async_v20 Documentation

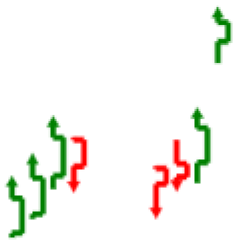
Release 8.0.0b1

James Peter Schinner

Nov 04, 2020

Contents

A foreign exchange client



CHAPTER 1

Disclaimer

- **Losses can exceed investment.**
- async_v20 and its creator has no affiliation with OANDA. And is not endorsed by OANDA in any manner.
- async_v20 is in Beta stage and has not been tested on a live OANDA account
- **Use at own risk**

CHAPTER 2

Features

- Exposes the entire v20 API ‘
- immutable objects
- No **args, **kwargs* In client methods. So no guessing what arguments a method takes
- Serialize objects directly into *pandas* **Series** or **DataFrame** objects
- Construct *concurrent* trading algorithms

CHAPTER 3

installation

```
$ pip install async_v20
```

async_v20 is built with aiohttp. It is therefore recommended to also install *cchardet* and *aiodns* as per *aiohttp* documentation '

```
$ pip install cchardet

$ pip install aiodns
```

CHAPTER 4

Why async_v20?

There are many OANDA clients for python already available so why create another? The main driver for creating async_v20 was to facilitate better risk management, by allowing user's to concurrently monitor account status and trade currency's.

An unintended consequence of async_v20 is the ability to create clear segregation between implementation ideas.

A simple example might contain a coroutine for the following:

- Monitoring overall account status
- Watching price stream and triggering buy/sell signals
- Monitoring individual trades and closing movements against held positions

A synchronous implementation would require considerable effort to determine which task communicates with the server next. async_v20 removes this burden by using aiohttp

Further goals of async_v20 has been to lower the barrier of entry for algorithmic trading by providing a complete and simple to use interface.

CHAPTER 5

Tutorial

Using async_v20

CHAPTER 6

Source code

Can be found on GitHub

Please feel free to file an issue on the bug tracker if you have found a bug or have some suggestion in order to improve the client.

CHAPTER 7

Dependencies

- **python >= 3.6**
- aiohttp >= 2.2.5
- ujson >= 1.35'
- yarl >= 0.12.0'
- pandas

CHAPTER 8

Contents

8.1 Getting started

8.1.1 Creating an Account

To use *async_v20* you must have an account with *OANDA*

- Create an account HERE
- Create an API *token* ALSO HERE

8.1.2 Setting up environment

- Install *async_v20* as per *installation*
- Create a new *environment variable* with the name *OANDA_TOKEN* and value as the *token* generated from *Creating an Account*.

Adding Environment Variables

Due to the abundance of information in regards to configuring environment variables, it can be challenging to find information that will result in success.

Here we will try and and point you in the correct direction.

PyCharm

https://stackoverflow.com/questions/42708389/how-to-set-environment-variables-in-pycharm

Mac

https://stackoverflow.com/questions/135688/setting-environment-variables-in-os-x/3756686#3756686

Windows

https://stackoverflow.com/questions/1672281/environment-variables-for-java-installation

Ubuntu

https://askubuntu.com/questions/58814/how-do-i-add-environment-variables

What is an environment variable?

For our requirements, we will think of an environment variable, as a variable stored outside the scope of the application.

Python programs can access these variables via the *os* module

Why store the token in an environment variable?

There are a couple of reason. Primarily:

- Convenience, create a client with *OandaClient()* not *OandaClient(xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)*
- Secure, prevents you from uploading the token to an online repository
- May simplify deployment of your trading program

Note:

- It is considered best practice use a *virtual environment*
- It is not required to store the token in an *environment variable*. The token can be passed to *OandaClient*

8.1.3 Using async_v20

Once an account has been created as per *Creating an Account* and the environment is configured as per *Setting up environment*, we are ready to begin.

Lets first take a look at this code example, then go though it line by line.

```
import asyncio

from async_v20 import OandaClient


async def get_account():
```

(continues on next page)

(continued from previous page)

```
    async with OandaClient() as client:
        return await client.account()


loop = asyncio.get_event_loop()
account = loop.run_until_complete(get_account())

# pandas Series
print(account.series())
```

First we need to import *asyncio* this allows us to run our *coroutine*

```
import asyncio
```

We then import *OandaClient* which provides us the means to interact with OANDA

```
from async_v20 import OandaClient
```

Because *OandaClient* returns *coroutines* we use *async def*. This allows the use of the *await* syntax

```
async def get_account():
```

OandaClient is a *context manager*, we use *async with* to instantiate a client instance. Doing so will automatically close the *http session* when we're done

```
    async with OandaClient() as client:
```

We then create and *await* the *coroutine* by calling *client*. **account()**

```
        return await client.account()
```

Now we have defined our *coroutine* we need to execute it. To do so we need an event loop. Achieved using *asyncio*. **get_event_loop()**

```
loop = asyncio.get_event_loop()
```

The value returned by executing the *account coroutine* is accessed through the event loop.

```
account = loop.run_until_complete(get_account())
```

async_v20 objects have a *Model*. **series()** method that returns a *pandas*. **Series**

```
print(account.series())
```

Note: Each application should only instantiate **ONE** OandaClient instance per account. See *Best Practices*.

8.2 Best Practices

8.2.1 One OandaClient per application

Using async_v20 example, used *OandaClient* as a context manager. This is correct when the client does not need to be shared between multiple *coroutines*.

This is a an example of multiple coroutines using the **One** OandaClient instance:

```
import asyncio

from async_v20 import OandaClient

client = OandaClient()


async def poll_account(poll_interval=6, client=client):
    while True:
        account = await client.account()
        print(account)
        await asyncio.sleep(poll_interval)


async def stream(instruments, client=client):
    async for price in await client.stream_pricing(instruments):
        print(price)


loop = asyncio.get_event_loop()
loop.run_until_complete(
    asyncio.gather(poll_account(), stream('EUR_USD'))
    )
client.close()
```

8.2.2 Initialize first

OandaClient requires initialization. The initialization procedure can delay execution of *OandaClient*. **methods**

If this is a concern for you, it is recommended to preemptively initialize the OandaClient instance.

```
import asyncio

from async_v20 import OandaClient

client = OandaClient()

loop = asyncio.get_event_loop()
response = loop.run_until_complete(client.initialize())

# Write your code here

loop.run_until_complete(client.close())
```

8.2.3 Check Services

It is encouraged to check the service you wish to consume is available. See *Health API*.

8.3 Beyond Getting Started

So you have read *Using async_v20* and need to know more.

Your first issue is knowing what methods to use.

The *OandaClient API* docs contains all the exposed methods *async_v20* provides.

8.3.1 What you need to know

- ***OandaClient* returns v20 API calls in `Response` objects** The *response* is a python dictionary and designed to reflect the responses defined by the v20 docs :
  - responses contain the equivalent python objects. As defined in *Class Definitions*
- *OandaClient*. Automatically supplys arguments to endpoints that require the following:

  class `async_v20.definitions.primitives.`**`AccountID`**
  Bases: `str`, `async_v20.definitions.primitives.Primitive`

  The string representation of an Account Identifier.

  class `async_v20.endpoints.annotations.`**`Authorization`**
  Bases: `str`

  Contains OANDA's v20 API authorization token

  class `async_v20.endpoints.annotations.`**`LastTransactionID`**
  Bases: `async_v20.definitions.primitives.TransactionID`

  Contains the most recent TransactionID

  class `async_v20.endpoints.annotations.`**`SinceTransactionID`**
  Bases: `async_v20.definitions.primitives.TransactionID`

  The account changes to get Since LastTransactionID for account_changes() method
- **OandaClient by default will connect to the practice server:**
  - OANDA's docs Contain host information

8.3.2 Underling Principles

All arguments passed to *OandaClient API* methods are used to create instances of the parameters annotation. Why is this useful?

- Prevents you from importing required class' and instantiating them manually
- HTTP requests are formatted based upon the objects the endpoint accepts. See *How Are Arguments Passed*
- The base class `Model` will convert the object into valid *JSON*
- Invalid arguments will raise *`InvalidValue`* catching mistakes earlier
- Provides flexibility when passing arguments

Here is an Example

```python
import asyncio

from async_v20 import OandaClient

client = OandaClient()

# This
coroutine_1 = client.create_order('AUD_USD', 10)

# Is the same as this
from async_v20 import InstrumentName, DecimalNumber

coroutine_2 = client.create_order(
    InstrumentName('AUD_USD'), DecimalNumber(10)
)

# Is the same as this
from async_v20 import OrderRequest

coroutine_3 = client.post_order(
    order_request=OrderRequest(
        instrument='AUD_USD', units=10, type='MARKET'
    )
)

loop = asyncio.get_event_loop()
loop.run_until_complete(
    asyncio.gather(
        coroutine_1,
        coroutine_2,
        coroutine_3
    )
)
```

Note: Executing this example **will** create a long position of the AUD/USD currency pair worth 30 units.

8.3.3 What might be useful

How Are Arguments Passed

All methods exposed by *async_v20*. **OandaClient** are written in a declarative fashion.

Lets take at look at an example:

```python

class InstrumentInterface(object):
    @endpoint(GETInstrumentsCandles)
    def get_candles(self,
                    instrument: InstrumentName,
                    price: PriceComponent = 'M',
                    granularity: CandlestickGranularity = 'S5',
                    count: Count = sentinel,
                    from_time: FromTime = sentinel,
                    to_time: ToTime = sentinel,
```

(continues on next page)

(continued from previous page)

```
                        smooth: Smooth = False,
                        include_first_query: IncludeFirstQuery = sentinel,
                        daily_alignment: DailyAlignment = 17,
                        alignment_timezone: AlignmentTimezone = 'America/New_York',
```

Note:

The *docstring* has been left of this example for brevity.

- This example is not complete without a *pass* statement

First

- We define the endpoint:

```
class InstrumentInterface(object):
```

Then

- We define arguments to pass to the endpoint

```
class InstrumentInterface(object):
    @endpoint(GETInstrumentsCandles)
    def get_candles(self,
                    instrument: InstrumentName,
                    price: PriceComponent = 'M',
                    granularity: CandlestickGranularity = 'S5',
                    count: Count = sentinel,
                    from_time: FromTime = sentinel,
                    to_time: ToTime = sentinel,
                    smooth: Smooth = False,
                    include_first_query: IncludeFirstQuery = sentinel,
                    daily_alignment: DailyAlignment = 17,
                    alignment_timezone: AlignmentTimezone = 'America/New_
↪York',
```

You will notice that all *arguments* have an *annotation*.

async_v20 uses these annotations to format arguments into the correct http request.

The http request formatting is defined by the *EndPoint*

In this case

```
class GETInstrumentsCandles(EndPoint):
    # the HTTP verb to use for this endpoint
    method = 'GET'

    # path to endpoint
    path = ('/v3/instruments/', InstrumentName, '/candles')

    # description of endpoint
    description = 'Fetch candlestick data for an instrument.'

    # parameters required to send to endpoint
    parameters = {Authorization: (HEADER, 'Authorization'),␣
↪AcceptDatetimeFormat: (HEADER, 'Accept-Datetime-Format'),
```

(continues on next page)

(continued from previous page)

```
                    InstrumentName: (PATH, 'instrument'), PriceComponent:␣
↪(QUERY, 'price'),
                    CandlestickGranularity: (QUERY, 'granularity'), Count:␣
↪(QUERY, 'count'), FromTime: (QUERY, 'from'),
                    ToTime: (QUERY, 'to'), Smooth: (QUERY, 'smooth'),␣
↪IncludeFirstQuery: (QUERY, 'includeFirst'),
                    DailyAlignment: (QUERY, 'dailyAlignment'),␣
↪AlignmentTimezone: (QUERY, 'alignmentTimezone'),
                    WeeklyAlignment: (QUERY, 'weeklyAlignment')}

    # valid responses
    responses = {
        200: {'instrument': InstrumentName, 'granularity':␣
↪CandlestickGranularity, 'candles': ArrayCandlestick}}

    # error msgs'
    error = (400, 401, 404, 405)
```

Notice that

```
    parameters = {Authorization: (HEADER, 'Authorization'),␣
↪AcceptDatetimeFormat: (HEADER, 'Accept-Datetime-Format'),
```

Contains *key* entries that coincide with the methods annotations. The annotation is then used to lookup up location of the argument in the HTTP request and the corresponding key that will be used with the passed data to create the correct key/value pair.

How Responses are Constructed

The *http* response from *OANDA* is handled by the *EndPoint* each *OandaClient*. **method** defines.

There is a two step process by which the response is constructed:

- The http *status* is used to look up the expected response
- Each JSON object is constructed into the appropriate python object

How Objects are Serialized

One of the main problems *async_v20* solves for you is correctly formatting *objects* into JSON that OANDA's v20 API accepts.

The issue here is that OANDA defines objects with *camelCase attributes*. Python programs typically reserve camel-Case for *class* definitions.

This means, in order to both satisfy python standards and OANDA, objects (as defined in *Class Definitions*) need to accept *camelCase* and *snake_case* arguments when being constructed.

Objects store there attributes as *snake_case* (as python programmers would expect), which adds a further requirement to convert these into *camelCase* when being serialized.

To solve this the `async_v20.definitions.attributes` module contains two dictionaries. `instance_attributes` & `json_attributes`:

- When creating objects, arguments are passed through *instance_attributes* dictionary
- When serializing objects, attributes are passed through *json_attributes* dictionary

8.4 Formatting Order Requests

OandaClient provides the option to format `OrderRequest`'s in the context of the instrument the OrderRequest is for.

Confused? So was I, Let's have a look at an *instrument*. **series()** representation of an `Instrument`.

```
display_name                           Brent Crude Oil
display_precision                                    3
margin_rate                                       0.05
maximum_order_units                             100000
maximum_position_size                                0
maximum_trailing_stop_distance                     100
minimum_trade_size                                   1
minimum_trailing_stop_distance                    0.05
name                                           BCO_USD
pip_location                                        -2
trade_units_precision                                0
type                                               CFD
dtype: object
```

The above instrument outlines the formatting of an *OrderRequest* for the *Instrument* (in this case *Brent Crude Oil*). Take *Instrument*. **trade_units_precision** as an example, this attribute defines how many decimal places `OrderRequest`. **units** may be used for this instrument (our example *0*).

You will also notice **minimum** and **maximum** values for other *OrderRequest* attributes.

OandaClient. **format_order_requests** is a boolean value which changes the degree to which *OrderRequest*'s will be modified to comply with the instrument specification. If **format_order_requests** is set to `True` *OandaClient* will modify values so that they are within the valid range specified by the instrument.

Note: I believe most users will want to use this feature as it dramatically reduces the complexity of placing valid OrderRequests. It is disabled by default.

Only enable this feature if you understand that your *OrderRequest*. **stop_loss_on_fill/ take_profit_on_fill/trailing_stop_loss_distance** and **price_bound** configuration may be altered to comply with the instruments valid ranges.

Note: The precision of `DecimalNumber` and `PriceValue` will **always** be rounded to the correct precision for the instrument, regardless of *OandaClient*. **format_order_requests** value.

Example:

```
>>> from async_v20 import OandaClient
>>> import asyncio
>>> client = OandaClient()
>>> run = loop.run_until_complete
>>> run(client.create_order('AUD_USD', 0))
Traceback (most recent call last):
ValueError: OrderRequest units 0.0 are less than the minimum trade size 1.0
>>> run(client.create_order('AUD_USD', 1))
<Status [201]: orderCreateTransaction, orderFillTransaction, relatedTransactionIDs,␣
↪lastTransactionID>
>>> client.format_order_requests
```

(continues on next page)

(continued from previous page)

```
False
>>> client.format_order_requests = True
>>> run(client.create_order('AUD_USD', 0))
<Status [201]: orderCreateTransaction, orderFillTransaction, relatedTransactionIDs,␣
↪lastTransactionID>
>>> client.format_order_requests = False
>>> run(client.create_order('AUD_USD', 1, trailing_stop_loss_on_fill=0))
Traceback (most recent call last):
ValueError: Trailing stop loss distance 0.0 is not within AUD_USD specified range 0.
↪0005 - 1.0
>>> client.format_order_requests = True
>>> run(client.create_order('AUD_USD', 1, trailing_stop_loss_on_fill=0))
<Status [201]: orderCreateTransaction, orderFillTransaction, relatedTransactionIDs,␣
↪lastTransactionID>
```

8.5 Dealing With Time

DateTimes in async_20 have a requirement to support two time formats:

- **RFC3339** - *'2017-08-11T15:04:31.639182000Z'*
- **UNIX** - *'1502463871.639182000'*

Note: These are the two valid arguments that may be supplied to `OandaClient`.

