

MT4 Pytrader_API.

Table of Contents

Introduction.....	3
Functions.	3
1. Instantiation.	3
2. Connect to server	3
3. Check connection.	4
4. Change time out value	4
5. Retrieve broker server time.	4
6. Get static account information.	5
7. Get dynamic account information.....	5
8. Get instrument information	5
9. Get last tick information.....	6
10. Get actual bar information	6
11. Get last x ticks from now	7
12. Get last x bars from now	7
13. Open order	7
14. Set SL and TP for position	8
15. Set SL and TP for order (pendings)	8
16. Get all orders	9
17. Get all (open) positions	9
18. Get all closed positions.....	9
19. Close_position_by_ticket	10
20. Close_position_partial_by_ticket.....	10
21. Delete order by ticket.....	11
22. Get all instruments in broker watch list	11
23. Get a specific bar for list of instruments	11
Installation of EA on MT4 terminal	12
1. MT4.....	12
Historical data.....	13
1. MT4.....	13
Instrument lookup table.....	13

Changes

Date	Version	Changes
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 MT4 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 MT4 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 MT4 terminal.
- A python script, name “Pytrader_API”, which functions as the connection with the MT4 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.

```
from utils.Pytrader_API_V1_03 import Pytrader_API
## instantiate
MT = Pytrader_API()
```

Utils is a subfolder in the project.

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.

```
## 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
## of course MT4 EA must use same port
Connected = MT.Connect(server='127.0.0.1', port=11111,
                       instrument_lookup=brokerInstrumentsLookup)
# or
Connected = MT.Connect(server='192.168.0.103', port=22222,
                       instrument_lookup=brokerInstrumentsLookup)
```

‘127.0.0.1’ = server. In this case is local host.

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

11111 = port (number). Server socket of the MT4 EA must use same port.

brokerInstrumentLookup = dictionary

Connection is always with a MT4 terminal and a broker account. Brokers often use different names for their instruments/symbols. Another way is to use config files. Example at the end of this document.

Connected = bool will be True or False.

If connection is made the MT.connected property will be set to True. There is a timeout of 60 seconds. If no connection MT.connected property will be set to False.

3. Check connection.

```
## check connection
CheckAlive= MT.Check_connection()
```

CheckAlive= bool, which will be True or False.

4. Change time out value

```
## Change the time out
MT.Set_timeout(timeout_in_seconds=120)
```

120 = time out value in seconds

5. Retrieve broker server time.

```
## retrieve broker server time
ServerTime = MT.Get_broker_server_time()
```

ServerTime = broker server time.

6. Get static account information.

```
## 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

```
## get dynamic account information
DynamicInfo = MT.Get_dynamic_account_info()
```

DynamicInfo = dictionary with the following information:

```
balance=3436.16
equity=3413.56
profit=-22.6
margin=40.6
margin_level=8106.05
margin_free=3101.64
```

8. Get instrument information

```
## 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

```
## get last tick information
LastTick = MT.Get_last_tick_info(instrument='EURUSD')
```

'EURUSD' = instrument.

LastTick = dictionary with the following information:

```
instrument=EURUSD
date=1591401419
ask=1.12907
bid=1.129
last=0.0
volume=0
```

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

10. Get actual bar information

```
## get actual bar information
ActualBar = MT.Get_actual_bar_info(instrument='EURUSD',
                                   timeframe=MT.get_timeframe_value('H4'))
```

'EURUSD' = instrument.

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

```
## get last x ticks from now
## if MT terminal does not have this as history it can take some time
##     MT terminal needs first to retrieve from broker
##     the max amount of ticks is broker dependent
##     socket time out is set to 60 seconds
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):

	date	ask	bid	last	volume
0	1591401298	1.12882	1.12879	0.0	0
1	1591401298	1.12881	1.12879	0.0	0
2	1591401299	1.12882	1.12879	0.0	0
3	1591401299	1.12881	1.12879	0.0	0
4	1591401299	1.12882	1.12879	0.0	0

12. Get last x bars from now

```
## get last x bars from MT terminal
## if MT terminal does not have this as history it can take some time
##     MT terminal needs first to retrieve from broker
##     the max amount of bars is broker depending
##     socket time out is set to 60 seconds
LastBars = MT.Get_last_x_bars_from_now(instrument='EURUSD',
                                       timeframe=MT.get_timeframe_value('M1'), nbrofbars=1000)
'EURUSD' = instrument.
```

MT.get_timeframe_value ('M1') = timeframe/period.

1000 = number of bars to retrieve.

LastBars = array with the following bar info(converted to data frame):

	date	open	high	low	close	volume
0	2020-06-05 07:17:00	1.13396	1.13400	1.13396	1.13397	12
1	2020-06-05 07:18:00	1.13397	1.13398	1.13393	1.13396	40
2	2020-06-05 07:19:00	1.13396	1.13405	1.13393	1.13394	32
3	2020-06-05 07:20:00	1.13394	1.13411	1.13392	1.13411	66
4	2020-06-05 07:21:00	1.13411	1.13420	1.13411	1.13418	24

13. Open order

```
## 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' = ordertype ('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 = magicnumber.

1.0830 = stoploss. The stop loss value is a market price (not in delta pips), of 0.0 then no stop loss set.

1.0950 = takeprofit. 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

NewOrder = ticket, if ticket has the value -1, 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.

