docker directory can be called multiple times to prevent frequent errors Causes database file expansion.)

```
SetErrorFilter(
↪"502:|503:|tcp|character|unexpected|network|timeout|WSARecv|Connect|GetAddr|no␣
↪such|reset|http|received|EOF|reused");
```

9.5 2.6.5 LogReset

```
LogReset()
```

Clear the log, you can pass a parameter, specify the number of recent logs to keep, clear the rest of the log.

9.6 2.6.6 LogProfitReset

```
LogProfitReset()
```

Clear all history logs, can take a number parameter, specify the number of reservations.

9.7 2.6.7 EnableLog

```
EnableLog(IsEnable)
```

Turn on or off logging of orders and error messages/

Parameter value: isEnable is bool type

9.8 2.6.8 LogVacuum

```
LogVacuum()
```

Reclaims the space occupied by SQLite when deleting data.

9.9 2.6.9 GetLastError

```
GetLastError()
```

Get the latest error message, generally do not need to use, because the program will automatically upload the error message to the log system.

Return value: string type

9.10 2.6.10 GetCommand

```
GetCommand()
```

Get Interactive Command (utf-8).

Get the command sent from the strategy interactive interface and clear it. If there is no command, it will return `null`. The returned command format is "Button name: parameter". If there is no parameter, the command is the button name.

A JavaScript example

```
function main(){
    while(true) {
        var cmd = GetCommand();
        if (cmd) {
            Log(cmd);
        }
        Sleep(1000);
    }
}
```

9.11 2.6.11 Sleep

```
Sleep(Millisecond)
```

Pause the robot program for a period of time.

Parameter value: Millisecond is number type

`Sleep(1000)` means sleep 1 second.

Warning: In almost all the situation, `Sleep` is necessary in `while` loops, otherwise you may exceed the exchange's API rate limits of REQUESTS.

9.12 2.6.12 IsVirtual

```
IsVirtual()
```

Your robot is runing in a simulated backtest or not.

Return value: bool type, Simulate back test state return true, the real market returns false

9.13 2.6.13 GetOS

```
GetOS()
```

Returns the information of the docker's system.

9.14 2.6.14 GetPid

```
GetPid()
```

Return robot process ID

Return value: string type

9.15 2.6.15 Mail

```
Mail(smtpServer, smtpUsername, smtpPassword, mailTo, title, body)
```

Send a e-mail.

Parameter values: all are string types

Return value: bool type, return true if successful

```
function main(){
    Mail("smtp.163.com", "test@163.com", "password", "usr@163.com", "title", "body");
}
```

9.16 2.6.16 Dial

```
Dial(Address, Timeout)
```

Get Original Socket access, support tcp, udp, tls, unix protocol.

Parameter value: Address is string type, fill in the address, TimeOut is the timeout

A JavaScript example:

```
function main(){
    var client = Dial("tls://www.baidu.com:443"); // Dial supports tcp://, udp://,␣
↪tls://, unix:// protocol, extra parameter to specify the number of seconds to␣
↪timeout
    if (client) {
        client.write("GET / HTTP/1.1\nConnection: Closed\n\n"); // Write can be␣
↪followed by a number parameter to specify the timeout, write to return the number␣
↪of bytes successfully sent
        while (true) {
            var buf = client.read();// Read can be followed by a number parameter to␣
↪specify a timeout, return null to indicate an error or timeout, or the socket is␣
↪already closed
            if (!buf) {
```

(continues on next page)

(continued from previous page)

```
                break;
            }
            Log(buf);
        }
    client.close();
    }
}
```

Support websocket.

A JavaScript example of connecting to binance websocket ticker.

```
function main() {
    LogStatus("connecting...");
    var client = Dial("wss://stream.binance.com:9443/ws/!ticker@arr");
    if (!client) {
        Log("Connection failed, program exited");
        return
    }
    Log("The connection is successful and will automatically reconnected")
    while (true) {
        var buf = client.read() // Read only returns data obtained after calling read
        if (!buf) {
            break;
        }
        var table = {
            type: 'table',
            title: 'Quotes chart',
            cols: ['Currency', 'highest', 'lowest', 'buy one', 'sell one', ' Last␣
↪traded price', 'volume', 'Update time'],
            rows: [],
        };
        var obj = JSON.parse(buf);
        _.each(obj, function(ticker) {
            table.rows.push([ticker.s, ticker.h, ticker.l, ticker.b, ticker.a, ticker.
↪c, ticker.q, _D(ticker.E)])
        });
        LogStatus('`' + JSON.stringify(table) + '`')
    }
    client.close();
}
```

