9. OCO Orders

This section is about **OCO ("One Cancels Other") orders**.

In [1]:

```
import fxcmpy
import pandas as pd
import datetime as dt
con = fxcmpy.fxcmpy(config_file='fxcm.cfg')
```

9.1. Creating an OCO Order

To begin with, check **the open positons**.

In [2]:

```
con.get_orders()
```

Out[2]:

OCO orders are created by the method `con.create_oco_order()`. A detailed description of the method's parameters is found in section *API Documentation*.

In [3]:

```
oco_order = con.create_oco_order(account_id='2815291',
                                 symbol='EUR/USD',
                                 is_buy=True, is_buy2=False,
                                 amount=30, is_in_pips= False,
                                 time_in_force='GTC',
                                 at_market=1,
                                 order_type='MarketRange',
                                 expiration='01122017',
                                 limit=1.13, limit2=1.12,
                                 rate=1.11, rate2=1.13,
                                 stop=1.1, stop2=1.15,
                                 trailing_step=0, trailing_step2=0,
                                 trailing_stop_step=0,
                                 trailing_stop_step2=0)
```

The `con.oco_orders` attribute is a `dict` object, containing all existing OCO orders as values and their `ids` as keys.

In [4]:

```
con.oco_orders
```

Out[4]:

```
{404812350: <fxcmpy.fxcmpy_oco_order.fxcmpy_oco_order at 0x7fac5c24d1d0>}
```

`con.get_oco_order_ids()` returns the OCO order `ids`.

In [5]:

```
con.get_oco_order_ids()
```

Out[5]:

```
[404812350]
```

After the OCO order placement, two orders a active. The `ocoBulkId` value connects the two orders to the OCO order.

In [6]:

```
con.get_orders().T
```

Out[6]:

	0	1
accountId	2815291	2815291
accountName	02815291	02815291
amountK	30	30
buy	1.11	0
currency	EUR/USD	EUR/USD
currencyPoint	3	3
expireDate		
isBuy	True	False
isELSOrder	False	False
isEntryOrder	True	True
isLimitOrder	True	False
isNetQuantity	False	False
isStopOrder	False	True
limit	1.13	1.12
limitPegBaseType	-1	-1
limitRate	1.13	1.12
ocoBulkId	404812350	404812350
orderId	404812349	404812351
range	0	0
ratePrecision	5	5
sell	0	1.13
status	1	1
stop	1.1	1.15
stopMove	0	0
stopPegBaseType	-1	-1
stopRate	1.1	1.15
t	3	3
time	06072018135952931	06072018135952932
timeInForce	GTC	GTC
tradeId	178168188	178168189
type	LE	SE

9.2. Changing OCO Orders

To add an existing order to an OCO order, the method `con.add_to_oco()` is used.

First, an entry **order is created**.

In [7]:

```
order = con.create_entry_order(symbol='USD/JPY', is_buy=True, rate = 110,
                               amount=50, is_in_pips = False,
                               time_in_force='GTC', stop=None,
                               limit=112, trailing_step=None)
```

The new order's `id` is:

In [8]:

```
order_id = order.get_orderId()
order_id
```

Out[8]:

```
404812374
```

The order's `ocoBulkId` should be 0:

In [9]:

```
order.get_ocoBulkId()
```

Out[9]:

```
0
```

Second, the **order is added** to the OCO order via the `ocoBulkId` value.

In [10]:

```
bulk_id = con.get_oco_order_ids()[0]
```

In [11]:

```
con.add_to_oco(oco_bulk_id=bulk_id, order_ids=[order_id])
```

Now the `ocoBulkId` value is the same as for the OCO order from above.

In [12]:

```
order.get_ocoBulkId()
```

Out[12]:

```
404812350
```

Removing an order is accomplished via the `con.remove_from_oco()` method.

In [13]:

```
con.remove_from_oco(order_ids=[order_id])
```

In [14]:

```
order.get_ocoBulkId()
```

Out[14]:

```
0
```

Once can also **add and remove orders** in one step, just use `edit_oco_order()`

In [15]:

```
order_id_2 = con.get_order_ids()[0]
```

In [16]:

```
con.edit_oco(oco_bulk_id=bulk_id,
             remove_order_ids=[order_id_2],
             add_order_ids=[order_id])
```

In [17]:

```
con.get_orders().T
```

Out[17]:

	0	1	2
accountId	2815291	2815291	2815291
accountName	02815291	02815291	02815291
amountK	30	30	50
buy	1.11	0	110
currency	EUR/USD	EUR/USD	USD/JPY
currencyPoint	3	3	4.54496
expireDate			
isBuy	True	False	True
isELSOrder	False	False	False
isEntryOrder	True	True	True
isLimitOrder	True	False	True
isNetQuantity	False	False	False
isStopOrder	False	True	False
limit	1.13	1.12	112

	0	1	2
limitPegBaseType	-1	-1	-1
limitRate	1.13	1.12	112
ocoBulkId	0	404812350	404812350
orderId	404812349	404812351	404812374
range	0	0	0
ratePrecision	5	5	3
sell	0	1.13	0
status	1	1	1
stop	1.1	1.15	0
stopMove	0	0	0
stopPegBaseType	-1	-1	-1
stopRate	1.1	1.15	0
t	3	3	3
time	06072018140139676	06072018135952932	06072018140139677
timeInForce	GTC	GTC	GTC
tradeId	178168188	178168189	178168201
type	LE	SE	LE

9.3. The `fxcmpy_oco_order` Class

The methods `con.get_oco_order()` and `con.create_oco_order()` both return an instance of the `fxcmpy_oco_order` class.

In [18]:

```
oco_order = con.get_oco_order(bulk_id)
```

In [19]:

```
oco_order
```

Out[19]:

```
<fxcmpy.fxcmpy_oco_order.fxcmpy_oco_order at 0x7fac5c24d1d0>
```

This class provides methods to get the instance's `ocoBulkId` ...

In [20]:

```
oco_order.get_ocoBulkId()
```

Out[20]:

```
404812350
```

... the `ids` of the contained orders ...

In [21]:

```
oco_order.get_order_ids()
```

Out[21]:

```
[404812351, 404812374]
```

... and the order objects itself.

In [22]:

```
oco_order.get_orders()
```

Out[22]:

```
[<fxcmpy.fxcmpy_order.fxcmpy_order at 0x7fac4839d9e8>,
 <fxcmpy.fxcmpy_order.fxcmpy_order at 0x7fac48391a20>]
```

The class has also methods to **add or remove existing orders**.

In [23]:

```
new_order = con.create_entry_order(symbol='GBP/USD', is_buy=True,
                                   amount=300, rate= 1.36,
                                   limit=1.37, is_in_pips = False,
                                   time_in_force='GTC')
```

In [24]:

```
oco_order.add_order([new_order])
```

In [25]:

```
oco_order.get_orders()
```

Out[25]:

```
[<fxcmpy.fxcmpy_order.fxcmpy_order at 0x7fac4839d9e8>,
 <fxcmpy.fxcmpy_order.fxcmpy_order at 0x7fac48391a20>,
 <fxcmpy.fxcmpy_order.fxcmpy_order at 0x7fac4833d9e8>]
```

In [26]:

```
oco_order.remove_order([new_order])
```

It is also possible to add orders to or remove from the OCO order **in a single step**.

In [27]:

```
order = oco_order.get_orders()[0]
```

In [28]:

```
oco_order.edit_order(add_orders=[new_order], remove_orders=[order])
```

In [29]:

```
oco_order.get_order_ids()
```

Out[29]:

```
[404812374, 404812521]
```

In [30]:

```
con.close()
```