MT5 Pytrader_API.

Table of Contents

Changes

Date	Version	Changes
06-07-2021	V2.07	Added some functions.
05-04-2021	V2.06	Spitted the function get_all_closed_positions in two separate functions. Also added deleted pending orders parameters in the result. Check all functions for retrieving position/order functions
23-02-2021	V2.05	Cleared some small bugs. Changed open order with market watch parameter Added functions for resetting SL and TP for position and order Synchronized Pytrader_API version number and EA version number
20-01-2021	V2.03	Documentation update for “Get all closed positions”
28-12-2020	V2.03 V1_04	Updated for licensing Pytrader_API
10-12-2020	V2.02	Profit loss function added
22-11.2020	V2.01	Changed authorization, added authorization indicator for MT5 market place
11-11-2020	V1.06 V1_03	Added 2 new functions
09-10-2020	V1.05	Removed DLL for MT5 EA and updated documentation for python script
27-07-2020	V1.02	Added partial close of positions
20-07-2020	V1.01	Original version

Introduction

The MT Pytrader_API consist of 2 pieces of software:

- An EA running on MT5 terminal. This EA works as the socket server. The EA has to run all the time. The EA will react on requests from the “Pytrader_API”(python script). At the end of this document is explained how to install the EA on MT5 terminal.
- A python script, name “Pytrader_API”, which functions as the connection with the MT5 EA

Functions.

General

1. The MT Pytrader_API is coded as a class.
2. After the execution of a function, the *MT.command_OK* property will be set to *True* or *False*.

Time out is set to 60 seconds as default. There is a separate function to change the ‘time out’ time.
Input parameters/settings are in green, results are in blue.

1. Instantiation.

declaration

from utils.Pytrader_BT_API_V1_01 import Pytrader_BT_API

instantiate
MT = Pytrader_BT_API()

Utils is a subfolder in my python project. Can be any other sub folder, or just the main folder

2. Connect to server

At connection time a broker instrument dictionary has to be passed as a parameter. This dictionary is a lookup table for translating general instrument/symbol names into specific broker instrument/symbol names. This is for compatibility with *Pytrader_API*.

Instrument lookup dictionary, key=general instrument/symbol name, value=broker instrument/symbol name

brokerInstrumentsLookup = {'EURUSD':'EURUSD.ecn', 'GOLD':'XAUUSD', 'DAX':'GER30'}

connect to server local or to computer in same local network
Connected = MT.Connect(server='127.0.0.1', port=10014,
instrument_lookup=brokerInstrumentsLookup)
or
Connected = MT.Connect(server='192.168.0.103', port=10014,

instrument_lookup=brokerInstrumentsLookup)

'192.168.0.103' = server. In this case other computer in same local network.

11111 = port (number). Server socket of the MT5 EA must use same port.

brokerInstrumentLookup = dictionary

Connected = bool, *True* or *False*.
If connection is made the *MT.connected* property will be set to *True*. If no connection *MT.connected* property will be set to *False*.

3. Check connection.

CheckAlive= MT.Check_connection()

CheckAlive = bool, *True* or *False*.

4. Change time out value

Result = MT.Set_timeout(timeout_in_seconds=120)

120 = time out value in seconds.

Result = bool, always *True*

5. Retrieve broker server time.

ServerTime = MT.Get_broker_server_time()

ServerTime = broker server time.

6. Get static account information.

StaticInfo = MT.Get_static_account_info()

StaticInfo = dictionary with following information:

name=…..
login=11117869
currency=USD
type=demo
leverage=100
trade_allowed=True
limit_orders=200
margin_call=100.0
margin_close=50.0

7. Get dynamic account information

DynamicInfo = MT.Get_dynamic_account_info()

DynamicInfo = dictionary with the following information:

balance = float, 3400.0
equity = float, 3350.0
profit = float, -50.0
margin=40.6
margin_level=8106.05
margin_free=3101.64

8. Get instrument information

InstrumentInfo = MT.Get_instrument_info(instrument='EURUSD')

'EURUSD' = instrument.
InstrumentInfo = dictionary with the following information(if instrument not known, result is "none"):

Instrument = EURUSD
digits = 5
max_lotsize = 200.0
min_lotsize = 0.01
lot_step = 0.01
point = 1e-05
tick_size = 1e-05
tick_value = 1.0

9. Get last tick information

LastTick = MT.Get_last_tick_info(instrument='EURUSD')

LastTick = dictionary with the following information:

instrument=EURUSD
date=1591401419
ask=1.12907
bid=1.129
last=0.0
volume=123

Remarks.