Here is another JavaScript example of connecting to OKEX websocket ticker. In this case, the bot needs to send a ping message every 30 seconds, and okex will send a pong back.

```
function main(){
    var ws = Dial("wss://real.okex.com:10441/websocket")
    if(ws){
        ws.write("{'event':'addChannel','channel':'ok_sub_spot_btc_usdt_ticker'}")
        var lastPingTime = new Date().getTime()
        while(1){
            var nowTime = new Date().getTime()
            var ret = ws.read(10000) //timeout is 10000ms
            Log("ret:", ret)
            if(nowTime - lastPingTime > 15000){
                var retPing = ws.write("{'event':'ping'}")
                lastPingTime = nowTime
```

(continues on next page)

(continued from previous page)

```
            }
            LogStatus("Now time", _D())
            Sleep(1000)
        }
    }
}
```

9.17 2.6.16 HttpQuery

```
HttpQuery(Url, PostData, Cookies, Headers, IsReturnHeader)
```

Web URL access, support `PUT`, `GET`, `POST`, `DELETE` ,etc.

Parameter values: all are string types

Get the content of a Url. If the second parameter PostData is a string, submit it as POST. The second parameter PostData can be a custom method such as:

```
HttpQuery("http://www.abc.com", {method:'PUT', data:'parameter1=value1&
↪parameter2=value2'});//PUT method
HttpQuery("http://www.abc.com", {method:'PUT', data:'parameter1=value1&
↪parameter2=value2', timeout:1000});
```

Passing the cookie string requires a third parameter, but does not require POST. Please set the second parameter to `null` When runing in the backtes, the function returns the fixed string Dummy Data because the URL cannot be simulated.

You can use this interface to send text messages or interact with other APIs

```
HttpQuery("http://www.google.com"); // Get
HttpQuery("http://www.google.com", "parameter1=value1&parameter2=value2"); // Post
HttpQuery("http://www.google.com", null, "a=10; b=20", "User-Agent: Mobile\nContent-
↪Type: text/html", true);
```

Example Accessing BIANACE APIs that do not require signatures:

```
var exchangeInfo = JSON.parse(HttpQuery('https://api.binance.com/api/v1/exchangeInfo
↪'));
Log(exchangeInfo);
var ticker = JSON.parse(HttpQuery('https://api.binance.com/api/v1/ticker/24hr'));
Log(ticker);
```

Note: The `HttpQuery` function only supports JavaScript, for Python, using the `urlib2` or `request` library to send http requests directly.

9.18 2.6.17 MD5

```
MD5(string)
```

Parameter value: string type

```
Log("MD5", MD5("hello world"))
```

9.19 2.6.18 Hash

```
Hash(Algo, OutputAlgo, Data)
```

Support hash calculation for md5/sha256/sha512/sha1, only supports in the real maket.

Parameter values: all are string types

The second parameter can be set to raw/hex/base64, which means output encrypted original content/hex encoded/base64 encoded.

```
function main(){
    Log(Hash("md5", "hex", "hello"));
    Log(Hash("sha512", "base64", "hello"));
}
```

9.20 2.6.19 HMAC

```
HMAC(Algo, OutputAlgo, Data, Key)
```

HMAC encryption calculation of md5/sha256/sha512/sha1 is supported, only supported in the real market.

Parameter values: all are string types.

The second parameter can be set to raw/hex/base64, which means output encrypted original content/hex encoded/base64 encoded.

9.21 2.6.20 UnixNano

```
UnixNano()
```

Return the nanosecond timestamp. If you need to obtain the millisecond timestamp, you can use the following code:

```
var time = UnixNano() / 1000000;
Log(_N(time, 0));
```

9.22 2.6.21 Unix

```
Unix()
```

Returns the second-level timestamp, which is only supported in a strategy written in C++.

```
uint64_t t = Unix();
Log(t);
```

9.23 2.6.21 _C

```
_C(function, args...)
```

Retry function

Will always call the specified function to return successfully (function returns null or false will retry), such as `_C(exchange.GetTicker)`, the default retry interval is 3 seconds, you can call `_CDelay` function to control the retry interval, such as `_CDelay (1000)` means change the `_C` function retry interval to 1 second, suggesting

Support Function:

- `exchange.GetTicker()`
- `exchange.GetDepth()`
- `exchange.GetTrade()`\`
- `exchange.GetRecords()`
- `exchange.GetAccount()`
- `exchange.GetOrders()`
- `exchange.GetOrder()`

A JavaScript example:

```
function main(){
    var ticker = _C(exchange.GetTicker);
    var depth = _C(exchange.GetDepth);
    Log(ticker);
    Log(depth);
}
```

Note: `_C()` can often be misused as `_C(exchange.GetRecords(PERIOD_H1))`, should be `_C(exchange.GetRecords,PERIOD_H1)`

9.24 2.6.22 _G

Global dictionary that can be saved after restart robot.

The KV dict is permanently stored in the local file. Each robot has a separate database. After the restart or the escrow withdrawal, K must be a number or a string. It is case-insensitive and V can be any JSON serialized content.