`DateTime` is responsible for handling datetimes in async_v20. calling **DateTime** with either a *RFC3339*, *UNIX*, time.time(), or datetime.datetime() representation of a datetime creates a `pandas.Timestamp`.

async_v20 adds an additional helper method *Timestamp*. **json()** for the purpose of serializing to the correct JSON format OANDA expects.

Note: This method is not part of the public API, it is documented here to give you an understanding of how async_v20 parses datetime like arguments into JSON that meets OANDA's specification

Example

```
>>> from async_v20 import OandaClient
>>> import asyncio
>>> from time import time
>>> from datetime import datetime
>>> loop = asyncio.get_event_loop()
>>> run = loop.run_until_complete
>>> client = OandaClient()
>>> rsp = run(client.get_candles(
...     'AUD_JPY',
...     granularity='M1', # 1 minute candles
...     from_time=time() - (10 * 60),   # 10 minutes ago
...     to_time=datetime.utcnow()   # Current time
... ))
>>> rsp
<Status [200]: instrument, granularity, candles>
```

(continues on next page)

(continued from previous page)

```
>>> len(rsp.candles)  # I was aiming for 10
11
```

Serializing

```
>>> from async_v20 import DateTime
>>> unix_example = '1502463871.639182000'
>>> rfc3339_example = '2017-08-11T15:04:31.639182000Z'
>>> dt = DateTime(unix_example)
>>> dt
Timestamp('2017-08-11 15:04:31.639182+0000', tz='UTC')
>>> dt.json('RFC3339')
'2017-08-11T15:04:31.639182000Z'
>>> dt.json('UNIX')
'1502463871.639182000'
>>> dt.json('UNIX') == unix_example
True
>>> dt = DateTime(rfc3339_example)
>>> dt.json('UNIX')
'1502463871.639182000'
>>> dt.json('UNIX') == unix_example
True
```

Creating from time.time()

```
>>> from async_v20 import DateTime
>>> from time import time
>>> dt = DateTime(time())
>>> dt
Timestamp('2017-12-21 01:22:37.762530+0000', tz='UTC')
>>> dt.json('UNIX')
'1513819357.762530000'
>>> dt.json('RFC3339')
'2017-12-21T01:22:37.762530000Z'
```

Creating from datetime.datetime.now()

```
>>> from async_v20 import DateTime
>>> from datetime import datetime
>>> dt = DateTime(datetime.now())
>>> dt
Timestamp('2017-12-21 12:31:03.982327')
```

DataFrame

```
>>> from async_v20 import OandaClient
>>> import asyncio
>>> loop = asyncio.get_event_loop()
>>> run = loop.run_until_complete
>>> client = OandaClient()
>>> rsp = run(client.get_candles('EUR_USD'))
>>> df = rsp.candles.dataframe()
>>> df.time[0]
... Timestamp('2017-12-20 23:30:40+0000', tz='UTC')
>>> df = rsp.candles.dataframe(datetime_format='RFC3339')
>>> df.time[0]
```

(continues on next page)

(continued from previous page)

```
'2017-12-20T23:30:40.000000000Z'
>>> df = rsp.candles.dataframe(datetime_format='UNIX')
>>> df.time[0]
1513812640000000000
>>> type(df.time[0])
# <class 'numpy.int64'>
>>> df = rsp.candles.dataframe(json=True, datetime_format='UNIX')
>>> df.time[0]
'1513812640.000000000'
>>> type(df.time[0])
# <class 'str'>
```

8.6 Health API

async_v20 includes OANDA's v20 health API.

During the initialization of OandaClient the statuses of the services are checked. A warning is logged for each service that is not currently up.

Users are encouraged to explicitly check the service they wish to use is available.

see *Health* for complete list of API calls.

Example:

```
>>> from async_v20 import OandaClient
>>> import asyncio
>>> client = OandaClient()
>>> loop = asyncio.get_event_loop()
>>> run = loop.run_until_complete
>>> rsp = run(client.list_services())
>>> rsp
# <Status [200]: services>
>>> rsp.services
# (<Service: id=fxtrade-practice-rest-api>,
# <Service: id=fxtrade-practice-streaming-api>,
# <Service: id=fxtrade-rest-api>,
# <Service: id=fxtrade-streaming-api>)
>>> rsp.services.get_id('fxtrade-practice-streaming-api')
# <Service: id=fxtrade-practice-streaming-api>
>>> rsp.services.get_id('fxtrade-practice-streaming-api').current_event.status.
↪description
# 'The service is up'
```

8.7 Traps for young players

Listed here will be traps that have caught me out. documented here so it doesn't catch you!

8.7.1 ERROR!

Have you written some error handling in the event the client connection falls over?

This is a very broad category of errors.

It may be:

- OANDA is down for maintenance
- Someone unplugged your computer
- The wifi dropped out
- Network config issue. IP conflict, routing, firewall…
- Your token expired (for some reason)

8.7.2 An Order is not an OrderRequest

Note: This is taken from the OANDA docs

Orders

- The specification of all Orders supported by the platform.

Order Requests:

- The request specification of all Orders supported by the platform. These objects are used by the API client to create Orders on the platform.

They key point here is that you need to use async_v20 objects that derive from OrderRequest when passing an order request to the order_request argument of:

```
post_order()
```

8.8 OandaClient API

Note: OandaClient will look for OANDA_TOKEN in the enviroment variables if no token is passed

8.8.1 OandaClient

class async_v20.**OandaClient**(*token=None*, *account_id=None*, *format_order_requests=False*, *max_transaction_history=100*, *rest_host='api-fxpractice.oanda.com'*, *rest_port=443*, *rest_scheme='https'*, *stream_host='stream-fxpractice.oanda.com'*, *stream_port=None*, *stream_scheme='https'*, *health_host='api-status.oanda.com'*, *health_port=80*, *health_scheme='http'*, *datetime_format='UNIX'*, *rest_timeout=10*, *stream_timeout=60*, *max_requests_per_second=99*, *max_simultaneous_connections=10*, *debug=False*)

Bases: `async_v20.interface.account.AccountInterface`, `async_v20.interface.instrument.InstrumentInterface`, `async_v20.interface.order.OrderInterface`, `async_v20.interface.position.PositionInterface`, `async_v20.interface.pricing.PricingInterface`, `async_v20.interface.trade.TradeInterface`, `async_v20.interface.transaction.TransactionInterface`, `async_v20.interface.user.UserInterface`, `async_v20.interface.health.HealthInterface`

Create an API context for v20 access

Parameters

- **`token`** – User generated token from the online account configuration page
- **`account_id`** – The account id the client will connect to. If None will default to the first account number returned by `list_accounts()`
- **`format_order_requests`** – True=Format all OrderRequests in the context of the orders instrument. False=Do not format OrderRequests, raise `InvalidOrderRequest` for values outside of allowed range.
- **`max_transaction_history`** – Maximum past transactions to store
- **`rest_host`** – The hostname of the v20 REST server
- **`rest_port`** – The port of the v20 REST server
- **`stream_host`** – The hostname of the v20 REST server
- **`stream_port`** – The port of the v20 REST server
- **`rest_scheme`** – The scheme of the connection to rest server.
- **`stream_scheme`** – The scheme of the connection to the stream server.
- **`health_host`** – The hostname of the health API server
- **`health_port`** – The port of the health server
- **`health_scheme`** – The scheme of the connection for the health server.
- **`datetime_format`** – The format to request when dealing with times
- **`rest_timeout`** – The timeout to use when making a polling request with the v20 REST server
- **`stream_timeout`** – Period to wait for an new json object during streaming
- **`max_requests_per_second`** – Maximum HTTP requests sent per second
- **`max_simultaneous_connections`** – Maximum concurrent HTTP requests
- **`debug`** – Set to True to log debug messages.

8.8.2 Account

`OandaClient.`**`account`**`()`

Get updated account

Returns `Account`

`OandaClient.`**`list_accounts`**`(`*self*`)`

Get a list of all Accounts authorized for the provided token.

Returns

status [200] `Response` (accounts=(`AccountProperties`, . . .),)

`OandaClient.`**`get_account_details`**`(`*self*`)`

Get the full details for a single Account that a client has access to. Full pending Order, open Trade and open Position representations are provided.

Returns

status [200] `Response` (account= `Account`, lastTransactionID= `TransactionID`)

`OandaClient.`**`account_summary`**(*self*)

Get a summary for a single Account that a client has access to.

Returns

status [200] *`Response`* (account= *`AccountSummary`*, lastTransactionID= *`TransactionID`*)

`OandaClient.`**`account_instruments`**(*self*, *instruments: Instruments= sentinel*)

Get the list of tradeable instruments for the given Account. The list of tradeable instruments is dependent on the regulatory division that the Account is located in, thus should be the same for all Accounts owned by a single user.

Parameters

- **`instruments`** – *`Instruments`*
- **`of instruments to query specifically.`** (*`list`*) –

Returns

status [200] *`Response`* (instruments=(`Instrument`, `...`), lastTransactionID= *`TransactionID`*)

`OandaClient.`**`configure_account`**(*self*, *alias: Alias= sentinel*, *margin_rate: DecimalNumber= sentinel*)

Set the client-configurable portions of an Account.

Parameters

- **`alias`** – *`Alias`* Client-defined alias (name) for the Account
- **`margin_rate`** – *`DecimalNumber`* The string representation of a decimal number.

Returns

status [200] *`Response`* (clientConfigureTransaction= *`ClientConfigureTransaction`*, lastTransactionID= *`TransactionID`*)

status [400] *`Response`* (clientConfigureRejectTransaction= *`ClientConfigureRejectTransaction`*, lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

status [403] *`Response`* (clientConfigureRejectTransaction= *`ClientConfigureRejectTransaction`*, lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`account_changes`**(*self*, *since_transaction_id: SinceTransactionID= sentinel*)

Endpoint used to poll an Account for its current state and changes since a specified TransactionID.

Note: OandaClient will supply since_transaction_id if None is provided

Parameters **`since_transaction_id`** – *`SinceTransactionID`* ID of the Transaction to get Account changes since.

Returns

status [200] *`Response`* (changes= *`AccountChanges`*, state= *`AccountChangesState`*, lastTransactionID= *`TransactionID`*)

8.8.3 Instrument

`OandaClient.`**`get_candles`**(*self*, *instrument: InstrumentName*, *price: PriceComponent=M*, *granularity: CandlestickGranularity=S5*, *count: Count= sentinel*, *from_time: FromTime= sentinel*, *to_time: ToTime= sentinel*, *smooth: Smooth=False*, *include_first_query: IncludeFirstQuery= sentinel*, *daily_alignment: DailyAlignment=17*, *alignment_timezone: AlignmentTimezone=America/New_York*, *weekly_alignment: WeeklyAlignment=Friday*)

Fetch candlestick data for an instrument.

Parameters

- **`include_first_query`** – *`IncludeFirstQuery`*
- **`instrument`** – *`InstrumentName`* Name of the Instrument
- **`price`** – `PriceComponent` The Price component(s) to get candlestick data for. Can contain any combination of the characters "M" (midpoint candles) "B" (bid candles) and "A" (ask candles).
- **`granularity`** – `CandlestickGranularity` The granularity of the candlesticks to fetch
- **`count`** – *`Count`* The number of candlesticks to return in the reponse. Count should not be specified if both the start and end parameters are provided, as the time range combined with the graularity will determine the number of candlesticks to return.
- **`from_time`** – *`FromTime`* The start of the time range to fetch candlesticks for.
- **`to_time`** – *`ToTime`* The end of the time range to fetch candlesticks for.
- **`smooth`** – *`Smooth`* A flag that controls whether the candlestick is "smoothed" or not. A smoothed candlestick uses the previous candle's close price as its open price, while an unsmoothed candlestick uses the first price from its time range as its open price.
- **`daily_alignment`** – *`DailyAlignment`* The hour of the day (in the specified timezone) to use for granularities that have daily alignments.
- **`alignment_timezone`** – *`AlignmentTimezone`* The timezone to use for the dailyAlignment parameter. Candlesticks with daily alignment will be aligned to the dailyAlignment hour within the alignmentTimezone.
- **`weekly_alignment`** – *`WeeklyAlignment`* The day of the week used for granularities that have weekly alignment.

Returns

status [200] *`Response`* (instrument= *`InstrumentName`*, granularity= *`CandlestickGranularity`*, candles=(*`Candlestick`*, . . .),)

`OandaClient.`**`get_order_book`**(*self*, *instrument: InstrumentName*, *time: DateTime= sentinel*)

Fetch a gzip compressed order book for an instrument

Parameters

- **`instrument`** – *`InstrumentName`* Name of the Instrument
- **`time`** – *`DateTime`* The time of the snapshot to fetch. If not specified, then the most recent snapshot is fetched

Returns

status [200] *`Response`* (orderBook= *`OrderBook`*)

OandaClient.**get_position_book**(*self*, *instrument: InstrumentName*, *time: DateTime= sentinel*)

Fetch a gzip compressed order book for an instrument

Parameters

- **instrument** – *InstrumentName* Name of the Instrument
- **time** – *DateTime* The time of the snapshot to fetch. If not specified, then the most recent snapshot is fetched

Returns

status [200] *Response* (positionBook= *PositionBook*)

8.8.4 Order

OandaClient.**post_order**(*self*, *order_request: OrderRequest= sentinel*)

Post an OrderRequest.