14. Set SL and TP for position

```
## set stoploss and takeprofit for position
```

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

53136604 = ticker for position to change settings

0.0 = stop loss value. If 0.0 then SL will not be changed

1.11001 = new take profit value.

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

15. Set SL and TP for order (pendings)

```
## set stoploss and takeprofit for order (pendings)
```

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

53136804 = ticker for order to change settings

0.0 = stop loss value. If 0.0 then SL will not be changed

1.12001 = new take profit value.

`ChangeOrder` = bool, `True` or `False`, `MT.order_return_message` and `MT.order_error` give more information

16. Get all orders

```
## get all orders(pendings)
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;

	ticket	instrument	order_type	...	stop_loss	take_profit	comment
0	54192423	EURCHF	buy_limit	...	1.07	1.09	Test comment
1	54191631	USDSEK	buy_stop	...	9.30	9.35	
2	54191423	CHFSGD	sell_limit	...	1.47	1.43	

17. Get all (open) positions

```
## get all open positions
AllPositions = MT.Get_all_open_positions()
```

`AllPositions` = data frame with the following info:

ticket, instrument, position_type, magic_number, volume, open_price, open_time, stop_loss, comment, take_profit, profit, swap, commission;

	ticket	instrument	position_type	...	comment	profit	swap
0	54096625	EURUSD	buy	...	H2 wave 4 ST	-5.52	-0.23
1	54095945	USDSEK	sell	...	H2 Wave 4 ST	-13.95	-0.09
2	53939125	AUDCAD	buy	...	H4 wave 4 IT	-8.40	-0.12
3	53782856	EURAUD	sell	...	H2 wave 4 LT	23.16	-0.12
4	53748502	GBPAUD	sell	...	H2 wave 4 IT	-16.89	-0.44

18. Get all closed positions

```
## get all closed position in a specified period
timezone = pytz.timezone("Etc/UTC")
AllClosedPositions = MT.Get_all_closed_positions(date_from=datetime(2020, 6, 3,
                                                                    tzinfo=timezone), date_to=datetime.now())
```

`date_from` = datetime(2020, 6, 3, tzinfo=timezone)

`date_to` = datetime.now()

`AllClosedPositions` = data frame with the following info:

position_ticket, instrument, order_ticket, position_type, magic_number, volume, open_price, open_time, close_price, close_time, comment, profit, swap, commission

	position_ticket	instrument	order_ticket	...	profit	swap	commission
0	52276947	GBPAUD	53493455	...	-76.40	-0.91	-0.22
1	53024510	GBPNZD	53493462	...	96.19	-0.48	-0.42
2	53521115	GBPNZD	53622957	...	6.03	0.00	-0.42
3	53682283	GOLD	53682381	...	-1.08	0.00	-0.42
4	53782204	AUDCAD	53782212	...	-0.22	0.00	-0.42
5	53569405	EURSGD	53784182	...	12.45	-0.30	-0.42
6	53623751	CHFJPY	53877649	...	57.52	-0.61	-0.42
7	53782247	AUDCAD	54048783	...	36.67	-0.11	-0.42
8	53796568	EURCHF	54068367	...	79.04	-0.08	-0.42

Be aware that for MT4 terminal the result of closed positions is based on your terminal settings.

19. Close_position_by_ticket

```
## close position by ticket
ClosePosition = MT.Close_position_by_ticket(ticket=597318718)
```

597318718= ticket. Ticket of position to close.

ClosePosition = bool, True or False.

If ok = False, the properties MT4.order_return_message and MT4.order_error can be checked for the reason.

20. Close_position_partial_by_ticket

```
## close position partly
PartialClose = MT.Close_position_partial_by_ticket(ticket=367014000,
                                                    volume_to_close=0.01)
if (PartialClose == False):
    print(MT.order_return_message)
```

367014000= ticket. Ticket of position to close partly.

0.01 = volume to close

PartialClose = bool, True or False.

If ok = False, the properties MT4.order_return_message and MT4.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 MT4 terminal will change

21. Delete order by ticket

```
## delete order by ticket(pending)
DeleteOrder = MT.Delete_order_by_ticket(ticket=49988037)
```

49988037= ticket. Ticket of order to delete(pendings).

DeleteOrder = bool, True or False.

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

22. Get all instruments in broker watch list

```
## List of all instruments in brokers market watch
# Broker_marketwatch_list = list[]
Broker_marketwatch_list = MT.Get_instruments()
```

Broker_marketwatch_list = List will all instruments in the broker watch list

23. Get a specific bar for list of instruments

```
# Get a specific bar (d, o, h, l, c, v) by index for a list of instruments
# SpecificBars = dict{dict{}}
SpecificBars = MT.Get_specific_bar(instrument_list = instrument_list,
                                   specific_bar_index=1, timeframe = MT.get_timeframe_value('H1'))
```

Instrument_list = List with instruments

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

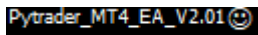
timeframe = bar interval

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

Installation of EA on MT4 terminal

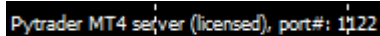
1. MT4

- 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_MT4.ex5 is available. 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 Pytrade_MT4.ex5, like "Market\", if 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.

Historical data

1. MT4

- The amount of historical data to retrieve depends on the history available on the MT4
- 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 a basic EA supplied by MT4, in principle any EA is OK
- Select the instrument
- Select time frame, in this example M1
- Select begin and end time
- Push the start button. Now the MT4 terminal will download 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.

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')
```