A JavaScript example:

```
_G('initValue', 1000); // set value
var initValue = _G('initValue'); // get value
_G("initValue", null); // remove global variable "initValue"
_G(null); //Remove all global variables
_G(); // Returns the ID of the current robot
```

9.25 2.6.23 _D

```
_D(timestamp, fmt="yyyy-MM-dd hh:mm:ss")
```

Returns the specified timestamp

Returns the specified timestamp (ms) string, returns the current time without any parameter, such as _D(), or _D(1478570053241), The default format is yyyy-MM-dd hh:mm:ss.

```
function main(){
    while(true){
        var time = _D();
        LogStatus('Last update time: ', time);
        //do some thing
    }
}
```

9.26 2.6.24 _N

```
_N(num, precision)
```

Format a float

Parameter value, num is number type, precision is integer number

For example `_N (3.1415, 2)` will delete the value after the two decimal points, return "3.14". `_N(1321,-2)` will return "1300".

9.27 2.6.25 _Cross

```
_Cross(arr1, arr2)
```

Returns the number of cross periods of the arrays arr1 and arr2.

A positive number is the upper pass period, a negative number indicates the period of the wear pass, and a 0 indicates the current price. Can be used in MACD strategy.

Parameter value: array of numbers

Specific instructions for use: built-in function _Cross analysis and instructions

```
var arr1 = [1,2,3,4,5,6,8,8,9]
var arr2 = [2,3,4,5,6,7,7,7,7]
function main(){
    Log("_Cross(arr1, arr2) : ", _Cross(arr1, arr2))
    Log("_Cross(arr2, arr1) : ", _Cross(arr2, arr1))
}
```

9.28 2.6.26 TA Indicator function

TA-Lib Indicator Library. support `MACD`, `EMA`, `KDJ`, `ATR`, `RSI`, etc...

Need to add TA. or talib. prefix when calling indicator function.

For more details about TA-Lib functions, check on http://mrjbq7.github.io/ta-lib/

You can also install TA-lib library of Python by yourself.

JavaScript example:

```
function main(){
    var records = exchange.GetRecords();
    var macd = TA.MACD(records);
    Log("DIF:", macd[0], "DEA:", macd[1], "MACD:", macd[2]);
    var atr = TA.ATR(records, 14);
    // Print out the last row of values
    Log(macd[0][records.length-1], macd[1][records.length-1],
    macd[2][records.length-1]);
    Log(atr[atr.length-1]);
    Log(talib.MACD(records));
    Log(talib.MACD(records, 12, 26, 9));
    Log(talib.OBV(records));
    // Talib can also pass in an array of numbers, which can be passed in successively
    // Such as: OBV(Records[Close], Records[Volume]), need Close, Volume two array␣
↪parameters
    Log(talib.OBV([1,2,3], [7.1, 6.2, 3,3]));
    // You can also directly pass in an array of records containing the Close, Volume␣
↪property
    Log(talib.OBV(records));
    Log(TA.Highest(records, 30, 'High'));
    Log(TA.Highest([1,2,3,4], 0));
    // For Python, the system extends the properties of the array returned by␣
↪GetRecords, adding Open, High, Low, Close, Volume, to facilitate talib calls, such␣
↪as
    talib.MACD(records.Close);
    /*For Python, the system expands the properties of the array returned by␣
↪GetRecords, adds Open, High, Low, Close, Volume,
    and facilitates talib calls. For example, the Close property returns the Close␣
↪property of all records members as a numpy array passed to talib.
    The same as other properties*/
}
```

9.29 2.6.27 Chart

```
Chart({...})
```

Will draw a figure in you robot management page.

Chart is based on HighStocks. check on http://api.highcharts.com/highstock for more details.

The parameter is a HighCharts.StockChart parameter that can be JSON-serialized and has a `__isStock` attribute that is not exist in the original one. If `__isStock` is `false`, chart will displayed as an ordinary chart.

The return object can call `add([series index(like 0), data])` to add data to the specified index series, call `reset()` to clear the chart data, reset can take a number parameter, specify the number of reservations.

You can call `add([series index, data, index of data in the series])` to change the data, Can be negative, -1 refers to the last, -2 is the second to last, such as: `Chart.add([0, 13.5, -1])`, change the data of the first-to-last point of series[0].data.

Supports the display of multiple charts. You only need to pass in array parameters like: `var chart = Chart([{...}, {...}, {...}])`.