Parameters **order_request** – *OrderRequest* or a class derived from OrderRequest

Returns

status [201] *Response* (orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderCancelTransaction= *OrderCancelTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= str, errorMessage= str)

status [401] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= str, errorMessage= str)

OandaClient.**create_order**(*self*, *instrument: InstrumentName*, *units: DecimalNumber*, *type: OrderType=MARKET*, *trade_id: TradeID= sentinel*, *price: PriceValue= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition= sentinel*, *client_extensions: ClientExtensions= sentinel*, *distance: PriceValue= sentinel*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

create an OrderRequest

Parameters

- **trade_id** – *TradeID*
- **price** – *PriceValue*
- **type** – *OrderType*
- **client_trade_id** – *ClientID*
- **time_in_force** – *TimeInForce*
- **gtd_time** – *DateTime*

- **trigger_condition** – *OrderTriggerCondition*
- **client_extensions** – *ClientExtensions*
- **distance** – *PriceValue*
- **instrument** – *InstrumentName*
- **units** – `Unit`
- **price_bound** – *PriceValue*
- **position_fill** – *OrderPositionFill*
- **take_profit_on_fill** – *TakeProfitDetails*
- **stop_loss_on_fill** – *StopLossDetails*
- **trailing_stop_loss_on_fill** – *TrailingStopLossDetails*
- **trade_client_extensions** – *ClientExtensions*

Returns

status [201] *Response* (orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderCancelTransaction= *OrderCancelTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`list_orders`**(*self*, *ids: Ids= sentinel*, *state: OrderStateFilter= sentinel*, *instrument: InstrumentName= sentinel*, *count: Count= sentinel*, *before_id: OrderID= sentinel*)

Get a list of Orders for an Account

Parameters

- **ids** – *Ids* list of Order IDs to retrieve
- **state** – *OrderStateFilter* The state to filter the requested Orders by
- **instrument** – *InstrumentName* The instrument to filter the requested orders by
- **count** – *Count* The maximum number of Orders to return
- **before_id** – *OrderID* The maximum Order ID to return. If not provided the most recent Orders in the Account are returned

Returns

status [200] *Response* (orders=(*Order*, . . .), lastTransactionID= *TransactionID*)

`OandaClient.`**`list_pending_orders`**(*self*)

List all pending Orders

Returns

status [200] *Response* (orders=(*Order*, . . .), lastTransactionID= *TransactionID*)

OandaClient.**get_order**(*self*, *order_specifier: OrderSpecifier= sentinel*)

Get details for a single Order

Parameters **order_specifier** – *OrderSpecifier* The Order Specifier

Returns

status [200] *Response* (order= *Order*, lastTransactionID= *TransactionID*)

OandaClient.**replace_order**(*self*, *order_specifier: OrderSpecifier= sentinel*, *order_request: OrderRequest= sentinel*)

Replace an Order by simultaneously cancelling it and creating a replacement Order

Parameters

- **order_specifier** – *OrderSpecifier* The Order Specifier
- **order_request** – *OrderRequest* Specification of the replacing Order

Returns

status [201] *Response* (orderCancelTransaction= *OrderCancelTransaction*, orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, replacingOrderCancelTransaction= *OrderCancelTransaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderCancelRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

OandaClient.**cancel_order**(*self*, *order_specifier: OrderSpecifier= sentinel*)

Cancel a pending Order

Parameters **order_specifier** – *OrderSpecifier* The Order Specifier

Returns

status [200] *Response* (orderCancelTransaction= *OrderCancelTransaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*)

status [400] *Response* (orderCancelRejectTransaction= *OrderCancelRejectTransaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

OandaClient.**set_client_extensions**(*self*, *order_specifier: OrderSpecifier= sentinel*, *client_extensions: ClientExtensions= sentinel*, *trade_client_extensions: TradeClientExtensions= sentinel*)

Update the Client Extensions for an Order . Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Parameters

- **order_specifier** – *OrderSpecifier* The Order Specifier
- **client_extensions** – *ClientExtensions* The Client Extensions to update for the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

- **`trade_client_extensions`** – *`TradeClientExtensions`* The Client Extensions to update for the Trade created when the Order is filled. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [200] *`Response`* (orderClientExtensionsModifyTransaction= *`OrderClientExtensionsModifyTransaction`*, lastTransactionID= *`TransactionID`*, relatedTransactionIDs=(*`TransactionID`*, . . .),)

status [400] *`Response`* (orderClientExtensionsModifyRejectTransaction= *`OrderClientExtensionsModifyRejectTransaction`*, lastTransactionID= *`TransactionID`*, relatedTransactionIDs=(*`TransactionID`*, . . .), errorCode= `str`, errorMessage= `str`)

status [401] *`Response`* (orderClientExtensionsModifyRejectTransaction= *`OrderClientExtensionsModifyRejectTransaction`*, lastTransactionID= *`TransactionID`*, relatedTransactionIDs=(*`TransactionID`*, . . .), errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`market_order`**(*self*, *instrument: InstrumentName*, *units: DecimalNumber*, *time_in_force: TimeInForce=FOK*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Create a Market Order Request

Parameters

- **`instrument`** – *`InstrumentName`* The Market Order's Instrument.
- **`units`** – `Unit` The quantity requested to be filled by the Market Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.
- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the Market Order. Restricted to FOK or IOC for a MarketOrder.
- **`price_bound`** – *`PriceValue`* The worst price that the client is willing to have the Market Order filled at.
- **`position_fill`** – *`OrderPositionFill`* Specification of how Positions in the Account are modified when the Order is filled.
- **`client_extensions`** – *`ClientExtensions`* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.
- **`take_profit_on_fill`** – *`TakeProfitDetails`* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.
- **`stop_loss_on_fill`** – *`StopLossDetails`* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.
- **`trailing_stop_loss_on_fill`** – *`TrailingStopLossDetails`* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

- **`trade_client_extensions`** – *`ClientExtensions`* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

Returns

status [201] *`Response`* (orderCreateTransaction= *`Transaction`*, orderFillTransaction= *`OrderFillTransaction`*, orderCancelTransaction= *`OrderCancelTransaction`*, orderReissueTransaction= *`Transaction`*, orderReissueRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, . . .), lastTransactionID= *`TransactionID`*)

status [400] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, . . .), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

status [401] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, . . .), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`limit_order`**(*self*, *instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Create a Limit Order

Parameters

- **`instrument`** – *`InstrumentName`* The Limit Order's Instrument.
- **`units`** – `Unit` The quantity requested to be filled by the Limit Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.
- **`price`** – *`PriceValue`* The price threshold specified for the Limit Order. The Limit Order will only be filled by a market price that is equal to or better than this price.
- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the Limit Order.
- **`gtd_time`** – *`DateTime`* The date/time when the Limit Order will be cancelled if its timeInForce is "GTD".
- **`position_fill`** – *`OrderPositionFill`* Specification of how Positions in the Account are modified when the Order is filled.
- **`trigger_condition`** – *`OrderTriggerCondition`* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **`client_extensions`** – *`ClientExtensions`* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.
- **`take_profit_on_fill`** – *`TakeProfitDetails`* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.
- **`stop_loss_on_fill`** – *`StopLossDetails`* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled

that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

- **trailing_stop_loss_on_fill** – *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.
- **trade_client_extensions** – *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderCancelTransaction= *OrderCancelTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`limit_replace_order`**(*self*, *instrument: InstrumentName*, *order_specifier: OrderSpecifier*, *units: DecimalNumber*, *price: PriceValue*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Replace a pending Limit Order

Parameters

- **instrument** – *InstrumentName* The Limit Order's Instrument.
- **order_specifier** – *OrderSpecifier* The ID of the Limit Order to replace
- **units** – `Unit` The quantity requested to be filled by the Limit Order. A positive number of units results in a long Order, and a negative number of units results in a short Order.
- **price** – *PriceValue* The price threshold specified for the Limit Order. The Limit Order will only be filled by a market price that is equal to or better than this price.
- **time_in_force** – *TimeInForce* The time-in-force requested for the Limit Order.
- **gtd_time** – *DateTime* The date/time when the Limit Order will be cancelled if its timeInForce is "GTD".
- **position_fill** – *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.

- **trigger_condition** – *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **client_extensions** – *ClientExtensions* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.
- **take_profit_on_fill** – *TakeProfitDetails* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.
- **stop_loss_on_fill** – *StopLossDetails* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.
- **trailing_stop_loss_on_fill** – *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.
- **trade_client_extensions** – *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCancelTransaction= *OrderCancelTransaction*, orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, replacingOrderCancelTransaction= *OrderCancelTransaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderCancelRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`stop_order`**(*self*, *instrument: InstrumentName*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Create a Stop Order

Parameters

- **instrument** – *InstrumentName* The StopOrder's Instrument.
- **trade_id** – *TradeID* The ID of the Trade to close when the price threshold is breached.
- **client_trade_id** – *TradeID* The client ID of the Trade to be closed when the price threshold is breached.
- **price** – *PriceValue* The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.

- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.
- **`gtd_time`** – *`DateTime`* The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".
- **`trigger_condition`** – *`OrderTriggerCondition`* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **`client_extensions`** – *`ClientExtensions`* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] *`Response`* (orderCreateTransaction= *`Transaction`*, orderFillTransaction= *`OrderFillTransaction`*, orderCancelTransaction= *`OrderCancelTransaction`*, orderReissueTransaction= *`Transaction`*, orderReissueRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*)

status [400] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

status [401] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`stop_replace_order`**(*self*, *instrument: InstrumentName*, *order_specifier: OrderSpecifier*, *units: DecimalNumber*, *price: PriceValue*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Replace a pending Stop Order

Parameters

- **`instrument`** – *`InstrumentName`* The Stop Order's Instrument.
- **`order_specifier`** – *`OrderSpecifier`* The ID of the Stop Order to replace
- **`units`** – `Unit` The quantity requested to be filled by the Stop Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.
- **`price`** – *`PriceValue`* The price threshold specified for the Stop Order. The Stop Order will only be filled by a market price that is equal to or worse than this price.
- **`price_bound`** – *`PriceValue`* The worst market price that may be used to fill this Stop Order. If the market gaps and crosses through both the price and the priceBound, the Stop Order will be cancelled instead of being filled.
- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the Stop Order.
- **`gtd_time`** – *`DateTime`* The date/time when the Stop Order will be cancelled if its timeInForce is "GTD".

- **position_fill** – *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.
- **trigger_condition** – *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **client_extensions** – *ClientExtensions* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.
- **take_profit_on_fill** – *TakeProfitDetails* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.
- **stop_loss_on_fill** – *StopLossDetails* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.
- **trailing_stop_loss_on_fill** – *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.
- **trade_client_extensions** – *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCancelTransaction= *OrderCancelTransaction*, orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, replacingOrderCancelTransaction= *OrderCancelTransaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderCancelRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`market_if_touched_order`**(*self*, *instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Create a market if touched order

Parameters

- **instrument** – *InstrumentName* The MarketIfTouched Order's Instrument.
- **units** – Unit The quantity requested to be filled by the MarketIfTouched Order. A positive number of units results in a long Order, and a negative number of units results in a short Order.
- **price** – *PriceValue* The price threshold specified for the MarketIfTouched Order. The MarketIfTouched Order will only be filled by a market price that crosses this price from the direction of the market price at the time when the Order was created (the initialMarketPrice). Depending on the value of the Order's price and initialMarketPrice, the MarketIfTouchedOrder will behave like a Limit or a Stop Order.
- **price_bound** – *PriceValue* The worst market price that may be used to fill this MarketIfTouched Order.
- **time_in_force** – *TimeInForce* The time-in-force requested for the MarketIfTouched Order. Restricted to "GTC", "GFD" and "GTD" for MarketIfTouched Orders.
- **gtd_time** – *DateTime* The date/time when the MarketIfTouched Order will be cancelled if its timeInForce is "GTD".
- **position_fill** – *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.
- **trigger_condition** – *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **client_extensions** – *ClientExtensions* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.
- **take_profit_on_fill** – *TakeProfitDetails* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.
- **stop_loss_on_fill** – *StopLossDetails* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.
- **trailing_stop_loss_on_fill** – *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.
- **trade_client_extensions** – *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderCancelTransaction= *OrderCancelTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(

`TransactionID`, ...), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

status [401] `Response` (orderRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, ...), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`market_if_touched_replace_order`**(*self*, *instrument: InstrumentName*, *order_specifier: OrderSpecifier*, *units: DecimalNumber*, *price: PriceValue*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Replace a pending market if touched order

Parameters

- **`instrument`** – `InstrumentName` The MarketIfTouched Order's Instrument.
- **`order_specifier`** – `OrderSpecifier` The ID of the MarketIfTouched Order to replace
- **`units`** – `Unit` The quantity requested to be filled by the MarketIfTouched Order. A positive number of units results in a long Order, and a negative number of units results in a short Order.
- **`price`** – `PriceValue` The price threshold specified for the MarketIfTouched Order. The MarketIfTouched Order will only be filled by a market price that crosses this price from the direction of the market price at the time when the Order was created (the initialMarketPrice). Depending on the value of the Order's price and initialMarketPrice, the MarketIfTouchedOrder will behave like a Limit or a Stop Order.
- **`price_bound`** – `PriceValue` The worst market price that may be used to fill this MarketIfTouched Order.
- **`time_in_force`** – `TimeInForce` The time-in-force requested for the MarketIfTouched Order. Restricted to "GTC", "GFD" and "GTD" for MarketIfTouched Orders.
- **`gtd_time`** – `DateTime` The date/time when the MarketIfTouched Order will be cancelled if its timeInForce is "GTD".
- **`position_fill`** – `OrderPositionFill` Specification of how Positions in the Account are modified when the Order is filled.
- **`trigger_condition`** – `OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **`client_extensions`** – `ClientExtensions` The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

- **take_profit_on_fill** – *TakeProfitDetails* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.
- **stop_loss_on_fill** – *StopLossDetails* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.
- **trailing_stop_loss_on_fill** – *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.
- **trade_client_extensions** – *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCancelTransaction= *OrderCancelTransaction*, orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, replacingOrderCancelTransaction= *OrderCancelTransaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderCancelRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`take_profit_order`**(*self*, *instrument: InstrumentName*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Create a take profit order

Parameters

- **instrument** – *InstrumentName* The TakeProfitOrder's Instrument.
- **trade_id** – *TradeID* The ID of the Trade to close when the price threshold is breached.
- **client_trade_id** – *TradeID* The client ID of the Trade to be closed when the price threshold is breached.
- **price** – *PriceValue* The price threshold specified for the TakeProfit Order. The associated Trade will be closed by a market price that is equal to or better than this threshold.
- **time_in_force** – *TimeInForce* The time-in-force requested for the TakeProfit Order. Restricted to "GTC", "GFD" and "GTD" for TakeProfit Orders.
- **gtd_time** – *DateTime* The date/time when the TakeProfit Order will be cancelled if its timeInForce is "GTD".

- **trigger_condition** – *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **client_extensions** – *ClientExtensions* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderCancelTransaction= *OrderCancelTransaction*, orderReissueTransaction= *Transaction*, orderReissueRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

status [401] *Response* (orderRejectTransaction= *Transaction*, relatedTransactionIDs=(*TransactionID*, . . .), lastTransactionID= *TransactionID*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`take_profit_replace_order`**(*self*, *instrument: InstrumentName*, *order_specifier: OrderSpecifier*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Replace a pending take profit order

Parameters

- **instrument** – *InstrumentName* The TakeProfitOrder's Instrument.
- **order_specifier** – *OrderSpecifier* The ID of the Take Profit Order to replace
- **trade_id** – *TradeID* The ID of the Trade to close when the price threshold is breached.
- **client_trade_id** – *TradeID* The client ID of the Trade to be closed when the price threshold is breached.
- **price** – *PriceValue* The price threshold specified for the TakeProfit Order. The associated Trade will be closed by a market price that is equal to or better than this threshold.
- **time_in_force** – *TimeInForce* The time-in-force requested for the TakeProfit Order. Restricted to "GTC", "GFD" and "GTD" for TakeProfit Orders.
- **gtd_time** – *DateTime* The date/time when the TakeProfit Order will be cancelled if its timeInForce is "GTD".
- **trigger_condition** – *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **client_extensions** – *ClientExtensions* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] *Response* (orderCancelTransaction= *OrderCancelTransaction*, orderCreateTransaction= *Transaction*, orderFillTransaction= *OrderFillTransaction*, orderReissueTransaction= `Transaction`, orderReissueRejectTransaction= `Transaction`, replacingOrderCancelTransaction= `OrderCancelTransaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`)

status [400] `Response` (orderRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

status [401] `Response` (orderCancelRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`stop_loss_order`**(*self*, *instrument: InstrumentName*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Create a Stop Loss Order

Parameters

- **`instrument`** – `InstrumentName` The StopLossOrder's Instrument.
- **`trade_id`** – `TradeID` The ID of the Trade to close when the price threshold is breached.
- **`client_trade_id`** – `TradeID` The client ID of the Trade to be closed when the price threshold is breached.
- **`price`** – `PriceValue` The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.
- **`time_in_force`** – `TimeInForce` The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.
- **`gtd_time`** – `DateTime` The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".
- **`trigger_condition`** – `OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **`client_extensions`** – `ClientExtensions` The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] `Response` (orderCreateTransaction= `Transaction`, orderFillTransaction= `OrderFillTransaction`, orderCancelTransaction= `OrderCancelTransaction`, orderReissueTransaction= `Transaction`, orderReissueRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`)

status [400] `Response` (orderRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

status [401] `Response` (orderRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`stop_loss_replace_order`**(*self*, *instrument: InstrumentName*, *order_specifier: OrderSpecifier*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Replace a pending Stop Loss Order

Parameters

- **`instrument`** – *`InstrumentName`* The StopLossOrder's Instrument.
- **`order_specifier`** – *`OrderSpecifier`* The ID of the Stop Loss Order to replace
- **`trade_id`** – *`TradeID`* The ID of the Trade to close when the price threshold is breached.
- **`client_trade_id`** – *`TradeID`* The client ID of the Trade to be closed when the price threshold is breached.
- **`price`** – *`PriceValue`* The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.
- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.
- **`gtd_time`** – *`DateTime`* The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".
- **`trigger_condition`** – *`OrderTriggerCondition`* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **`client_extensions`** – *`ClientExtensions`* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] *`Response`* (orderCancelTransaction= *`OrderCancelTransaction`*, orderCreateTransaction= *`Transaction`*, orderFillTransaction= *`OrderFillTransaction`*, orderReissueTransaction= *`Transaction`*, orderReissueRejectTransaction= *`Transaction`*, replacingOrderCancelTransaction= *`OrderCancelTransaction`*, relatedTransactionIDs=(*`TransactionID`*, . . .), lastTransactionID= *`TransactionID`*)

status [400] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, . . .), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

status [401] *`Response`* (orderCancelRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, . . .), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`trailing_stop_loss_order`**(*self*, *instrument: InstrumentName*, *trade_id: TradeID*, *distance: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Create a Trailing Stop Loss Order

Parameters

- **`instrument`** – *`InstrumentName`* The TrailingStopLossOrder's Instrument.