- This function can be used for live streaming of tick data.

10. Get actual bar information

ActualBar = MT.Get_actual_bar_info(instrument='EURUSD',
timeframe=MT.get_timeframe_value('H4'))

MT.get_timeframe_value('H4') = timeframe/period.

ActualBar = dictionary with the following information:

instrument = EURUSD
date = 1591315200
open = 1.13369
high = 1.13838
low = 1.12784
close = 1.129
volume = 98291

This function can be used for live streaming of actual bar data

11. Get last x ticks from now

LastTicks = MT.Get_last_x_ticks_from_now(instrument='EURUSD', nbrofticks=500)

'*EURUSD'* = instrument.
500 = number of ticks.
LastTicks = array with the following tick info(converted to data frame):

instrument=EURUSD
date=1591401419
ask=1.12907
bid=1.129
last=0.0
volume=123

12. Get last x bars from now

LastBars = MT.Get_last_x_bars_from_now(instrument='EURUSD',
timeframe=MT.get_timeframe_value('M1'), nbrofbars=1000)

LastBars = array with the following bar info:
Date, open, high, low, close and volume

13. Get a specific bar for list of instruments

Specific_bars = MT.Get_specific_bar(instrument_list = instrument_list,
specific_bar_index=1, timeframe = MT.get_timeframe_value('H1'))

Index = bar index, 0= actual bar, 1= last closed bar, etcetera

Specific_bars = Dictionary with for every instrument a dictionary with (d, o, h, l, c, v)

14. Open order

NewOrder = MT.Open_order(instrument='EURUSD', ordertype='buy', volume=0.01,
openprice=0.0, slippage=10, magicnumber=2000, stoploss=0.0, takeprofit=0.0,
comment='Test')

open pending order
NewOrder = MT.Open_order(instrument='EURUSD', ordertype='buy_stop', volume=0.04,
openprice=1.0870, slippage=10, magicnumber=2000, stoploss=1.0830, takeprofit=1.0950,
comment='Test')

'EURUSD' = instrument.
'buy' = order type ('buy', 'sell', 'buy_stop', 'sell_stop', 'buy_limit', 'sell_limit').
0.02 = volume/lot size.

0.0 = open price. For market orders price will be zero (0.0), for pending orders price must have an appropriate value.
10 = slippage.
1000 = magic number.
1.0830 = stop loss. The stop loss value is a market price (not in delta pips), of 0.0 then no stop loss set.
1.0950 = take profit. The take profit is a market price (not in delta pips), if 0.0 then no take profit set.
Test = comment. The comment may not contain the characters !#$, these are used internally order

NewOrder = ticket, if ticket has the value -1, placing of the order failed.

Remark:
If a ticket has the value -1, the following properties can be checked:

- *MT.order_return_message*. It is a string with the reason for fail.
- *MT.order_error*. It is an integer with MT4 error code.

15. Set SL and TP for position

ModifyPosition = MT.Set_sl_and_tp_for_position(ticket=53136604, stoploss=0.0, takeprofit=1.11001)

ModifyPosition = bool, *True* or *False*, *MT.order_return_message* and *MT.order_error* give more information

16. Set SL and TP for order (pendings)

ModifyOrder = MT.Set_sl_and_tp_for_order(ticket=53136804, stoploss=0.0, takeprofit=1.12001)

ModifyOrder = bool, *True* or *False*, *MT.order_return_message* and *MT.order_error* give more information

17. Get all (open)orders

AllOrders = MT.Get_all_orders()