A JavaScript example of using Chart to draw the prices of two coins.two exchanges need to be added before run the robot.

```
// This chart is an object in the JS language. Before using the Chart function, we need to declare an object variable chart that configures the chart.
var chart = {
    // Whether the mark is a general chart, if you are interested, you can change it to false and run it.
    __isStock: true,
    tooltip: {xDateFormat: '%Y-%m-%d %H:%M:%S, %A'},     // Zoom tool
    title : { text : 'Spread Analysis Chart'},           // title
    rangeSelector: {                                    // Selection range
        buttons:  [{type: 'hour',count: 1, text: '1h'}, {type: 'hour',count: 3, text: '3h'}, {type: 'hour', count: 8, text: '8h'}, {type: 'all',text: 'All'}],
        selected: 0,
        inputEnabled: false
    },
    xAxis: { type: 'datetime'},                         // The horizontal axis of the coordinate axis is the x axis and the current setting type is :time
    yAxis : {                                           // The vertical axis of the axis is the y axis, and the default value is adjusted with the data size.
        title: {text: 'Spread'},                        // title
        opposite: false,                                // Whether to enable the right vertical axis
    },
    series : [                                          // Data series, this attribute is saved for each data series (line, K-line graph, label, etc...)
        {name : "line1", id : "Line 1,buy1Price", data : []},  // The index is 0, the data array is stored in the index series of data
        {name : "line2", id : "Line 2,lastPrice", dashStyle : 'shortdash', data : []},
        // The index is 1, dashStyle is set: 'shortdash' ie: Set the dotted line.
    ]
};
function main(){
    var ObjChart = Chart(chart);                    // Call the Chart function to initialize the chart.
    ObjChart.reset();                               // Empty the chart
    while(true){
        var nowTime = new Date().getTime();         // Get the timestamp of this poll, which is a millisecond timestamp. Used to determine the position of the X axis written to the chart.
        var tickerOne = _C(exchanges[0].GetTicker);   // Get market data
        var tickerTwo = _C(exchanges[1].GetTicker);
        ObjChart.add([0, [nowTime, tickerOne.Last]]); // Use the timestamp as the X value and buy the price as the Y value to pass the index 0 data sequence.
        ObjChart.add([1, [nowTime, tickerTwo.Last]]); // Same as above
        ObjChart.update(chart);                     // Update the chart to show it.
        Sleep(2000);
    }
}
```

9.30 2.6.28 Third-party Library

JavaScript:

- http://mathjs.org/
- http://mikemcl.github.io/decimal.js/
- http://underscorejs.org/
- http://ta-lib.org/

C++:

- https://nlohmann.github.io/json/

Python:

- install any libray you want.

CHAPTER 10

3.1 Use Websocket

For JavaScript:

```
function main() {
    LogStatus("connecting...");
    var client = Dial("wss://stream.binance.com:9443/ws/!ticker@arr");
    if (!client) {
        Log("Connection failed, program exited");
        return
    }
    Log("The connection is successful and the disconnected line is automatically␣
↪reconnected")
    while (true) {
        var buf = client.read() // Read only returns data obtained after calling read
        if (!buf) {
            break;
        }
        var table = {
            type: 'table',
            title: 'Quotes chart',
            cols: ['Currency', 'highest', 'lowest', 'Bid', 'Ask', ' Last traded price
↪', 'volume', 'Update time'],
            rows: [],
        };
        var obj = JSON.parse(buf);
        _.each(obj, function(ticker) {
            table.rows.push([ticker.s, ticker.h, ticker.l, ticker.b, ticker.a, ticker.
↪c, ticker.q, _D(ticker.E)])
        });
        LogStatus('`' + JSON.stringify(table) + '`')
    }
    client.close();
}
```

For Python, you can use `Dial()` function as well, or `websocket_client` library to access websocket.

CHAPTER 11

3.2 Use C++

There are some difference between C++ and JavaScript in writing strategy.

```
void main() {
    if (!Test("c++")) {
        Panic("please download the lastest docker");
    }
    // json: https://github.com/nlohmann/json
    // using Valid to judge return object is null or not.
    LogProfitReset();
    LogReset();
    Log(_N(9.12345, 2));
    Log("use _C", _C(exchange.GetTicker), _C(exchange.GetAccount));

    // test async
    auto routineTicker = exchange.Go("GetTicker");
    auto routineDepth = exchange.Go("GetDepth");
    Ticker asyncTicker;
    Depth asyncDepth;

    // No timeout, wait until Ticker is returned
    if (routineTicker.wait(asyncTicker)) {
        Log("Wait Ticker OK", asyncTicker.Valid, asyncTicker, asyncTicker.Last);
    } else {
        Log("Wait Ticker fail");
    }
    // With 300ms tiemout, 0 means return immediately.
    if (routineDepth.wait(asyncDepth, 200)) {
        // not sure the depth returned is valid, use asyncDepth.Valid.
        Log("Wait Depth OK", asyncDepth.Valid, asyncDepth);
    } else {
        Log("Wait Depth timeout");
    }

    auto records = _C(exchange.GetRecords);
```

(continues on next page)