- **`trade_id`** – *`TradeID`* The ID of the Trade to close when the price threshold is breached.
- **`client_trade_id`** – *`TradeID`* The client ID of the Trade to be closed when the price threshold is breached.
- **`distance`** – *`PriceValue`* The price distance specified for the TrailingStopLoss Order.
- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the TrailingStopLoss Order. Restricted to "GTC", "GFD" and "GTD" for TrailingStopLoss Orders.
- **`gtd_time`** – *`DateTime`* The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".
- **`trigger_condition`** – *`OrderTriggerCondition`* Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **`client_extensions`** – *`ClientExtensions`* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] *`Response`* (orderCreateTransaction= *`Transaction`*, orderFillTransaction= *`OrderFillTransaction`*, orderCancelTransaction= *`OrderCancelTransaction`*, orderReissueTransaction= *`Transaction`*, orderReissueRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*)

status [400] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

status [401] *`Response`* (orderRejectTransaction= *`Transaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`trailing_stop_loss_replace_order`**(*self*, *instrument: InstrumentName*, *order_specifier: OrderSpecifier*, *trade_id: TradeID*, *distance: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Replace a pending Trailing Stop Loss Order

Parameters

- **`instrument`** – *`InstrumentName`* The TrailingStopLossOrder's Instrument.
- **`order_specifier`** – *`OrderSpecifier`* The ID of the Take Profit Order to replace
- **`trade_id`** – *`TradeID`* The ID of the Trade to close when the price threshold is breached.
- **`client_trade_id`** – *`TradeID`* The client ID of the Trade to be closed when the price threshold is breached.
- **`distance`** – *`PriceValue`* The price distance specified for the TrailingStopLoss Order.
- **`time_in_force`** – *`TimeInForce`* The time-in-force requested for the TrailingStopLoss Order. Restricted to "GTC", "GFD" and "GTD" for TrailingStopLoss Orders.

- **gtd_time** – `DateTime` The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".
- **trigger_condition** – `OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.
- **client_extensions** – `ClientExtensions` The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

Returns

status [201] `Response` (orderCancelTransaction= `OrderCancelTransaction`, orderCreateTransaction= `Transaction`, orderFillTransaction= `OrderFillTransaction`, orderReissueTransaction= `Transaction`, orderReissueRejectTransaction= `Transaction`, replacingOrderCancelTransaction= `OrderCancelTransaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`)

status [400] `Response` (orderRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

status [401] `Response` (orderCancelRejectTransaction= `Transaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`, errorCode= `str`, errorMessage= `str`)

8.8.5 Position

`OandaClient.`**`list_positions`**(*self*)

List all Positions for an Account. The Positions returned are for every instrument that has had a position during the lifetime of an the Account.

Returns

status [200] `Response` (positions=(`Position`, . . .), lastTransactionID= `TransactionID`)

`OandaClient.`**`list_open_positions`**(*self*)

List all open Positions for an Account. An open Position is a Position in an Account that currently has a Trade opened for it.

Returns

status [200] `Response` (positions=(`Position`, . . .), lastTransactionID= `TransactionID`)

`OandaClient.`**`get_position`**(*self*, *instrument: InstrumentName= sentinel*)

Get the details of a single Instrument's Position in an Account. The Position may by open or not.

Parameters **instrument** – `InstrumentName` Name of the Instrument

Returns

status [200] `Response` (position= `Position`, lastTransactionID= `TransactionID`)

`OandaClient.`**`close_position`**(*self*, *instrument: InstrumentName= sentinel*, *long_units: LongUnits= sentinel*, *long_client_extensions: LongClientExtensions= sentinel*, *short_units: ShortUnits= sentinel*, *short_client_extensions: ShortClientExtensions= sentinel*)

Closeout the open Position for a specific instrument in an Account.

Note:

- Either long_units or short_units **MUST** be specified.
- Do **NOT** specify *ALL* for *long_units* **or** *short_units* if there are no units to close.

Parameters

- **`instrument`** – *`InstrumentName`* Name of the Instrument
- **`long_units`** – *`LongUnits`* Indication of how much of the long Position to closeout. Either the string "ALL", the string "NONE", or a DecimalNumber representing how many units of the long position to close using a PositionCloseout MarketOrder. The units specified must always be positive.
- **`long_client_extensions`** – *`LongClientExtensions`* The client extensions to add to the MarketOrder used to close the long position.
- **`short_units`** – *`ShortUnits`* Indication of how much of the short Position to close-out. Either the string "ALL", the string "NONE", or a DecimalNumber representing how many units of the short position to close using a PositionCloseout MarketOrder. The units specified must always be positive.
- **`short_client_extensions`** – *`ShortClientExtensions`* The client extensions to add to the MarketOrder used to close the short position.

Returns

status [200] *`Response`* (longOrderCreateTransaction= *`MarketOrderTransaction`*, longOrderFillTransaction= *`OrderFillTransaction`*, longOrderCancelTransaction= *`OrderCancelTransaction`*, shortOrderCreateTransaction= *`MarketOrderTransaction`*, shortOrderFillTransaction= *`OrderFillTransaction`*, shortOrderCancelTransaction= *`OrderCancelTransaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*)

status [400] *`Response`* (longOrderRejectTransaction= *`MarketOrderRejectTransaction`*, shortOrderRejectTransaction= *`MarketOrderRejectTransaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

status [401] *`Response`* (longOrderRejectTransaction= *`MarketOrderRejectTransaction`*, shortOrderRejectTransaction= *`MarketOrderRejectTransaction`*, relatedTransactionIDs=(*`TransactionID`*, ...), lastTransactionID= *`TransactionID`*, errorCode= `str`, errorMessage= `str`)

8.8.6 Pricing

`OandaClient.`**`get_pricing`**(*self*, *instruments: Instruments= sentinel*, *since: DateTime= sentinel*)

Get pricing information for a specified list of Instruments within an Account.

Parameters

- **`instruments`** – *`Instruments`*
- **`of Instruments to get pricing for.`** (*`list`*) –
- **`since`** – *`DateTime`* Date/Time filter to apply to the returned prices. Only prices with a time later than this filter will be provided.

Returns

status [200] *Response* (prices=(*Price*, . . .), time= *DateTime*)

`OandaClient.`**`stream_pricing`**(*self*, *instruments: Instruments= sentinel*, *snapshot: Snapshot= sentinel*)

Get a stream of Account Prices starting from when the request is made. This pricing stream does not include every single price created for the Account, but instead will provide at most 4 prices per second (every 250 milliseconds) for each instrument being requested. If more than one price is created for an instrument during the 250 millisecond window, only the price in effect at the end of the window is sent. This means that during periods of rapid price movement, subscribers to this stream will not be sent every price. Pricing windows for different connections to the price stream are not all aligned in the same way (i.e. they are not all aligned to the top of the second). This means that during periods of rapid price movement, different subscribers may observe different prices depending on their alignment.

Parameters

- **instruments** – *Instruments*
- **of Instruments to stream Prices for.** (*list*) –
- **snapshot** – *Snapshot* Flag that enables/disables the sending of a pricing snapshot when initially connecting to the stream.

Returns

status [200] *Response* (price= *Price*)

OR

Response (heartbeat= *PricingHeartbeat*)

8.8.7 Trade

`OandaClient.`**`list_trades`**(*self*, *ids: Ids= sentinel*, *state: TradeStateFilter= sentinel*, *instrument: InstrumentName= sentinel*, *count: Count= sentinel*, *trade_id: TradeID= sentinel*)

Get a list of Trades for an Account

Parameters

- **ids** – *Ids* List of Trade IDs to retrieve.
- **state** – *TradeStateFilter* The state to filter the requested Trades by.
- **instrument** – *InstrumentName* The instrument to filter the requested Trades by.
- **count** – *Count* The maximum number of Trades to return.
- **trade_id** – *TradeID* The maximum Trade ID to return. If not provided the most recent Trades in the Account are returned.

Returns

status [200] *Response* (trades=(*Trade*, . . .), lastTransactionID= *TransactionID*)

`OandaClient.`**`list_open_trades`**(*self*)

Get the list of open Trades for an Account

Returns

status [200] *Response* (trades=(*Trade*, . . .), lastTransactionID= *TransactionID*)

`OandaClient.`**`get_trade`**(*self*, *trade_specifier: TradeSpecifier= sentinel*)

Get the details of a specific Trade in an Account

Parameters **trade_specifier** – *TradeSpecifier* Specifier for the Trade

Returns

status [200] *Response* (trade= *Trade*, lastTransactionID= *TransactionID*)

OandaClient.**close_trade**(*self*, *trade_specifier: TradeSpecifier= sentinel*, *units: Units= sentinel*)

Close (partially or fully) a specific open Trade in an Account

Parameters

- **trade_specifier** – *TradeSpecifier* Specifier for the Trade
- **units** – *Units* Indication of how much of the Trade to close. Either the string "ALL" (indicating that all of the Trade should be closed), or a DecimalNumber representing the number of units of the open Trade to Close using a TradeClose MarketOrder. The units specified must always be positive, and the magnitude of the value cannot exceed the magnitude of the Trade's open units.

Returns

status [200] *Response* (orderCreateTransaction= *MarketOrderTransaction*, orderFillTransaction= *OrderFillTransaction*, orderCancelTransaction= *OrderCancelTransaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*)

status [400] *Response* (orderRejectTransaction= *MarketOrderRejectTransaction*, errorCode= str, errorMessage= str)

status [401] *Response* (orderRejectTransaction= *MarketOrderRejectTransaction*, lastTransactionID= *TransactionID*, relatedTransactionIDs=(*TransactionID*, ...), errorCode= str, errorMessage= str)

OandaClient.**close_all_trades**()

Close all open trades

Returns class'~async_v20.interface.response.Response', ...])

Return type tuple(bool, [

OandaClient.**set_client_extensions_trade**(*self*, *trade_specifier: TradeSpecifier= sentinel*, *client_extensions: ClientExtensions= sentinel*)

Update the Client Extensions for a Trade. Do not add, update, or delete the Client Extensions if your account is associated with MT4.

Parameters

- **trade_specifier** – *TradeSpecifier* Specifier for the Trade
- **client_extensions** – *ClientExtensions* The Client Extensions to update the Trade with. Do not add, update, or delete the Client Extensions if your account is associated with MT4.

Returns

status [200] *Response* (tradeClientExtensionsModifyTransaction= *TradeClientExtensionsModifyTransaction*, relatedTransactionIDs=(*TransactionID*, ...), lastTransactionID= *TransactionID*)

status [400] *Response* (tradeClientExtensionsModifyRejectTransaction= *TradeClientExtensionsModifyRejectTransaction*, lastTransactionID= *TransactionID*, relatedTransactionIDs=(*TransactionID*, ...), errorCode= str, errorMessage= str)

status [401] `Response` (tradeClientExtensionsModifyRejectTransaction= `TradeClientExtensionsModifyRejectTransaction`, lastTransactionID= `TransactionID`, relatedTransactionIDs=(`TransactionID`, . . .), errorCode= `str`, errorMessage= `str`)

`OandaClient.`**`set_dependent_orders_trade`**(*self*, *trade_specifier: TradeSpecifier= sentinel*, *take_profit: TakeProfitDetails= sentinel*, *stop_loss: StopLossDetails= sentinel*, *trailing_stop_loss: TrailingStopLossDetails= sentinel*)

Create, replace and cancel a Trade's dependent Orders (Take Profit, Stop Loss and Trailing Stop Loss) through the Trade itself

Parameters

- **`trade_specifier`** – `TradeSpecifier` Specifier for the Trade
- **`take_profit`** – `TakeProfitDetails` The specification of the Take Profit to create/modify/cancel. If takeProfit is set to null, the Take Profit Order will be cancelled if it exists. If takeProfit is not provided, the existing Take Profit Order will not be modified. If a sub- field of takeProfit is not specified, that field will be set to a default value on create, and be inherited by the replacing order on modify.
- **`stop_loss`** – `StopLossDetails` The specification of the Stop Loss to create/modify/cancel. If stopLoss is set to null, the Stop Loss Order will be cancelled if it exists. If stopLoss is not provided, the existing Stop Loss Order will not be modified. If a sub-field of stopLoss is not specified, that field will be set to a default value on create, and be inherited by the replacing order on modify.
- **`trailing_stop_loss`** – `TrailingStopLossDetails` The specification of the Trailing Stop Loss to create/modify/cancel. If trailingStopLoss is set to null, the Trailing Stop Loss Order will be cancelled if it exists. If trailingStopLoss is not provided, the existing Trailing Stop Loss Order will not be modified. If a sub-field of trailingStopLoss is not specified, that field will be set to a default value on create, and be inherited by the replacing order on modify.

Returns

status [200] `Response` (takeProfitOrderCancelTransaction= `OrderCancelTransaction`, takeProfitOrderTransaction= `TakeProfitOrderTransaction`, takeProfitOrderFillTransaction= `OrderFillTransaction`, takeProfitOrderCreatedCancelTransaction= `OrderCancelTransaction`, stopLossOrderCancelTransaction= `OrderCancelTransaction`, stopLossOrderTransaction= `StopLossOrderTransaction`, stopLossOrderFillTransaction= `OrderFillTransaction`, stopLossOrderCreatedCancelTransaction= `OrderCancelTransaction`, trailingStopLossOrderCancelTransaction= `OrderCancelTransaction`, trailingStopLossOrderTransaction= `TrailingStopLossOrderTransaction`, relatedTransactionIDs=(`TransactionID`, . . .), lastTransactionID= `TransactionID`)

status [400] `Response` (takeProfitOrderCancelRejectTransaction= `OrderCancelRejectTransaction`, takeProfitOrderRejectTransaction= `TakeProfitOrderRejectTransaction`, stopLossOrderCancelRejectTransaction= `OrderCancelRejectTransaction`, stopLossOrderRejectTransaction= `StopLossOrderRejectTransaction`, trailingStopLossOrderCancelRejectTransaction= `OrderCancelRejectTransaction`, trailingStopLossOrderRejectTransaction= `TrailingStopLossOrderRejectTransaction`, lastTransactionID= `TransactionID`, relatedTransactionIDs=(`TransactionID`, . . .), errorCode= `str`, errorMessage= `str`)

8.8.8 Transaction

`OandaClient.`**`list_transactions`**(*self*, *from_time: FromTime= sentinel*, *to_time: ToTime= sentinel*, *page_size: PageSize=100*, *type_: Type= sentinel*)

Get a list of Transactions pages that satisfy a time-based Transaction query.

Parameters

- **`from_time`** – *`FromTime`* The starting time (inclusive) of the time range for the Transactions being queried.
- **`to_time`** – *`ToTime`* The ending time (inclusive) of the time range for the Transactions being queried.
- **`page_size`** – *`PageSize`* The number of Transactions to include in each page of the results.
- **`type`** – *`Type`* A filter for restricting the types of Transactions to retrieve.

Returns

status [200] *`Response`* (from= *`DateTime`*, to= *`DateTime`*, pageSize= `int`, type=(*`TransactionFilter`*, ...), count= `int`, pages=(`str`, ...), lastTransactionID= *`TransactionID`*)

`OandaClient.`**`get_transaction`**(*self*, *transaction_id: TransactionID*)

Get the details of a single Account Transaction.