AllOrders = data frame with the following info(only pending orders):
ticket, instrument, order_type, magic_number, volume, open_price, stop_loss, take_profit, comment;

18. Get all deleted orders within window

AllOrders = MT.Get_all_deleted_orders_within_window(
(date_from=datetime(2020, 6, 3, tzinfo=timezone), date_to=datetime.now()))

date_from = datetime(2020, 6, 3, tzinfo=timezone
date_to = datetime.now() + **"delta broker time and local time"**

AllOrders = data frame with the following info(only pending orders):
ticket, instrument, order_type, magic_number, volume, open_price, open_time, stop_loss, take_profit, delete_price, delete_time, comment;

19. Get all deleted orders

AllOrders = MT.Get_all_deleted_orders()

AllOrders = data frame with the following info(only pending orders):
ticket, instrument, order_type, magic_number, volume, open_price, open_time, stop_loss, take_profit, delete_price, delete_time, comment;

20. Get all (open) positions

AllPositions = MT.Get_all_open_positions()

AllPositions = data frame with the following info per position:
ticket, instrument, position_type, magic_number, volume, open_price, open_time, stop_loss, comment, take_profit, profit, swap, commission

21. Get all closed positions within window

timezone = pytz.timezone("Etc/UTC")

AllClosedPositions = MT.Get_closed_positions_within_window
(date_from=datetime(2020, 6, 3, tzinfo=timezone), date_to=datetime.now())

date_from = datetime(2020, 6, 3, tzinfo=timezone
date_to = datetime.now() + **"delta broker time and local time"**

AllClosedPositions = data frame with the following info:
position_ticket, instrument, order_ticket, position_type, magic_number, volume, open_price, open_time, stop_loss, take_profit, close_price, close_time, comment, profit, swap, commission

Be aware:

- The positions must be opened and closed within the window
- That there is probably a difference between local time and broker server time. Positions are in broker server time.

22. Get all closed positions

AllClosedPositions = MT.Get_all_closed_positions()

AllClosedPositions = data frame with the following infoper position:
position_ticket, instrument, order_ticket, position_type, magic_number, volume, open_price, open_time, close_price, close_time, comment, profit, swap, commission, open_log, close_log, dd_min, dd_plus

Be aware:

- The positions must be opened and closed within the window
- That there is probably a difference between local time and broker server time. Positions are in broker server time.

23. Close_position_by_ticket

ClosePosition = MT.Close_position_by_ticket(ticket=597318718)

ClosePosition = bool, *True* or *False*.

If ok = False, the properties *MT.order_return_message* and *MT.order_error* can be checked for the reason.

24. Close_position_partial_by_ticket

PartialClose = MT.Close_position_partial_by_ticket(ticket=367014000,
volume_to_close=0.01)

367014000= ticket. Ticket of position to close partly.
0.01 = volume to close

PartialClose = bool, *True* or *False*.

If ok = False, the properties MT5.order_return_message and MT5.order_error can be checked for the reason.

Remarks:

- If volume_to_close is smaller than minimum volume, the volume_to_close will be changed into minimum volume.
- After successful partial close the position ticket number for MT5 terminal will change

25. Delete order by ticket

DeleteOrder = MT.Delete_order_by_ticket(ticket=49988037)

DeleteOrder = bool, *True* or *False*.

If ok = False, the properties. *MT.order_return_message* and *MT.order_error* can be checked for the reason.

26. Get all instruments in broker watch list

Broker_marketwatch_list = MT.Get_instruments()

Broker_marketwatch_list = List will all instruments in the broker watch list

27. Get profit and loss over specified time period

timezone = pytz.timezone("Etc/UTC")

PnL = MT.Get_PnL(date_from=datetime(2020, 6, 3, tzinfo=timezone), date_to=datetime.now())

date_from = datetime(2020, 6, 3, tzinfo=timezone
date_to = datetime.now()

PnL = dictionary with the following info:

realized_profit=profit over all closed positions
unrealized_profit=profit over all open positions
buy_profit=profit over all closed buy positions
sell_profit=profit over all closed sell positions
positions_in_profit=number of profit positions
positions_in_loss=number of loss positions
volume_in_profit=total volume of profit positions
volume_in_loss=total volume of loss positions