(continued from previous page)

```
    Log("Last Kline", records[records.size()-1].Time, _D(records[records.size()-1].
↪Time), records[records.size()-1]);
    // Test TA
    auto ema = TA.EMA(records, 20);
    Log("EMA", ema[ema.size()-1], ema[ema.size()-2], ema[ema.size()-3]);

    auto macd = talib.MACD(records);
    Log("MACD", macd[0][macd[0].size()-1], macd[1][macd[1].size()-1], macd[2][macd[2].
↪size()-1]);

    //SetErrorFilter("timeout");
    Log(GetOS(), GetPid(), GetLastError(), "MD5", MD5("hello"));
    Log("Hash", Hash("md5", "hex", "hello"), Hash("sha512", "base64", "hello"));
    Log("HMAC", HMAC("sha512", "base64", "hello", "pass"));
    Log(Version(), Unix(), UnixNano());
    Log("Start Test Chart");
    Chart c = Chart(R"EOF({"chart":{"type":"line"},"title":{"text":"Simple Chart"},
↪"xAxis":{"title":{"text":"Date"}},"yAxis":{"title":{"text":"Number"}},"series":[{
↪"name":"number","data":[]}]})EOF");
    c.reset();
    for (size_t i = 0; i < 10; i++) {
        c.add(0, {(Unix() + i)*1000, rand() % 100});
    }
    Log(exchange.GetName(), exchange.GetLabel(), exchanges.size());
    auto acc = exchange.GetAccount();
    if (acc.Valid) {
        Log(acc);
    }

    // Using LogStatus and json draw a table, learn more about json libray: https://
↪github.com/nlohmann/json
    json tbl = R"({"type" : "table", "title" : "AAA", "cols" : ["Head1", "Head2"],
↪"rows": []})"_json;
    tbl["rows"].push_back({"111", "222"});
    tbl["rows"].push_back({"col2", "col22"});
    LogStatus("`"+tbl.dump()+"`");

    auto ticker = exchange.GetTicker();
    if (ticker.Valid) {
        Log(ticker);
        Log(ticker.Info); // Info Struct is json object
    }

    auto d = exchange.GetDepth();
    if (d.Valid) {
        Log(d.Asks[0], d.Bids[0]);
    }
    // Test features
    if (exchange.GetName() == "Futures_OKCoin") {
        exchange.SetContractType("this_week");
        exchange.SetMarginLevel(20);
        exchange.SetDirection("closebuy");

        auto positions = exchange.GetPosition();
        if (positions.Valid) {
            Log(positions);
        }
```

(continues on next page)

(continued from previous page)

```
    }
    // test other function
    Log("HttpQuery", HttpQuery("http://www.baidu.com/404").size());
    auto obj = json::parse(HttpQuery("http://www.baidu.com/404", "", "", "", true));
    string body = obj["Body"];
    Log("HttpQuery", body.size(), obj["Header"].dump());
    Log(Mail("smtp://smtp.163.com", "test@163.com", "password", "admin@163.com",
↪"title", "test c++ email"));
    // Test Dial
    auto client = Dial("tcp://www.baidu.com:80");
    if (client.Valid) {
        client.write("GET / HTTP/1.1\nHost: www.baidu.com\nConnection: Close\n\n");
        while (true) {
            string buf = client.read();
            if (buf == "") {
                break;
            }
            Log("Dial receive", buf.size());
        }
        client.close();
    }

    _G("OK","xxx");
    Log(_G("OK"));
    _G("OK","yyyyyy");
    Log(_G("OK"));
    _G("OK",NULL);
    Log(_G("OK"));
}
```

CHAPTER 12

3.3 Sell ALL AltCoin to BTC in Binance

This strategy will sell all your AltCoin to BTC (or ETH, BNB, USDT), learn how to trade multiple trading pair, and follow the price and amount filter of exchange.

```
function main() {
    var quoteCurrency = 'BTC'; //Can be 'ETH', 'BNB', 'USDT'
    Log("Quote Currency", quoteCurrency);
    var symbols = JSON.parse(_C(HttpQuery, "https://api.binance.com/api/v1/
↪exchangeInfo")).symbols;
    _.each(_C(exchange.GetAccount).Info.balances, function(ele) {
        if (ele.asset == quoteCurrency) {
            return
        }
        var totalV = parseFloat(ele.free) + parseFloat(ele.locked);
        if (totalV == 0) {
            return;
        }
        var cfg = _.findWhere(symbols, {symbol: ele.asset+quoteCurrency});
        if (!cfg) {
            Log("Not found", ele.asset, "trading pair", ele);
            return;
        }
        var filter = _.findWhere(cfg.filters, {filterType: "LOT_SIZE"});
        if (!filter) {
            return;
        }

        var v = _N(parseInt(totalV/filter.stepSize)*filter.stepSize, cfg.
↪baseAssetPrecision);
        if (v > 0) {
            Log(ele, "stepSize", filter.stepSize);
            exchange.IO("currency", ele.asset + "_"+quoteCurrency);
            while (true) {
                var orders = _C(exchange.GetOrders);
                _.each(orders, function(order) {
```

(continues on next page)

(continued from previous page)

```
                    exchange.CancelOrder(order.Id);
                });
                if (orders.length == 0) {
                    break;
                }
            }
            exchange.Sell(-1, v);
            Log(ele);
        }
    });
    Log("Done, Now", quoteCurrency, "Balance", _C(exchange.GetAccount).Balance);
}
```

CHAPTER 13

3.4 Moving Average Strategy

The moving average (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated average price.