Parameters **`transaction_id`** – *`TransactionID`* A Transaction ID

Returns

status [200] *`Response`* (transaction= *`Transaction`*, lastTransactionID= *`TransactionID`*)

`OandaClient.`**`transaction_range`**(*self*, *from_transaction: FromTransactionID*, *to_transaction: ToTransactionID*, *type_: Type= sentinel*)

Get a range of Transactions for an Account based on the Transaction IDs.

Parameters

- **`from_transaction`** – *`FromTransactionID`* The starting Transaction ID (inclusive) to fetch.
- **`to_transaction`** – *`ToTransactionID`* The ending Transaction ID (inclusive) to fetch.
- **`type`** – *`Type`* The filter that restricts the types of Transactions to retrieve.

Returns

status [200] *`Response`* (transactions=(*`Transaction`*, ...), lastTransactionID= *`TransactionID`*)

`OandaClient.`**`since_transaction`**(*self*, *transaction_id: TransactionID= sentinel*)

Get a range of Transactions for an Account starting at (but not including) a provided Transaction ID.

Parameters **`transaction_id`** – *`TransactionID`* The ID of the last Transaction fetched. This query will return all Transactions newer than the TransactionID.

Returns

status [200] *`Response`* (transactions=(*`Transaction`*, ...), lastTransactionID= *`TransactionID`*)

`OandaClient.`**`stream_transactions`**`(`*self*`)`

Get a stream of Transactions for an Account starting from when the request is made.

Returns

status [200] *`Response`* (transaction= *`Transaction`*)

OR

`Response` (Heartbeat= *`TransactionHeartbeat`*)

8.8.9 User

`OandaClient.`**`get_user_info`**`(`*self*, *user_specifier: UserSpecifier*`)`

Fetch the user information for the specified user. This endpoint is intended to be used by the user thyself to obtain their own information.

Parameters **`user_specifier`** – *`UserSpecifier`* The User Specifier

Returns

status [200] *`Response`* (userInfo= `UserInfo`)

`OandaClient.`**`get_external_user_info`**`(`*self*, *user_specifier: UserSpecifier*`)`

Fetch the externally-available user information for the specified user. This endpoint is intended to be used by 3rd parties that have been authorized by a user to view their personal information.

Parameters **`user_specifier`** – *`UserSpecifier`* The User Specifier

Returns

status [200] *`Response`* (userInfo= `UserInfoExternal`)

8.8.10 Health

`OandaClient.`**`get_current_event`**`(`*self*, *service_id: ServiceID*`)`

Get the current event for a service

Parameters **`service_id`** – *`ServiceID`* The service to get the current event for

Returns

status [200] *`Response`* (Event= `Event`)

`OandaClient.`**`get_event`**`(`*self*, *service_id: ServiceID*, *event_sid: EventSid*`)`

Get an individual event

Parameters

- **`service_id`** – *`ServiceID`* The service to event for
- **`event_sid`** – *`EventSid`* The event to get from the specified service

Returns

status [200] *`Response`* (Event= `Event`)

`OandaClient.`**`get_service`**`(`*self*, *service_id: ServiceID*`)`

Get a single service

Parameters **`service_id`** – *`ServiceID`* Name of the service to get

Returns

status [200] `Response` (Service= `Service`)

`OandaClient.`**`get_service_list`**(*self*, *service_list_id: ServiceListID*)

Get a single service list

Parameters **`service_list_id`** – `ServiceListID` The service list to get.

Returns

status [200] `Response` (lists= `ServiceList`)

`OandaClient.`**`get_status`**(*self*, *status_id: StatusID*)

Get an individual status

Parameters **`status_id`** – `StatusID` The status to get

Returns

status [200] `Response` (Status= `Status`)

`OandaClient.`**`list_events`**(*self*, *service_id: ServiceID*)

List all events for a service

Parameters **`service_id`** – `ServiceID` The service to get events for.

Returns

status [200] `Response` (lists=(`Event`,. . .))

`OandaClient.`**`list_images`**(*self*)

List all status images

Returns

status [200] `Response` (images=(`Image`, . . .))

`OandaClient.`**`list_service_lists`**(*self*)

List all service lists

Returns

status [200] `Response` (lists=(`ServiceList`, . . .))

`OandaClient.`**`list_services`**(*self*)

List all the services

Returns

status [200] `Response` (services=(`Service`, . . .))

`OandaClient.`**`list_statuses`**(*self*)

List all statuses

Returns

status [200] `Response` (statuses=(`Event`, . . .))

8.9 The Response Object

All API methods apart from `account()` and `close_all_trades()` return `Response` objects

The response object is a `dict` with a few added methods:

- Truth testing returns true when the Response contains an expected status
- __repr__ displays all keys

class `async_v20.interface.response.`**`Response`**(*data*, *status*, *bool*, *datetime_format*)

Bases: *`dict`*

A response from OANDA.

Allows dotted attribute access

`Response.`**`json`**(*datetime_format=None*)

Return the json equivalent of the response

`Response.`**`dict`**(*json=False*, *datetime_format=None*)

Convert the response to a nested dictionary

Parameters **`json`** – Convert object attributes to the *JSON* representation

8.10 Particularly Pertinent Programming Interface

There are two additional helper methods that *OandaClient* exposes, beyond OANDA's own v20 client; *`account()`*, *`close_all_trades()`*.

8.10.1 Getting the Account Status

OANDA's v20 API is a RESTful service. Meaning that the server only sends the changes to the account, rather then the full account. This is useful, in that it reduces network traffic, but requires addition computation on the client side. As changes sent from the server must be incorporated locally.

This functionality is implemented in the *`account()`* method.

In order to check the status of the account. Use the await syntax, like so:

```
async def get_account():
    async with OandaClient() as client:
        return await client.account()
```

Note: The *`account()`* method uses the response from *`account_changes()`* to update the local account object. the response from *`account_changes()`* contains more information than the `Account` specifies. Meaning that some information contained in the *`account_changes()`* response is lost when updating the account. It has been chosen to only keep `Order`'s `Trade`'s `Position`'s, in the account that are currently open. This is to resemble the behaviour of OANDA's Web based browser interface

`Transaction`'s are stored on the client as `transactions`

8.10.2 Closing all Trades

The *`close_all_trades()`* method is provided to help facilitate your risk management policy. async_v20 is intended to be used as an algorithmic trading platform, which naturally raises concerns of run away losses, to the unbeknown user.

`close_all_trades()` is intended to help mitigate this concern by allowing users to programme a *global* stop loss by which all trades can be terminated.

Note: Users who implement this feature should account for the **Two** possible outcomes.

- A `CloseAllTradesFailure` is raised

OR

- returns (**True**, *closed_trade_responses*) - All trades were closed

8.11 Class Definitions

This page contains python class definitions of all types defined in OANDA's docs

8.11.1 Account

class async_v20.**AccountProperties**(*id: AccountID= sentinel*, *mt4_account_id: int= sentinel*, *tags: ArrayStr= sentinel*)

Bases: `async_v20.definitions.base.Model`

Properties related to an Account.

id
`AccountID` The Account's identifier

mt4account_id
`AccountID` The Account's associated MT4 Account ID. This field will not be present if the Account is not an MT4 account.

tags
(`str`, ...) The Account's tags

class async_v20.**AccountChangesState**(*unrealized_pl: AccountUnits= sentinel*, *nav: AccountUnits= sentinel*, *margin_used: AccountUnits= sentinel*, *margin_available: AccountUnits= sentinel*, *position_value: AccountUnits= sentinel*, *margin_closeout_unrealized_pl: AccountUnits= sentinel*, *margin_closeout_nav: AccountUnits= sentinel*, *margin_closeout_margin_used: AccountUnits= sentinel*, *margin_closeout_percent: DecimalNumber= sentinel*, *margin_closeout_position_value: DecimalNumber= sentinel*, *withdrawal_limit: AccountUnits= sentinel*, *margin_call_margin_used: AccountUnits= sentinel*, *margin_call_percent: DecimalNumber= sentinel*, *orders: ArrayDynamicOrderState= sentinel*, *trades: ArrayCalculatedTradeState= sentinel*, *positions: ArrayCalculatedPositionState= sentinel*, *balance: AccountUnits= sentinel*, *pl: AccountUnits= sentinel*, *resettable_pl: AccountUnits= sentinel*, *commission: AccountUnits= sentinel*, *guaranteed_execution_fees: AccountUnits= sentinel*)

Bases: `async_v20.definitions.base.Model`

An AccountState Object is used to represent an Account's current price- dependent state. Price-dependent Account state is dependent on OANDA's current Prices, and includes things like unrealized PL, NAV and Trailing Stop Loss Order state.

unrealized_pl
: *AccountUnits* The total unrealized profit/loss for all Trades currently open in the Account. Represented in the Account's home currency.

nav
: *AccountUnits* The net asset value of the Account. Equal to Account balance + unrealizedPL. Represented in the Account's home currency.

margin_used
: *AccountUnits* Margin currently used for the Account. Represented in the Account's home currency.

margin_available
: *AccountUnits* Margin available for Account. Represented in the Account's home currency.

position_value
: *AccountUnits* The value of the Account's open positions represented in the Account's home currency.

margin_closeout_unrealized_pl
: *AccountUnits* The Account's margin closeout unrealized PL.

margin_closeout_nav
: *AccountUnits* The Account's margin closeout NAV.

margin_closeout_margin_used
: *AccountUnits* The Account's margin closeout margin used.

margin_closeout_percent
: *DecimalNumber* The Account's margin closeout percentage. When this value is 1.0 or above the Account is in a margin closeout situation.

margin_closeout_position_value
: *DecimalNumber* The value of the Account's open positions as used for margin closeout calculations represented in the Account's home currency.

withdrawal_limit
: *AccountUnits* The current WithdrawalLimit for the account which will be zero or a positive value indicating how much can be withdrawn from the account.

margin_call_margin_used
: *AccountUnits* The Account's margin call margin used.

margin_call_percent
: *DecimalNumber* The Account's margin call percentage. When this value is 1.0 or above the Account is in a margin call situation.

orders
: (*DynamicOrderState*, . . .) The price-dependent state of each pending Order in the Account.

trades
: (*CalculatedTradeState*, . . .) The price-dependent state for each open Trade in the Account.

positions
: (*CalculatedPositionState*, ..) The price-dependent state for each open Position in the Account.

# TODO add documentation for Pl and resettabel_pl

class async_v20.**AccountSummary**(*id: AccountID= sentinel, alias: str= sentinel, currency: Currency= sentinel, balance: AccountUnits= sentinel, created_by_user_id: int= sentinel, created_time: DateTime= sentinel, pl: AccountUnits= sentinel, resettable_pl: AccountUnits= sentinel, resettabled_pl_time: DateTime= sentinel, commission: AccountUnits= sentinel, margin_rate: DecimalNumber= sentinel, margin_call_enter_time: DateTime= sentinel, margin_call_extension_count: int= sentinel, last_margin_call_extension_time: DateTime= sentinel, open_trade_count: int= sentinel, open_position_count: int= sentinel, pending_order_count: int= sentinel, hedging_enabled: bool= sentinel, unrealized_pl: AccountUnits= sentinel, nav: AccountUnits= sentinel, margin_used: AccountUnits= sentinel, margin_available: AccountUnits= sentinel, position_value: AccountUnits= sentinel, margin_closeout_unrealized_pl: AccountUnits= sentinel, margin_closeout_nav: AccountUnits= sentinel, margin_closeout_margin_used: AccountUnits= sentinel, margin_closeout_percent: DecimalNumber= sentinel, margin_closeout_position_value: DecimalNumber= sentinel, withdrawal_limit: AccountUnits= sentinel, margin_call_margin_used: AccountUnits= sentinel, margin_call_percent: DecimalNumber= sentinel, last_transaction_id: TransactionID= sentinel, trades: ArrayTradeSummary= sentinel, positions: ArrayPosition= sentinel, orders: ArrayOrder= sentinel, financing: DecimalNumber= sentinel, guaranteed_stop_loss_order_mode: GuaranteedStopLossOrderMode= sentinel, resettable_pl_time: DateTime= sentinel, guaranteed_execution_fees: AccountUnits= sentinel, dividend: DecimalNumber= sentinel*)

Bases: `async_v20.definitions.base.Model`

A summary representation of a client's Account. The AccountSummary does not provide to full specification of pending Orders, open Trades and Positions.

id

AccountID The Account's identifier

alias

`str` Client-assigned alias for the Account. Only provided if the Account has an alias set

currency

Currency The home currency of the Account

balance

AccountUnits The current balance of the Account. Represented in the Account's home currency.

created_by_user_id

`int` ID of the user that created the Account.

created_time

DateTime The date/time when the Account was created.

pl

AccountUnits The total profit/loss realized over the lifetime of the Account. Represented in the Account's home currency.

resettable_pl

AccountUnits The total realized profit/loss for the Account since it was last reset by the client. Represented in the Account's home currency.

resettabled_pl_time
DateTime The date/time that the Account's resettablePL was last reset.

commission
AccountUnits The total amount of commission paid over the lifetime of the Account. Represented in the Account's home currency.

margin_rate
DecimalNumber Client-provided margin rate override for the Account. The effective margin rate of the Account is the lesser of this value and the OANDA margin rate for the Account's division. This value is only provided if a margin rate override exists for the Account.

margin_call_enter_time
DateTime The date/time when the Account entered a margin call state. Only provided if the Account is in a margin call.

margin_call_extension_count
`int` The number of times that the Account's current margin call was extended.

last_margin_call_extension_time
DateTime The date/time of the Account's last margin call extension.

open_trade_count
`int` The number of Trades currently open in the Account.

open_position_count
`int` The number of Positions currently open in the Account.

pending_order_count
`int` The number of Orders currently pending in the Account.

hedging_enabled
`bool` Flag indicating that the Account has hedging enabled.

unrealized_pl
AccountUnits The total unrealized profit/loss for all Trades currently open in the Account. Represented in the Account's home currency.

nav
AccountUnits The net asset value of the Account. Equal to Account balance + unrealizedPL. Represented in the Account's home currency.

margin_used
AccountUnits Margin currently used for the Account. Represented in the Account's home currency.

margin_available
AccountUnits Margin available for Account. Represented in the Account's home currency.

position_value
AccountUnits The value of the Account's open positions represented in the Account's home currency.

margin_closeout_unrealized_pl
AccountUnits The Account's margin closeout unrealized PL.

margin_closeout_nav
AccountUnits The Account's margin closeout NAV.

margin_closeout_margin_used
AccountUnits The Account's margin closeout margin used.

margin_closeout_percent
DecimalNumber The Account's margin closeout percentage. When this value is 1.0 or above the Account is in a margin closeout situation.

margin_closeout_position_value
`DecimalNumber` The value of the Account's open positions as used for margin closeout calculations represented in the Account's home currency.

withdrawal_limit
`AccountUnits` The current WithdrawalLimit for the account which will be zero or a positive value indicating how much can be withdrawn from the account.

margin_call_margin_used
`AccountUnits` The Account's margin call margin used.

margin_call_percent
`DecimalNumber` The Account's margin call percentage. When this value is 1.0 or above the Account is in a margin call situation.

last_transaction_id
`TransactionID` The ID of the last Transaction created for the Account.