28. Reset SL and TP for a position

Reset = MT.Reset_sl_and_tp_for_position(ticket=53136604)

Reset = bool, *True* or *False*, *MT.order_return_message* and *MT.order_error* give more information

29. Reset SL and TP for an order

Reset = MT.Reset_sl_and_tp_for_order(ticket=53136604)

Reset = bool, *True* or *False*, *MT.order_return_message* and *MT.order_error* give more information

30. Set bar date ascending or descending

In MT4/5 the actual bar has index 0([0]). So the BT will do the same as default. You can do the opposite by using this function.

Result = MT.Set_date_asc_or_desc(asc_desc=True)

asc_desc = True, row [0] is oldest bar
asc_desc = False, row [0] is actual bar

Result = bool, *True*, always

31. Check for license.

License = MT.Check_license()

License = bool, "*True*" or "*False*"

32. Check if trading is allowed

Trading_allowed = MT.Check_trading_allowed(instrument='EURUSD')

'*EURUSD*' = instrument to check, function is instrument specific

Trading_allowed = bool, "*True*" or "*False*"

Installation of EA on MT5 terminal

1. MT5

- The EA can run on same computer, local network or on a remote server. Up to you
- Move the EA into the ..\Experts folder
- Check if Indicator **Pytrader MT5.ex5** is available under the sub folder **Market**. Installation by MT5 market place. If not, the EA will work in demo full functions only limited for the following instruments; EURUSD, AUDCHF, NZDCHF, GBPNZD and USDCAD.
- Move the EA into an arbitrary chart.

- For switching from demo into licensed first remove the EA from chart and insert again.
- Set the proper socket/port number, python script must have same port number
- For licensed version fill in the path for the **Pytrader MT5.ex5**, like "**Market\ Pytrader MT5**,", the indicator is under the subfolder Market in the MT4 terminal
- Check if DLL's are allowed. The EA uses some standard windows DLL's for the socket communication
- Trading must be allowed

- In the right upper corner the EA must be green.

  , version number can be higher

- In the left upper corner you must see.

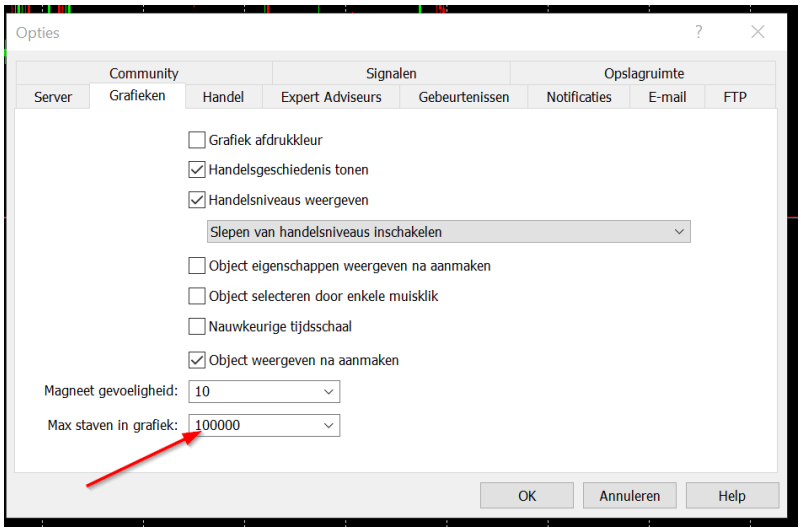
  , port number can be different, can be (demo) too

Remarks:

- This EA does not trade on its own. All commands have to come from your own coded strategy in a python script.