Check on https://www.fmz.com/strategy/103070 for parameters configs.The global variables `FastPeriod`, `SlowPeriod`,etc... are defined in configs.

Source code:

```
function main() {
var initAccount = _C(exchange.GetAccount);//using _C() to retry
var ticker = exchange.GetTicker();
//calc InitValue to log profit
var InitValue = (initAccount.Stocks + initAccount.FrozenStocks)*ticker.Last \
              + initAccount.Balance + initAccount.FrozenBalance;
while (true) {
    var records = _C(exchange.GetRecords);
    ticker =_C(exchange.GetTicker);
    var FastRecords = TA.MA(records,FastPeriod);// using MA of TA-Lib
    var SlowRecords = TA.MA(records,SlowPeriod);
    var NowAccount = _C(exchange.GetAccount);
    var n = _Cross(FastRecords, SlowRecords);// using _Cross(), check on global␣
↪function
    if (n >= EnterPeriod && NowAccount.Balance > 0) {
        var Price = _N(ticker.Sell+Slippage, 2); // add Slippage to make sure order␣
↪can be done
        var Amount = _N(0.99*NowAccount.Balance/Price, 3);
        if(Amount>0.1){
           var id = exchange.Buy(Price, Amount);
           //Cancel the pending order
           if(exchange.GetOrders(id).Status == ORDER_STATE_PENDING){exchange.
↪CancelOrder(id);}
           LogProfit((NowAccount.Stocks + NowAccount.FrozenStocks)*ticker.Last\
                     + NowAccount.Balance + NowAccount.FrozenBalance - InitValue);
        }
    }
```

(continues on next page)

(continued from previous page)

```
    if(n <= -EnterPeriod && NowAccount.Stocks > 0) {
        var Price = _N(ticker.Buy-Slippage, 2);
        var Amount = _N(NowAccount.Stocks, 3);
        if(Amount>0.1){
            var id = exchange.Sell(Price, Amount);
            if(exchange.GetOrders(id).Status == ORDER_STATE_PENDING){exchange.
↪CancelOrder(id);}
            LogProfit((NowAccount.Stocks + NowAccount.FrozenStocks)*ticker.Last\
                    + NowAccount.Balance + NowAccount.FrozenBalance - InitValue);
        }
    }
    Sleep(Interval*1000);
}
```

}

CHAPTER 14

3.5 Iceberg Buy Order

Check on https://www.fmz.com/strategy/103319.

Source code:

```
var floatAmountBuy = 20
var floatAmountSell = 20
var diffPrice = 3
var Interval = 3000

function CancelPendingOrders() {
    var orders = _C(exchange.GetOrders);
    for (var j = 0; j < orders.length; j++) {
        exchange.CancelOrder(orders[j].Id, orders[j])
    }
}

function GetPrice(depth) {
    var price = {buy:0, sell:0}
    var askAmount = 0
    var bidAmount = 0
    for(var i=0; i<depth.Bids.length; i++){
        askAmount += depth.Asks[i].Amount
        bidAmount += depth.Bids[i].Amount
        if(askAmount >= floatAmountBuy && !price.buy){
            price.buy = depth.Asks[i].Price
        }
        if(bidAmount >= floatAmountSell && !price.sell){
            price.sell = depth.Bids[i].Price
        }
    }
    if(!price.buy || !price.sell){
        price = {buy:depth.Asks[depth.Asks.length-1].Price, sell:depth.Bids[depth.
↪Bids.length-1].Price}
    }
    return price
```

(continues on next page)

(continued from previous page)

```
}

function onTick() {
    var price = GetPrice(_C(exchange.GetDepth))
    var buyPrice = price.buy + 0.01
    var sellPrice = price.sell - 0.01
    if ((sellPrice - buyPrice) <= diffPrice){
        buyPrice -= 10
        sellPrice += 10
    }
    CancelPendingOrders()
    var account = _C(exchange.GetAccount)
    var amountBuy = _N((account.Balance / buyPrice-0.01), 2)
    var amountSell = _N((account.Stocks), 2)
    if (amountSell > 0.02) {
        exchange.Sell(sellPrice, amountSell)
    }
    if (amountBuy > 0.02) {
        exchange.Buy(buyPrice, amountBuy)
    }
}

function main() {
    while (true) {
        onTick()
        Sleep(Interval)
    }
}
```

CHAPTER 15

3.6 Dual Thrust OKEX Feature

A classic breakout strategy, Check on https://www.fmz.com/strategy/103247 for configs.