dividend
`DecimalNumber` Dividend

dividendAdjustment
`AccountUnits` Something

class async_v20.**Account**(*id: AccountID= sentinel, alias: str= sentinel, currency: Currency= sentinel, balance: AccountUnits= sentinel, created_by_user_id: int= sentinel, created_time: DateTime= sentinel, pl: AccountUnits= sentinel, resettable_pl: AccountUnits= sentinel, resettabled_pl_time: DateTime= sentinel, commission: AccountUnits= sentinel, margin_rate: DecimalNumber= sentinel, margin_call_enter_time: DateTime= sentinel, margin_call_extension_count: int= sentinel, last_margin_call_extension_time: DateTime= sentinel, open_trade_count: int= sentinel, open_position_count: int= sentinel, pending_order_count: int= sentinel, hedging_enabled: bool= sentinel, unrealized_pl: AccountUnits= sentinel, nav: AccountUnits= sentinel, margin_used: AccountUnits= sentinel, margin_available: AccountUnits= sentinel, position_value: AccountUnits= sentinel, margin_closeout_unrealized_pl: AccountUnits= sentinel, margin_closeout_nav: AccountUnits= sentinel, margin_closeout_margin_used: AccountUnits= sentinel, margin_closeout_percent: DecimalNumber= sentinel, margin_closeout_position_value: DecimalNumber= sentinel, withdrawal_limit: AccountUnits= sentinel, margin_call_margin_used: AccountUnits= sentinel, margin_call_percent: DecimalNumber= sentinel, last_transaction_id: TransactionID= sentinel, trades: ArrayTradeSummary= sentinel, positions: ArrayPosition= sentinel, orders: ArrayOrder= sentinel, financing: DecimalNumber= sentinel, guaranteed_stop_loss_order_mode: GuaranteedStopLossOrderMode= sentinel, resettable_pl_time: DateTime= sentinel, dividend: DecimalNumber= sentinel, dividend_adjustment: AccountUnits= sentinel, guaranteed_execution_fees: AccountUnits= sentinel*)

Bases: `async_v20.definitions.types.AccountSummary`

The full details of a client's Account. This includes full open Trade, open Position and pending Order representation.

id
`AccountID` The Account's identifier

alias
`str` Client-assigned alias for the Account. Only provided if the Account has an alias set

currency
Currency The home currency of the Account

balance
AccountUnits The current balance of the Account. Represented in the Account's home currency.

created_by_user_id
`int` ID of the user that created the Account.

created_time
DateTime The date/time when the Account was created.

pl
AccountUnits The total profit/loss realized over the lifetime of the Account. Represented in the Account's home currency.

resettable_pl
AccountUnits The total realized profit/loss for the Account since it was last reset by the client. Represented in the Account's home currency.

resettabled_pl_time
DateTime The date/time that the Account's resettablePL was last reset.

commission
AccountUnits The total amount of commission paid over the lifetime of the Account. Represented in the Account's home currency.

margin_rate
DecimalNumber Client-provided margin rate override for the Account. The effective margin rate of the Account is the lesser of this value and the OANDA margin rate for the Account's division. This value is only provided if a margin rate override exists for the Account.

margin_call_enter_time
DateTime The date/time when the Account entered a margin call state. Only provided if the Account is in a margin call.

margin_call_extension_count
`int` The number of times that the Account's current margin call was extended.

last_margin_call_extension_time
DateTime The date/time of the Account's last margin call extension.

open_trade_count
`int` The number of Trades currently open in the Account.

open_position_count
`int` The number of Positions currently open in the Account.

pending_order_count
`int` The number of Orders currently pending in the Account.

hedging_enabled
`bool` Flag indicating that the Account has hedging enabled.

unrealized_pl
AccountUnits The total unrealized profit/loss for all Trades currently open in the Account. Represented in the Account's home currency.

nav
AccountUnits The net asset value of the Account. Equal to Account balance + unrealizedPL. Represented in the Account's home currency.

margin_used
AccountUnits Margin currently used for the Account. Represented in the Account's home currency.

margin_available
AccountUnits Margin available for Account. Represented in the Account's home currency.

position_value
AccountUnits The value of the Account's open positions represented in the Account's home currency.

margin_closeout_unrealized_pl
AccountUnits The Account's margin closeout unrealized PL.

margin_closeout_nav
AccountUnits The Account's margin closeout NAV.

margin_closeout_margin_used
AccountUnits The Account's margin closeout margin used.

margin_closeout_percent
DecimalNumber The Account's margin closeout percentage. When this value is 1.0 or above the Account is in a margin closeout situation.

margin_closeout_position_value
DecimalNumber The value of the Account's open positions as used for margin closeout calculations represented in the Account's home currency.

withdrawal_limit
AccountUnits The current WithdrawalLimit for the account which will be zero or a positive value indicating how much can be withdrawn from the account.

margin_call_margin_used
AccountUnits The Account's margin call margin used.

margin_call_percent
DecimalNumber The Account's margin call percentage. When this value is 1.0 or above the Account is in a margin call situation.

last_transaction_id
TransactionID The ID of the last Transaction created for the Account.

trades
(*TradeSummary*, . . .) The details of the Trades currently open in the Account.

positions
(*Position*, . . .) The details all Account Positions.

orders
(*Order*, . . .) The details of the Orders currently pending in the Account.

dividend
DecimalNumber Dividend

dividendAdjustment
DecimalNumber Undocumented

class `async_v20.`**`AccountChanges`**(*orders_created: ArrayOrder= sentinel*, *orders_cancelled: ArrayOrder= sentinel*, *orders_filled: ArrayOrder= sentinel*, *orders_triggered: ArrayOrder= sentinel*, *trades_opened: ArrayTradeSummary= sentinel*, *trades_reduced: ArrayTradeSummary= sentinel*, *trades_closed: ArrayTradeSummary= sentinel*, *positions: ArrayPosition= sentinel*, *transactions: ArrayTransaction= sentinel*)

Bases: `async_v20.definitions.base.Model`

An AccountChanges Object is used to represent the changes to an Account's Orders, Trades and Positions since a specified Account TransactionID in the past.

`orders_created`
(`Order`, . . .) The Orders created. These Orders may have been filled, cancelled or triggered in the same period.

`orders_cancelled`
(`Order`, . . .) The Orders cancelled.

`orders_filled`
(`Order`, . . .) The Orders filled.

`orders_triggered`
(`Order`, . . .) The Orders triggered.

`trades_opened`
(`TradeSummary`, . . .) The Trades opened.

`trades_reduced`
(`TradeSummary`, . . .) The Trades reduced.

`trades_closed`
(`TradeSummary`, . . .) The Trades closed.

`positions`
(`Position`, . . .) The Positions changed.

`transactions`
(`Transaction`, . . .) The Transactions that have been generated.

8.11.2 Instrument

class `async_v20.`**`Candlestick`**(*time: DateTime= sentinel*, *bid: CandlestickData= sentinel*, *ask: CandlestickData= sentinel*, *mid: CandlestickData= sentinel*, *volume: int= sentinel*, *complete: bool= sentinel*)

Bases: `async_v20.definitions.base.Model`

The Candlestick representation

`time`
`DateTime` The start time of the candlestick

`bid`
`CandlestickData` The candlestick data based on bids. Only provided if bid-based candles were requested.

`ask`
`CandlestickData` The candlestick data based on asks. Only provided if ask-based candles were requested.

mid
`CandlestickData` The candlestick data based on midpoints. Only provided if midpoint-based candles were requested.

volume
`int` The number of prices created during the time-range represented by the candlestick.

complete
`bool` A flag indicating if the candlestick is complete. A complete candlestick is one whose ending time is not in the future.

class async_v20.**CandlestickData**(*o: PriceValue= sentinel*, *h: PriceValue= sentinel*, *l: PriceValue= sentinel*, *c: PriceValue= sentinel*)
Bases: `async_v20.definitions.base.Model`

The price data (open, high, low, close) for the Candlestick representation.

o
`PriceValue` The first (open) price in the time-range represented by the candlestick.

h
`PriceValue` The highest price in the time-range represented by the candlestick.

l
`PriceValue` The lowest price in the time-range represented by the candlestick.

c
`PriceValue` The last (closing) price in the time-range represented by the candlestick.

class async_v20.**OrderBook**(*instrument: InstrumentName= sentinel*, *time: DateTime= sentinel*, *unix_time: DateTime= sentinel*, *price: PriceValue= sentinel*, *bucket_width: PriceValue= sentinel*, *buckets: ArrayOrderBookBucket= sentinel*)
Bases: `async_v20.definitions.base.Model`

The representation of an instrument's order book at a point in time

instrument
`InstrumentName` The order book's instrument

time
`DateTime` The time when the order book snapshot was created.

unix_time
`DateTime` The time when the order book snapshot was created in unix format.

price
`PriceValue` The price (midpoint) for the order book's instrument at the time of the order book snapshot

bucket_width
`PriceValue` The price width for each bucket. Each bucket covers the price range from the bucket's price to the bucket's price + bucketWidth.

buckets
(`OrderBookBucket`, ...) The partitioned order book, divided into buckets using a default bucket width. These buckets are only provided for price ranges which actually contain order or position data.

class async_v20.**OrderBookBucket**(*price: PriceValue= sentinel*, *long_count_percent: DecimalNumber= sentinel*, *short_count_percent: DecimalNumber= sentinel*)
Bases: `async_v20.definitions.base.Model`

The order book data for a partition of the instrument's prices.

price
`PriceValue` The lowest price (inclusive) covered by the bucket. The bucket covers the price range from the price to price + the order book's bucketWidth.

long_count_percent
`DecimalNumber` The percentage of the total number of orders represented by the long orders found in this bucket.

short_count_percent
`DecimalNumber` The percentage of the total number of orders represented by the short orders found in this bucket.

class async_v20.**PositionBook**(*instrument: InstrumentName= sentinel*, *time: DateTime= sentinel*, *unix_time: DateTime= sentinel*, *price: PriceValue= sentinel*, *bucket_width: PriceValue= sentinel*, *buckets: ArrayPositionBookBucket= sentinel*)
Bases: `async_v20.definitions.base.Model`

The representation of an instrument's position book at a point in time

instrument
`InstrumentName` The position book's instrument

time
`DateTime` The time when the position book snapshot was created

price
`PriceValue` The price (midpoint) for the position book's instrument at the time of the position book snapshot

bucket_width
`PriceValue` The price width for each bucket. Each bucket covers the price range from the bucket's price to the bucket's price + bucketWidth.

buckets
(`PositionBookBucket`, ...) The partitioned position book, divided into buckets using a default bucket width. These buckets are only provided for price ranges which actually contain order or position data.

class async_v20.**PositionBookBucket**(*price: PriceValue= sentinel*, *long_count_percent: DecimalNumber= sentinel*, *short_count_percent: DecimalNumber= sentinel*)
Bases: `async_v20.definitions.base.Model`

The position book data for a partition of the instrument's prices.

price
`PriceValue` The lowest price (inclusive) covered by the bucket. The bucket covers the price range from the price to price + the position book's bucketWidth.

long_count_percent
`DecimalNumber` The percentage of the total number of positions represented by the long positions found in this bucket.

short_count_percent
`DecimalNumber` The percentage of the total number of positions represented by the short positions found in this bucket.

8.11.3 Order

class async_v20.**Order**(*id: OrderID= sentinel*, *create_time: DateTime= sentinel*, *state: OrderState= sentinel*, *client_extensions: ClientExtensions= sentinel*, *trade_id: TradeID= sentinel*, *price: PriceValue= sentinel*, *type: OrderType= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition= sentinel*, *filling_transaction_id: TransactionID= sentinel*, *filled_time: DateTime= sentinel*, *trade_opened_id: TradeID= sentinel*, *trade_reduced_id: TradeID= sentinel*, *trade_closed_ids: ArrayTradeID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *cancelled_time: DateTime= sentinel*, *replaces_order_id: OrderID= sentinel*, *replaced_by_order_id: OrderID= sentinel*, *distance: PriceValue= sentinel*, *trailing_stop_value: PriceValue= sentinel*, *instrument: InstrumentName= sentinel*, *units: DecimalNumber= sentinel*, *partial_fill: str= sentinel*, *position_fill: OrderPositionFill= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *price_bound: PriceValue= sentinel*, *initial_market_price: PriceValue= sentinel*, *trade_close: MarketOrderTradeClose= sentinel*, *long_position_closeout: MarketOrderPositionCloseout= sentinel*, *short_position_closeout: MarketOrderPositionCloseout= sentinel*, *margin_closeout: MarketOrderMarginCloseout= sentinel*, *delayed_trade_close: MarketOrderDelayedTradeClose= sentinel*, *trigger_distance: PriceValue= sentinel*, *is_trigger_distance_exact: bool= sentinel*, *guaranteed: bool= sentinel*)

Bases: `async_v20.definitions.base.Model`

The base Order definition. Contains all possible attributes an Order may contain

id
OrderID The Order's identifier, unique within the Order's Account.

create_time
DateTime The time when the Order was created.

state
OrderState The current state of the Order.

client_extensions
ClientExtensions The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

trade_id
TradeID

price
PriceValue

type
OrderType

client_trade_id
ClientID

time_in_force
TimeInForce

gtd_time
DateTime

trigger_condition
    OrderTriggerCondition

filling_transaction_id
    TransactionID

filled_time
    DateTime

trade_opened_id
    TradeID

trade_reduced_id
    TradeID

trade_closed_ids
    (*TradeID*, ...),

cancelling_transaction_id
    TransactionID

cancelled_time
    DateTime

replaces_order_id
    OrderID

replaced_by_order_id
    OrderID

distance
    PriceValue

trailing_stop_value
    PriceValue

instrument
    InstrumentName

units
    DecimalNumber

partial_fill
    str

position_fill
    OrderPositionFill

take_profit_on_fill
    TakeProfitDetails

stop_loss_on_fill
    StopLossDetails

trailing_stop_loss_on_fill
    TrailingStopLossDetails

trade_client_extensions
    ClientExtensions

price_bound
    PriceValue

initial_market_price
    PriceValue

trade_close
    MarketOrderTradeClose

long_position_closeout
    MarketOrderPositionCloseout

short_position_closeout
    MarketOrderPositionCloseout

margin_closeout
    MarketOrderMarginCloseout

delayed_trade_close
    MarketOrderDelayedTradeClose

trigger_distance
    PriceValue

is_trigger_distance_exact
    bool

class async_v20.**MarketOrder**(*instrument: InstrumentName*, *units: DecimalNumber*, *id: OrderID= sentinel*, *create_time: DateTime= sentinel*, *state: OrderState= sentinel*, *client_extensions: ClientExtensions= sentinel*, *time_in_force: TimeInForce=FOK*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trade_close: MarketOrderTradeClose= sentinel*, *long_position_closeout: MarketOrderPositionCloseout= sentinel*, *short_position_closeout: MarketOrderPositionCloseout= sentinel*, *margin_closeout: MarketOrderMarginCloseout= sentinel*, *delayed_trade_close: MarketOrderDelayedTradeClose= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *filling_transaction_id: TransactionID= sentinel*, *filled_time: DateTime= sentinel*, *trade_opened_id: TradeID= sentinel*, *trade_reduced_id: TradeID= sentinel*, *trade_closed_ids: ArrayTradeID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *cancelled_time: DateTime= sentinel*)

Bases: `async_v20.definitions.types.Order`

A MarketOrder is an order that is filled immediately upon creation using the current market price.

instrument
    InstrumentName The Market Order's Instrument.

units
    DecimalNumber The quantity requested to be filled by the Market Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

id
    OrderID The Order's identifier, unique within the Order's Account.

create_time
    DateTime The time when the Order was created.

state
    OrderState The current state of the Order.

client_extensions
ClientExtensions The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

time_in_force
TimeInForce The time-in-force requested for the Market Order. Restricted to FOK or IOC for a MarketOrder.

price_bound
PriceValue The worst price that the client is willing to have the Market Order filled at.