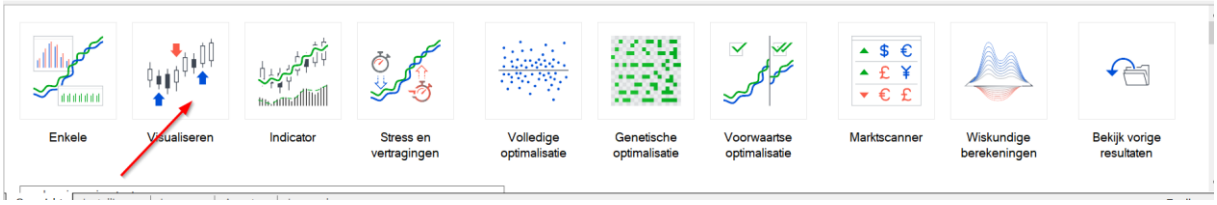
2. Historical data

- The amount of historical data you can retrieve depends on the history available on the MT5 terminal.
- This is also time frame and broker dependent.
- If many data are needed first set the max number of bars per chart to a higher value under tools, options, graphs

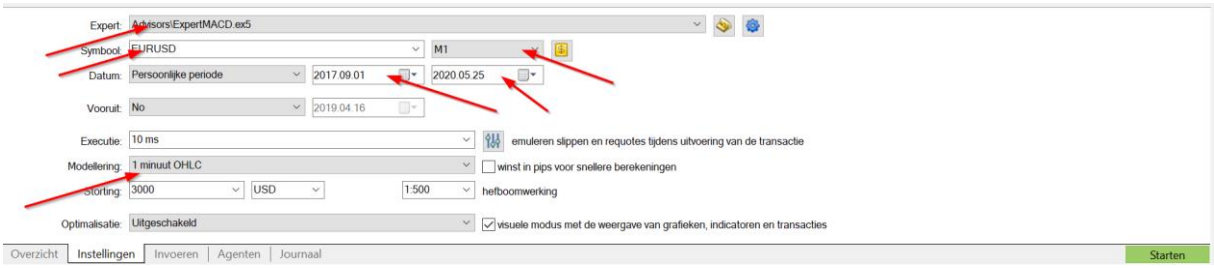


Next you can scroll back in a chart for the instrument you need the M1 bars for. There are also scripts on the internet for downloading historical data. Google is your friend.

A more elaborated way is to start the EA back tester; Cntrl+R. Select visual mode.



Next you will see this.



- Select a basic EA supplied by MT5
- Select the instrument

- Select time frame, in this example M1
- Select begin and end time
- Select bar OHLC, in this case M1
- Push the start button. Now the MT5 terminal will down load the Bars in the defined time period. The maximum to download is broker depending. F.i. with IC Markets you can download 1 million bars. Maybe even more.
- When the back testing starts you can abort.

3. Instrument lookup table.

Brokers use different names for instruments, especially indexes. To make it more general at connection time a lookup dictionary is passed as parameter. In here the python scripts find the translation between general instrument names and typical broker instrument names. This will make the application more general. A nice way is to do by a config file. In the config file you can define the lookups for different brokers. See below

[ICM]
AUDCAD: AUDCAD
AUDCHF: AUDCHF
AUDJPY: AUDJPY
AUDNZD: AUDNZD
AUDUSD: AUDUSD
BTCUSD: BTCUSD
CADCHF: CADCHF
CADJPY: CADJPY
CHFJPY: CHFJPY
CHFSGD: CHFSGD
EURAUD: EURAUD
[FXPIG]
AUDCAD: AUDCAD.spa
AUDCHF: AUDCHF.spa
AUDUSD: AUDUSD.spa
AUDNZD: AUDNZD.spa
AUDJPY: AUDJPY.spa

With the next code you can easy select the lookup table for a typical broker

The python script only recognizes the instruments defined in the lookup dictionary.

```
def config_instruments(config, section):
        dict1 = {}
        options = config.options(section)
        for option in options:
            try:
                option = option.upper()
                dict1[option] = config.get(section, option)
                if dict1[option] == -1:
                    print("skip: %s" % option)
            except:
                print("exception on %s!" % option)
```

```
                dict1[option] = None
        return dict1

#Read in config
CONFIG_FILE= "Instrument.conf"
config = configparser.ConfigParser()
config.read(CONFIG_FILE)

brokerInstrumentsLookup = config_instruments(config,'ICM')
```