You can learn how to trade features and draw charts from the source code.

learn more on https://www.quantconnect.com/tutorials/strategy-library/dual-thrust-trading-algorithm

Source code:

```
var ChartCfg = {
    __isStock: true,
    title: {
        text: 'Dual Thrust Up-Down Track'
    },
    yAxis: {
        plotLines: [{value: 0,
            color: 'red',
            width: 2,
            label: {
                text: 'Up Track',
                align: 'center'}
                },
            {value: 0,
            color: 'green',
            width: 2,
            label: {
                text: 'Down Track',
                align: 'center'},
            }
        ]
    },
    series: [{type: 'candlestick',
        name: 'current cycle',
        id: 'primary',
        data: []
        },
```

(continues on next page)

(continued from previous page)

```
        {type: 'flags',
        onSeries: 'primary',
        data: [],
        }
    ]
};

var STATE_IDLE = 0;
var STATE_LONG = 1;
var STATE_SHORT = 2;
var State = STATE_IDLE;

var LastBarTime = 0;
var UpTrack = 0;
var BottomTrack = 0;
var chart = null;
var InitAccount = null;
var LastAccount = null;
var Counter = {
    w: 0,
    l: 0
};


function GetPosition(posType) {
    var positions = exchange.GetPosition();
    for (var i = 0; i < positions.length; i++) {
        if (positions[i].Type === posType) {
            return [positions[i].Price, positions[i].Amount];
        }
    }
    return [0, 0];
}

function CancelPendingOrders() {
    while (true) {
        var orders = exchange.GetOrders();
        for (var i = 0; i < orders.length; i++) {
            exchange.CancelOrder(orders[i].Id);
            Sleep(Interval);
        }
        if (orders.length === 0) {
            break;
        }
    }
}

function Trade(currentState, nextState) {
    var pfn = nextState === STATE_LONG ? exchange.Buy : exchange.Sell;
    if (currentState !== STATE_IDLE) {
        exchange.SetDirection(currentState === STATE_LONG ? "closebuy" : "closesell");
        while (true) {
            var amount = GetPosition(currentState === STATE_LONG ? PD_LONG : PD_
↪SHORT)[1];
            if (amount === 0) {
                break;
            }
```

(continues on next page)

(continued from previous page)

```
            // pfn(amount);
            pfn(nextState === STATE_LONG ? _C(exchange.GetTicker).Sell * 1.001 : _
↪C(exchange.GetTicker).Buy * 0.999, amount);
            Sleep(Interval);
            CancelPendingOrders();
        }
        var account = exchange.GetAccount();

        if (account.Stocks > LastAccount.Stocks) {
            Counter.w++;
        } else {
            Counter.l++;
        }

        LogProfit(_N(account.Stocks - InitAccount.Stocks), "Profit rate:", _
↪N((account.Stocks - InitAccount.Stocks) * 100 / InitAccount.Stocks) + '%');
        LastAccount = account;
    }
    exchange.SetDirection(nextState === STATE_LONG ? "buy" : "sell");
    while (true) {
        var pos = GetPosition(nextState === STATE_LONG ? PD_LONG : PD_SHORT);
        if (pos[1] >= AmountOP) {
            Log("Average Price", pos[0], "amount:", pos[1]);
            break;
        }
        // pfn(AmountOP-pos[1]);
        pfn(nextState === STATE_LONG ? _C(exchange.GetTicker).Sell * 1.001 : _
↪C(exchange.GetTicker).Buy * 0.999, AmountOP-pos[1]);
        Sleep(Interval);
        CancelPendingOrders();
    }
}

function onTick(exchange) {
    var records = exchange.GetRecords();
    if (!records || records.length <= NPeriod) {
        return;
    }
    var Bar = records[records.length - 1];
    if (LastBarTime !== Bar.Time) {
        var HH = TA.Highest(records, NPeriod, 'High');
        var HC = TA.Highest(records, NPeriod, 'Close');
        var LL = TA.Lowest(records, NPeriod, 'Low');
        var LC = TA.Lowest(records, NPeriod, 'Close');

        var Range = Math.max(HH - LC, HC - LL);

        UpTrack = _N(Bar.Open + (Ks * Range));
        DownTrack = _N(Bar.Open - (Kx * Range));
        if (LastBarTime > 0) {
            var PreBar = records[records.length - 2];
            chart.add(0, [PreBar.Time, PreBar.Open, PreBar.High, PreBar.Low, PreBar.
↪Close], -1);
        } else {
            for (var i = Math.min(records.length, NPeriod * 3); i > 1; i--) {
                var b = records[records.length - i];
                chart.add(0, [b.Time, b.Open, b.High, b.Low, b.Close]);
```

(continues on next page)