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trade_close
MarketOrderTradeClose Details of the Trade requested to be closed, only provided when the Market Order is being used to explicitly close a Trade.

long_position_closeout
MarketOrderPositionCloseout Details of the long Position requested to be closed out, only provided when a Market Order is being used to explicitly closeout a long Position.

short_position_closeout
MarketOrderPositionCloseout Details of the short Position requested to be closed out, only provided when a Market Order is being used to explicitly closeout a short Position.

margin_closeout
MarketOrderMarginCloseout Details of the Margin Closeout that this Market Order was created for

delayed_trade_close
MarketOrderDelayedTradeClose Details of the delayed Trade close that this Market Order was created for

take_profit_on_fill
TakeProfitDetails TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
StopLossDetails StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
TrailingStopLossDetails TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

filling_transaction_id
TransactionID ID of the Transaction that filled this Order (only provided when the Order's state is FILLED)

filled_time
`DateTime` Date/time when the Order was filled (only provided when the Order's state is FILLED)

trade_opened_id
`TradeID` Trade ID of Trade opened when the Order was filled (only provided when the Order's state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
`TradeID` Trade ID of Trade reduced when the Order was filled (only provided when the Order's state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
(`TradeID`, . . .), Trade IDs of Trades closed when the Order was filled (only provided when the Order's state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
`TransactionID` ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
`DateTime` Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

class async_v20.**LimitOrder**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *id: OrderID= sentinel*, *create_time: DateTime= sentinel*, *state: OrderState= sentinel*, *client_extensions: ClientExtensions= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *filling_transaction_id: TransactionID= sentinel*, *filled_time: DateTime= sentinel*, *trade_opened_id: TradeID= sentinel*, *trade_reduced_id: TradeID= sentinel*, *trade_closed_ids: ArrayTradeID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *cancelled_time: DateTime= sentinel*, *replaces_order_id: OrderID= sentinel*, *replaced_by_order_id: OrderID= sentinel*)
Bases: `async_v20.definitions.types.Order`

A LimitOrder is an order that is created with a price threshold, and will only be filled by a price that is equal to or better than the threshold.

instrument
`InstrumentName` The Limit Order's Instrument.

units
`DecimalNumber` The quantity requested to be filled by the Limit Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
`PriceValue` The price threshold specified for the Limit Order. The Limit Order will only be filled by a market price that is equal to or better than this price.

id
`OrderID` The Order's identifier, unique within the Order's Account.

create_time
`DateTime` The time when the Order was created.

state
OrderState The current state of the Order.

client_extensions
ClientExtensions The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

time_in_force
TimeInForce The time-in-force requested for the Limit Order.

gtd_time
DateTime The date/time when the Limit Order will be cancelled if its timeInForce is “GTD”.

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

take_profit_on_fill
TakeProfitDetails TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade’s dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
StopLossDetails StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade’s dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
TrailingStopLossDetails TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade’s dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

filling_transaction_id
TransactionID ID of the Transaction that filled this Order (only provided when the Order’s state is FILLED)

filled_time
DateTime Date/time when the Order was filled (only provided when the Order’s state is FILLED)

trade_opened_id
TradeID Trade ID of Trade opened when the Order was filled (only provided when the Order’s state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
TradeID Trade ID of Trade reduced when the Order was filled (only provided when the Order’s state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
(*TradeID*, . . .), Trade IDs of Trades closed when the Order was filled (only provided when the Order’s state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
: `TransactionID` ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
: `DateTime` Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

replaces_order_id
: `OrderID` The ID of the Order that was replaced by this Order (only provided if this Order was created as part of a cancel/replace).

replaced_by_order_id
: `OrderID` The ID of the Order that replaced this Order (only provided if this Order was cancelled as part of a cancel/replace).

class async_v20.**StopOrder**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *id: OrderID= sentinel*, *create_time: DateTime= sentinel*, *state: OrderState= sentinel*, *client_extensions: ClientExtensions= sentinel*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *filling_transaction_id: TransactionID= sentinel*, *filled_time: DateTime= sentinel*, *trade_opened_id: TradeID= sentinel*, *trade_reduced_id: TradeID= sentinel*, *trade_closed_ids: ArrayTradeID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *cancelled_time: DateTime= sentinel*, *replaces_order_id: OrderID= sentinel*, *replaced_by_order_id: OrderID= sentinel*)

Bases: `async_v20.definitions.types.Order`

A StopOrder is an order that is created with a price threshold, and will only be filled by a price that is equal to or worse than the threshold.

instrument
: `InstrumentName` The Stop Order's Instrument.

units
: `DecimalNumber` The quantity requested to be filled by the Stop Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
: `PriceValue` The price threshold specified for the Stop Order. The Stop Order will only be filled by a market price that is equal to or worse than this price.

id
: `OrderID` The Order's identifier, unique within the Order's Account.

create_time
: `DateTime` The time when the Order was created.

state
: `OrderState` The current state of the Order.

client_extensions
: `ClientExtensions` The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

price_bound
: *PriceValue* The worst market price that may be used to fill this Stop Order. If the market gaps and crosses through both the price and the priceBound, the Stop Order will be cancelled instead of being filled.

time_in_force
: *TimeInForce* The time-in-force requested for the Stop Order.

gtd_time
: *DateTime* The date/time when the Stop Order will be cancelled if its timeInForce is "GTD".

position_fill
: *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
: *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.

take_profit_on_fill
: *TakeProfitDetails* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
: *StopLossDetails* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
: *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
: *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

filling_transaction_id
: *TransactionID* ID of the Transaction that filled this Order (only provided when the Order's state is FILLED)

filled_time
: *DateTime* Date/time when the Order was filled (only provided when the Order's state is FILLED)

trade_opened_id
: *TradeID* Trade ID of Trade opened when the Order was filled (only provided when the Order's state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
: *TradeID* Trade ID of Trade reduced when the Order was filled (only provided when the Order's state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
: (*TradeID*, ...), Trade IDs of Trades closed when the Order was filled (only provided when the Order's state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
: *TransactionID* ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
`DateTime` Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

replaces_order_id
`OrderID` The ID of the Order that was replaced by this Order (only provided if this Order was created as part of a cancel/replace).

replaced_by_order_id
`OrderID` The ID of the Order that replaced this Order (only provided if this Order was cancelled as part of a cancel/replace).

class async_v20.**MarketIfTouchedOrder**(*instrument: InstrumentName, units: DecimalNumber, price: PriceValue, id: OrderID= sentinel, create_time: DateTime= sentinel, state: OrderState= sentinel, client_extensions: ClientExtensions= sentinel, price_bound: PriceValue= sentinel, time_in_force: TimeInForce=GTC, gtd_time: DateTime= sentinel, position_fill: OrderPositionFill=DEFAULT, trigger_condition: OrderTriggerCondition=DEFAULT, initial_market_price: PriceValue= sentinel, take_profit_on_fill: TakeProfitDetails= sentinel, stop_loss_on_fill: StopLossDetails= sentinel, trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel, trade_client_extensions: ClientExtensions= sentinel, filling_transaction_id: TransactionID= sentinel, filled_time: DateTime= sentinel, trade_opened_id: TradeID= sentinel, trade_reduced_id: TradeID= sentinel, trade_closed_ids: ArrayTradeID= sentinel, cancelling_transaction_id: TransactionID= sentinel, cancelled_time: DateTime= sentinel, replaces_order_id: OrderID= sentinel, replaced_by_order_id: OrderID= sentinel*)
Bases: `async_v20.definitions.types.Order`

A MarketIfTouchedOrder is an order that is created with a price threshold, and will only be filled by a market price that is touches or crosses the threshold.

instrument
`InstrumentName` The MarketIfTouched Order's Instrument.

units
`DecimalNumber` The quantity requested to be filled by the MarketIfTouched Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
`PriceValue` The price threshold specified for the MarketIfTouched Order. The MarketIfTouched Order will only be filled by a market price that crosses this price from the direction of the market price at the time when the Order was created (the initialMarketPrice). Depending on the value of the Order's price and initialMarketPrice, the MarketIfTouchedOrder will behave like a Limit or a Stop Order.

id
`OrderID` The Order's identifier, unique within the Order's Account.

create_time
`DateTime` The time when the Order was created.

state
`OrderState` The current state of the Order.

client_extensions
ClientExtensions The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

price_bound
PriceValue The worst market price that may be used to fill this MarketIfTouched Order.

time_in_force
TimeInForce The time-in-force requested for the MarketIfTouched Order. Restricted to "GTC", "GFD" and "GTD" for MarketIfTouched Orders.

gtd_time
DateTime The date/time when the MarketIfTouched Order will be cancelled if its timeInForce is "GTD".

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

initial_market_price
PriceValue The Market price at the time when the MarketIfTouched Order was created.

take_profit_on_fill
TakeProfitDetails TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
StopLossDetails StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
TrailingStopLossDetails TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

filling_transaction_id
TransactionID ID of the Transaction that filled this Order (only provided when the Order's state is FILLED)

filled_time
DateTime Date/time when the Order was filled (only provided when the Order's state is FILLED)

trade_opened_id
TradeID Trade ID of Trade opened when the Order was filled (only provided when the Order's state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
TradeID Trade ID of Trade reduced when the Order was filled (only provided when the Order's state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
(`TradeID`, . . .), Trade IDs of Trades closed when the Order was filled (only provided when the Order's state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
`TransactionID` ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
`DateTime` Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

replaces_order_id
`OrderID` The ID of the Order that was replaced by this Order (only provided if this Order was created as part of a cancel/replace).

replaced_by_order_id
`OrderID` The ID of the Order that replaced this Order (only provided if this Order was cancelled as part of a cancel/replace).

class async_v20.**TakeProfitOrder**(*trade_id: TradeID*, *price: PriceValue*, *id: OrderID= sentinel*, *create_time: DateTime= sentinel*, *state: OrderState= sentinel*, *client_extensions: ClientExtensions= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *filling_transaction_id: TransactionID= sentinel*, *filled_time: DateTime= sentinel*, *trade_opened_id: TradeID= sentinel*, *trade_reduced_id: TradeID= sentinel*, *trade_closed_ids: ArrayTradeID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *cancelled_time: DateTime= sentinel*, *replaces_order_id: OrderID= sentinel*, *replaced_by_order_id: OrderID= sentinel*)

Bases: `async_v20.definitions.types.Order`

A TakeProfitOrder is an order that is linked to an open Trade and created with a price threshold. The Order will be filled (closing the Trade) by the first price that is equal to or better than the threshold. A TakeProfitOrder cannot be used to open a new Position.

trade_id
`TradeID` The ID of the Trade to close when the price threshold is breached.

price
`PriceValue` The price threshold specified for the TakeProfit Order. The associated Trade will be closed by a market price that is equal to or better than this threshold.

id
`OrderID` The Order's identifier, unique within the Order's Account.

create_time
`DateTime` The time when the Order was created.

state
`OrderState` The current state of the Order.

client_extensions
`ClientExtensions` The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

client_trade_id
`ClientID` The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
: *TimeInForce* The time-in-force requested for the TakeProfit Order. Restricted to "GTC", "GFD" and "GTD" for TakeProfit Orders.

gtd_time
: *DateTime* The date/time when the TakeProfit Order will be cancelled if its timeInForce is "GTD".

trigger_condition
: *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.

filling_transaction_id
: *TransactionID* ID of the Transaction that filled this Order (only provided when the Order's state is FILLED)

filled_time
: *DateTime* Date/time when the Order was filled (only provided when the Order's state is FILLED)

trade_opened_id
: *TradeID* Trade ID of Trade opened when the Order was filled (only provided when the Order's state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
: *TradeID* Trade ID of Trade reduced when the Order was filled (only provided when the Order's state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
: (*TradeID*, ...), Trade IDs of Trades closed when the Order was filled (only provided when the Order's state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
: *TransactionID* ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
: *DateTime* Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

replaces_order_id
: *OrderID* The ID of the Order that was replaced by this Order (only provided if this Order was created as part of a cancel/replace).

replaced_by_order_id
: *OrderID* The ID of the Order that replaced this Order (only provided if this Order was cancelled as part of a cancel/replace).

class async_v20.**StopLossOrder**(*trade_id: TradeID, price: PriceValue, id: OrderID= sentinel, create_time: DateTime= sentinel, state: OrderState= sentinel, client_extensions: ClientExtensions= sentinel, client_trade_id: ClientID= sentinel, time_in_force: TimeInForce=GTC, gtd_time: DateTime= sentinel, trigger_condition: OrderTriggerCondition=DEFAULT, guaranteed: bool= sentinel, filling_transaction_id: TransactionID= sentinel, filled_time: DateTime= sentinel, trade_opened_id: TradeID= sentinel, trade_reduced_id: TradeID= sentinel, trade_closed_ids: ArrayTradeID= sentinel, cancelling_transaction_id: TransactionID= sentinel, cancelled_time: DateTime= sentinel, replaces_order_id: OrderID= sentinel, replaced_by_order_id: OrderID= sentinel*)
: Bases: `async_v20.definitions.types.Order`

A StopLossOrder is an order that is linked to an open Trade and created with a price threshold. The Order will be filled (closing the Trade) by the first price that is equal to or worse than the threshold. A StopLossOrder cannot be used to open a new Position.

:class:`~async_v20.TradeID`
The ID of the Trade to close when the price threshold is breached.

price
PriceValue The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.

id
OrderID The Order's identifier, unique within the Order's Account.

create_time
DateTime The time when the Order was created.

state
OrderState The current state of the Order.

client_extensions
ClientExtensions The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

client_:class:`~async_v20.TradeID`
The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
TimeInForce The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.

gtd_time
DateTime The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

guaranteed

Class *bool*

Flag indicating that the Stop Loss Order is guaranteed. The default value depends on the GuaranteedStopLossOrderMode of the account, if it is REQUIRED, the default will be true, for DISABLED or ENABLED the default is false.

filling_transaction_id
TransactionID ID of the Transaction that filled this Order (only provided when the Order's state is FILLED)

filled_time
DateTime Date/time when the Order was filled (only provided when the Order's state is FILLED)

trade_opened_id
TradeID Trade ID of Trade opened when the Order was filled (only provided when the Order's state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
TradeID Trade ID of Trade reduced when the Order was filled (only provided when the Order's state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
(*TradeID*, . . .), Trade IDs of Trades closed when the Order was filled (only provided when the Order's state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
TransactionID ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
DateTime Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

replaces_order_id
OrderID The ID of the Order that was replaced by this Order (only provided if this Order was created as part of a cancel/replace).

replaced_by_order_id
OrderID The ID of the Order that replaced this Order (only provided if this Order was cancelled as part of a cancel/replace).

class async_v20.**TrailingStopLossOrder**(*trade_id: TradeID, distance: PriceValue, id: OrderID= sentinel, create_time: DateTime= sentinel, state: OrderState= sentinel, client_extensions: ClientExtensions= sentinel, client_trade_id: ClientID= sentinel, time_in_force: TimeInForce=GTC, gtd_time: DateTime= sentinel, trigger_condition: OrderTriggerCondition=DEFAULT, trailing_stop_value: PriceValue= sentinel, filling_transaction_id: TransactionID= sentinel, filled_time: DateTime= sentinel, trade_opened_id: TradeID= sentinel, trade_reduced_id: TradeID= sentinel, trade_closed_ids: ArrayTradeID= sentinel, cancelling_transaction_id: TransactionID= sentinel, cancelled_time: DateTime= sentinel, replaces_order_id: OrderID= sentinel, replaced_by_order_id: OrderID= sentinel*)
Bases: `async_v20.definitions.types.Order`

A TrailingStopLossOrder is an order that is linked to an open Trade and created with a price distance. The price distance is used to calculate a trailing stop value for the order that is in the losing direction from the market price at the time of the order's creation. The trailing stop value will follow the market price as it moves in the winning direction, and the order will filled (closing the Trade) by the first price that is equal to or worse than the trailing stop value. A TrailingStopLossOrder cannot be used to open a new Position.