(continued from previous page)

```
            }
        }
        chart.add(0, [Bar.Time, Bar.Open, Bar.High, Bar.Low, Bar.Close]);
        ChartCfg.yAxis.plotLines[0].value = UpTrack;
        ChartCfg.yAxis.plotLines[1].value = DownTrack;
        ChartCfg.subtitle = {
            text: 'Up Track: ' + UpTrack + '  Down Track: ' + DownTrack
        };
        chart.update(ChartCfg);
        chart.reset(PeriodShow);

        LastBarTime = Bar.Time;
    } else {
        chart.add(0, [Bar.Time, Bar.Open, Bar.High, Bar.Low, Bar.Close], -1);
    }

    LogStatus("Price:", Bar.Close, "Up:", UpTrack, "Down:", DownTrack, "Wins: ",␣
↪Counter.w, "Losses:", Counter.l, "Date:", new Date());
    var msg;
    if (State === STATE_IDLE || State === STATE_SHORT) {
        if (Bar.Close >= UpTrack) {
            msg  = 'Long Price: ' + Bar.Close + ' Up Track:' + UpTrack;
            Log(msg);
            Trade(State, STATE_LONG);
            State = STATE_LONG;
            chart.add(1, {x:Bar.Time, color: 'red', shape: 'flag', title: 'Long',␣
↪text: msg});
        }
    }

    if (State === STATE_IDLE || State === STATE_LONG) {
        if (Bar.Close <= DownTrack) {
            msg = 'Short Price: ' + Bar.Close + ' Down Track:' + DownTrack;
            Log(msg);
            Trade(State, STATE_SHORT);
            chart.add(1, {x:Bar.Time, color: 'green', shape: 'circlepin', title:
↪'Short', text: msg});
            State = STATE_SHORT;
        }
    }
}

function onexit() {
    var pos = exchange.GetPosition();
    if (pos.length > 0) {
        Log("Warning, has positions when exiting", pos);
    }
}

function main() {
    if (exchange.GetName() !== 'Futures_OKCoin') {
        throw "Only support OKEX features";
    }
    exchange.SetRate(1);
    exchange.SetContractType(["this_week", "next_week", "quarter"][ContractTypeIdx]);
    exchange.SetMarginLevel([10, 20][MarginLevelIdx]);
```

(continues on next page)

(continued from previous page)

```
    if (exchange.GetPosition().length > 0) {
        throw "Can't have Positions when start.";}

    CancelPendingOrders();

    InitAccount = LastAccount = exchange.GetAccount();
    LoopInterval = Math.min(1, LoopInterval);
    Log('Exchange Name:', exchange.GetName(), InitAccount);
    LogStatus("Ready...");

    LogProfitReset();
    chart = Chart(ChartCfg);
    chart.reset();

    LoopInterval = Math.max(LoopInterval, 1);
    while (true) {
        onTick(exchange);
        Sleep(LoopInterval * 1000);
    }
}
```

CHAPTER 16

3.7 High Frequency Marketmaker

This is a simple but powerful strategy that used to earn thousands of times in real BTC spot markets. Can't run it on exchanges that have high trade fee.

Source code:

```
var floatAmountBuy = 20;
var floatAmountSell = 20;
var diffPrice = 3;
var Interval = 3000;

function CancelPendingOrders() {
    var orders = _C(exchange.GetOrders);
    for (var j = 0; j < orders.length; j++) {
        exchange.CancelOrder(orders[j].Id, orders[j]);}
}

function GetPrice(Type, depth) {
    var amountBids = 0;
    var amountAsks = 0;
    if(Type == "Buy"){
        for(var i=0;i<depth.Bids.length;i++){
            amountBids += depth.Bids[i].Amount;
            if (amountBids > floatAmountBuy){
                return depth.Bids[i].Price + 0.01;
            }
        }
    }
    if(Type == "Sell"){
        for(var j=0; j<depth.Asks.length; j++){
            amountAsks += depth.Asks[j].Amount;
            if (amountAsks > floatAmountSell){
                return depth.Asks[j].Price - 0.01;
            }
        }
    }
```

(continues on next page)

(continued from previous page)

```
    return depth.Asks[0].Price
}

function onTick() {
    var depth = _C(exchange.GetDepth);
    var buyPrice = GetPrice("Buy", depth);
    var sellPrice = GetPrice("Sell", depth);
    if ((sellPrice - buyPrice) <= diffPrice){
        buyPrice -= 10;
        sellPrice += 10;
    }
    CancelPendingOrders();
    var account = _C(exchange.GetAccount);
    var amountBuy = _N((account.Balance / buyPrice-0.01), 2);
    var amountSell = _N((account.Stocks), 2);
    if (amountSell > 0.02) {
        exchange.Sell(sellPrice, amountSell);
    }
    if (amountBuy > 0.02) {
        exchange.Buy(buyPrice, amountBuy);
    }
}

function main() {
    while (true) {
        onTick();
        Sleep(Interval);
    }
}
```