:class:`~async_v20.TradeID`
The ID of the Trade to close when the price threshold is breached.

distance
PriceValue The price distance specified for the TrailingStopLoss Order.

id
OrderID The Order's identifier, unique within the Order's Account.

create_time
DateTime The time when the Order was created.

state
OrderState The current state of the Order.

client_extensions
ClientExtensions The client extensions of the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

client_:class:`~async_v20.TradeID`
The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
TimeInForce The time-in-force requested for the TrailingStopLoss Order. Restricted to "GTC", "GFD" and "GTD" for TrailingStopLoss Orders.

gtd_time
DateTime The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

trailing_stop_value
PriceValue The trigger price for the Trailing Stop Loss Order. The trailing stop value will trail (follow) the market price by the TSL order's configured "distance" as the market price moves in the winning direction. If the market price moves to a level that is equal to or worse than the trailing stop value, the order will be filled and the Trade will be closed.

filling_transaction_id
TransactionID ID of the Transaction that filled this Order (only provided when the Order's state is FILLED)

filled_time
DateTime Date/time when the Order was filled (only provided when the Order's state is FILLED)

trade_opened_id
TradeID Trade ID of Trade opened when the Order was filled (only provided when the Order's state is FILLED and a Trade was opened as a result of the fill)

trade_reduced_id
TradeID Trade ID of Trade reduced when the Order was filled (only provided when the Order's state is FILLED and a Trade was reduced as a result of the fill)

trade_closed_ids
(*TradeID*, . . .), Trade IDs of Trades closed when the Order was filled (only provided when the Order's state is FILLED and one or more Trades were closed as a result of the fill)

cancelling_transaction_id
TransactionID ID of the Transaction that cancelled the Order (only provided when the Order's state is CANCELLED)

cancelled_time
DateTime Date/time when the Order was cancelled (only provided when the state of the Order is CANCELLED)

replaces_order_id
OrderID The ID of the Order that was replaced by this Order (only provided if this Order was created as part of a cancel/replace).

replaced_by_order_id
OrderID The ID of the Order that replaced this Order (only provided if this Order was cancelled as part of a cancel/replace).

class async_v20.**OrderRequest**(*instrument: InstrumentName*, *trade_id: TradeID= sentinel*, *price: PriceValue= sentinel*, *type: OrderType= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition= sentinel*, *client_extensions: ClientExtensions= sentinel*, *distance: PriceValue= sentinel*, *units: DecimalNumber= sentinel*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.base.Model`

The base Order specification. Contains all attributes an OrderRequest may contain.

instrument
InstrumentName

trade_id
TradeID

price
PriceValue

type
OrderType

client_trade_id
ClientID

time_in_force
TimeInForce

gtd_time
DateTime

trigger_condition
OrderTriggerCondition

client_extensions
ClientExtensions

distance
PriceValue

instrument
InstrumentName

units
DecimalNumber

price_bound
PriceValue

position_fill
OrderPositionFill

take_profit_on_fill
TakeProfitDetails

stop_loss_on_fill
StopLossDetails

trailing_stop_loss_on_fill
: *TrailingStopLossDetails*

trade_client_extensions
: *ClientExtensions*

class async_v20.**MarketOrderRequest**(*instrument: InstrumentName*, *units: DecimalNumber*, *time_in_force: TimeInForce=FOK*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.OrderRequest`

A MarketOrderRequest specifies the parameters that may be set when creating a Market Order.

instrument
: *InstrumentName* The Market Order's Instrument.

units
: *DecimalNumber* The quantity requested to be filled by the Market Order. A positive number of units results in a long Order, and a negative number of units results in a short Order.

time_in_force
: *TimeInForce* The time-in-force requested for the Market Order. Restricted to FOK or IOC for a MarketOrder.

price_bound
: *PriceValue* The worst price that the client is willing to have the Market Order filled at.

position_fill
: *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.

client_extensions
: *ClientExtensions* The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

take_profit_on_fill
: *TakeProfitDetails* TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
: *StopLossDetails* StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
: *TrailingStopLossDetails* TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
: *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

class async_v20.**LimitOrderRequest**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.OrderRequest`

A LimitOrderRequest specifies the parameters that may be set when creating a Limit Order.

instrument
InstrumentName The Limit Order's Instrument.

units
DecimalNumber The quantity requested to be filled by the Limit Order. A positive number of units results in a long Order, and a negative number of units results in a short Order.

price
PriceValue The price threshold specified for the Limit Order. The Limit Order will only be filled by a market price that is equal to or better than this price.

time_in_force
TimeInForce The time-in-force requested for the Limit Order.

gtd_time
DateTime The date/time when the Limit Order will be cancelled if its timeInForce is "GTD".

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

client_extensions
ClientExtensions The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

take_profit_on_fill
TakeProfitDetails TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
StopLossDetails StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
TrailingStopLossDetails TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a

Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

class `async_v20.`**`StopOrderRequest`**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.OrderRequest`

A StopOrderRequest specifies the parameters that may be set when creating a Stop Order.

instrument
`InstrumentName` The Stop Order's Instrument.

units
`DecimalNumber` The quantity requested to be filled by the Stop Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
`PriceValue` The price threshold specified for the Stop Order. The Stop Order will only be filled by a market price that is equal to or worse than this price.

price_bound
`PriceValue` The worst market price that may be used to fill this Stop Order. If the market gaps and crosses through both the price and the priceBound, the Stop Order will be cancelled instead of being filled.

time_in_force
`TimeInForce` The time-in-force requested for the Stop Order.

gtd_time
`DateTime` The date/time when the Stop Order will be cancelled if its timeInForce is "GTD".

position_fill
`OrderPositionFill` Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
`OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.

client_extensions
`ClientExtensions` The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

take_profit_on_fill
`TakeProfitDetails` TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
`StopLossDetails` StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
TrailingStopLossDetails TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

class async_v20.**MarketIfTouchedOrderRequest**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.OrderRequest`

A MarketIfTouchedOrderRequest specifies the parameters that may be set when creating a Market-if-Touched Order.

instrument
InstrumentName The MarketIfTouched Order's Instrument.

units
DecimalNumber The quantity requested to be filled by the MarketIfTouched Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
PriceValue The price threshold specified for the MarketIfTouched Order. The MarketIfTouched Order will only be filled by a market price that crosses this price from the direction of the market price at the time when the Order was created (the initialMarketPrice). Depending on the value of the Order's price and initialMarketPrice, the MarketIfTouchedOrder will behave like a Limit or a Stop Order.

price_bound
PriceValue The worst market price that may be used to fill this MarketIfTouched Order.

time_in_force
TimeInForce The time-in-force requested for the MarketIfTouched Order. Restricted to "GTC", "GFD" and "GTD" for MarketIfTouched Orders.

gtd_time
DateTime The date/time when the MarketIfTouched Order will be cancelled if its timeInForce is "GTD".

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

client_extensions
ClientExtensions The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

take_profit_on_fill
TakeProfitDetails TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

stop_loss_on_fill
StopLossDetails StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

trailing_stop_loss_on_fill
TrailingStopLossDetails TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, or delete tradeClientExtensions if your account is associated with MT4.

class async_v20.**TakeProfitOrderRequest**(*instrument: InstrumentName*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)
Bases: `async_v20.definitions.types.OrderRequest`

A TakeProfitOrderRequest specifies the parameters that may be set when creating a Take Profit Order.

instrument
InstrumentName The TakeProfitOrderRequest instrument.

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

client_trade_id
TradeID The client ID of the Trade to be closed when the price threshold is breached.

price
PriceValue The price threshold specified for the TakeProfit Order. The associated Trade will be closed by a market price that is equal to or better than this threshold.

time_in_force
TimeInForce The time-in-force requested for the TakeProfit Order. Restricted to "GTC", "GFD" and "GTD" for TakeProfit Orders.

gtd_time
DateTime The date/time when the TakeProfit Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

client_extensions

ClientExtensions The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

class async_v20.**StopLossOrderRequest**(*instrument: InstrumentName*, *trade_id: TradeID*, *price: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.OrderRequest`

A StopLossOrderRequest specifies the parameters that may be set when creating a Stop Loss Order.

instrument
InstrumentName The StopLossOrderRequest instrument.

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

client_trade_id
TradeID The client ID of the Trade to be closed when the price threshold is breached.

price
PriceValue The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.

time_in_force
TimeInForce The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.

gtd_time
DateTime The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

client_extensions
ClientExtensions The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

class async_v20.**TrailingStopLossOrderRequest**(*instrument: InstrumentName*, *trade_id: TradeID*, *distance: PriceValue*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.OrderRequest`

A TrailingStopLossOrderRequest specifies the parameters that may be set when creating a Trailing Stop Loss Order.

instrument
InstrumentName The TrailingStopLossOrderRequest instrument

instrument
InstrumentName The TrailingStopLossOrderRequest instrument

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

client_trade_id
: `TradeID` The client ID of the Trade to be closed when the price threshold is breached.

distance
: `PriceValue` The price distance specified for the TrailingStopLoss Order.

time_in_force
: `TimeInForce` The time-in-force requested for the TrailingStopLoss Order. Restricted to "GTC", "GFD" and "GTD" for TrailingStopLoss Orders.

gtd_time
: `DateTime` The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
: `OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.

client_extensions
: `ClientExtensions` The client extensions to add to the Order. Do not set, modify, or delete clientExtensions if your account is associated with MT4.

class `async_v20.`**`DynamicOrderState`**(*id: OrderID= sentinel, trailing_stop_value: PriceValue= sentinel, trigger_distance: PriceValue= sentinel, is_trigger_distance_exact: bool= sentinel*)

Bases: `async_v20.definitions.base.Model`

The dynamic state of an Order. This is only relevant to TrailingStopLoss Orders, as no other Order type has dynamic state.

id
: `OrderID` The Order's ID.

trailing_stop_value
: `PriceValue` The Order's calculated trailing stop value.

trigger_distance
: `PriceValue` The distance between the Trailing Stop Loss Order's trailingStopValue and the current Market Price. This represents the distance (in price units) of the Order from a triggering price. If the distance could not be determined, this value will not be set.

is_trigger_distance_exact
: `bool` True if an exact trigger distance could be calculated. If false, it means the provided trigger distance is a best estimate. If the distance could not be determined, this value will not be set.

class `async_v20.`**`UnitsAvailableDetails`**(*long: DecimalNumber= sentinel, short: DecimalNumber= sentinel*)

Bases: `async_v20.definitions.base.Model`

Representation of how many units of an Instrument are available to be traded for both long and short Orders.

long
: `DecimalNumber` The units available for long Orders.

short
: `DecimalNumber` The units available for short Orders.

class `async_v20.`**`UnitsAvailable`**(*default: UnitsAvailableDetails= sentinel, reduce_first: UnitsAvailableDetails= sentinel, reduce_only: UnitsAvailableDetails= sentinel, open_only: UnitsAvailableDetails= sentinel*)

Bases: `async_v20.definitions.base.Model`

Representation of how many units of an Instrument are available to be traded by an Order depending on its position Fill option.

default
`UnitsAvailableDetails` The number of units that are available to be traded using an Order with a positionFill option of "DEFAULT". For an Account with hedging enabled, this value will be the same as the "OPEN_ONLY" value. For an Account without hedging enabled, this value will be the same as the "REDUCE_FIRST" value.

reduce_first
`UnitsAvailableDetails` The number of units that may are available to be traded with an Order with a positionFill option of "REDUCE_FIRST".

reduce_only
`UnitsAvailableDetails` The number of units that may are available to be traded with an Order with a positionFill option of "REDUCE_ONLY".

open_only
`UnitsAvailableDetails` The number of units that may are available to be traded with an Order with a positionFill option of "OPEN_ONLY".

8.11.4 Positions

class async_v20.**Position**(*instrument: InstrumentName= sentinel*, *pl: AccountUnits= sentinel*, *unrealized_pl: AccountUnits= sentinel*, *resettable_pl: AccountUnits= sentinel*, *commission: AccountUnits= sentinel*, *long: PositionSide= sentinel*, *short: PositionSide= sentinel*, *financing: DecimalNumber= sentinel*, *margin_used: AccountUnits= sentinel*, *guaranteed_execution_fees: AccountUnits= sentinel*)
Bases: `async_v20.definitions.base.Model`

The specification of a Position within an Account.

instrument
`InstrumentName` The Position's Instrument.

pl
`AccountUnits` Profit/loss realized by the Position over the lifetime of the Account.

unrealized_pl
`AccountUnits` The unrealized profit/loss of all open Trades that contribute to this Position.

resettable_pl
`AccountUnits` Profit/loss realized by the Position since the Account's resettablePL was last reset by the client.

commission
`AccountUnits` The total amount of commission paid for this instrument over the lifetime of the Account. Represented in the Account's home currency.

long
`PositionSide` The details of the long side of the Position.

short
`PositionSide` The details of the short side of the Position.

class async_v20.**PositionSide**(*units: DecimalNumber= sentinel*, *average_price: PriceValue= sentinel*, *trade_ids: ArrayTradeID= sentinel*, *pl: AccountUnits= sentinel*, *unrealized_pl: AccountUnits= sentinel*, *resettable_pl: AccountUnits= sentinel*, *financing: DecimalNumber= sentinel*, *guaranteed_execution_fees: AccountUnits= sentinel*)
Bases: `async_v20.definitions.base.Model`

The representation of a Position for a single direction (long or short).

units
`DecimalNumber` Number of units in the position (negative value indicates short position, positive indicates long position).

average_price
`PriceValue` Volume-weighted average of the underlying Trade open prices for the Position.

trade_ids
(`TradeID`, . . .), List of the open Trade IDs which contribute to the open Position.

pl
`AccountUnits` Profit/loss realized by the PositionSide over the lifetime of the Account.

unrealized_pl
`AccountUnits` The unrealized profit/loss of all open Trades that contribute to this PositionSide.

resettable_pl
`AccountUnits` Profit/loss realized by the PositionSide since the Account's resettablePL was last reset by the client.

class async_v20.**CalculatedPositionState**(*instrument: InstrumentName= sentinel, net_unrealized_pl: AccountUnits= sentinel, long_unrealized_pl: AccountUnits= sentinel, short_unrealized_pl: AccountUnits= sentinel, margin_used: AccountUnits= sentinel*)
Bases: `async_v20.definitions.base.Model`

The dynamic (calculated) state of a Position

instrument
`InstrumentName` The Position's Instrument.

net_unrealized_pl
`AccountUnits` The Position's net unrealized profit/loss

long_unrealized_pl
`AccountUnits` The unrealized profit/loss of the Position's long open Trades

short_unrealized_pl
`AccountUnits` The unrealized profit/loss of the Position's short open Trades

Margin_used
Margin currently used by the Position

8.11.5 Pricing

class async_v20.**Price**(*type: str= sentinel, instrument: InstrumentName= sentinel, time: DateTime= sentinel, status: PriceStatus= sentinel, tradeable: bool= sentinel, bids: ArrayPriceBucket= sentinel, asks: ArrayPriceBucket= sentinel, closeout_bid: PriceValue= sentinel, closeout_ask: PriceValue= sentinel, quote_home_conversion_factors: QuoteHomeConversionFactors= sentinel, units_available: UnitsAvailable= sentinel*)
Bases: `async_v20.definitions.base.Model`

The specification of an Account-specific Price.

type
`str` The string "PRICE". Used to identify the a Price object when found in a stream.

instrument
: `InstrumentName` The Price's Instrument.

time
: `DateTime` The date/time when the Price was created

status
: `PriceStatus` The status of the Price.

tradeable
: `bool` Flag indicating if the Price is tradeable or not

bids
: (`PriceBucket`, . . .), The list of prices and liquidity available on the Instrument's bid side. It is possible for this list to be empty if there is no bid liquidity currently available for the Instrument in the Account.

asks
: (`PriceBucket`, . . .), The list of prices and liquidity available on the Instrument's ask side. It is possible for this list to be empty if there is no ask liquidity currently available for the Instrument in the Account.

closeout_bid
: `PriceValue` The closeout bid Price. This Price is used when a bid is required to closeout a Position (margin closeout or manual) yet there is no bid liquidity. The closeout bid is never used to open a new position.

closeout_ask
: `PriceValue` The closeout ask Price. This Price is used when a ask is required to closeout a Position (margin closeout or manual) yet there is no ask liquidity. The closeout ask is never used to open a new position.

quote_home_conversion_factors
: `QuoteHomeConversionFactors` The factors used to convert quantities of this price's Instrument's quote currency into a quantity of the Account's home currency.

units_available
: `UnitsAvailable` Representation of how many units of an Instrument are available to be traded by an Order depending on its postionFill option.

class `async_v20.`**PriceBucket**(*price: PriceValue= sentinel*, *liquidity: int= sentinel*)
Bases: `async_v20.definitions.base.Model`

A Price Bucket represents a price available for an amount of liquidity

price
: `PriceValue` The Price offered by the PriceBucket

liquidity
: `int` The amount of liquidity offered by the PriceBucket

class `async_v20.`**QuoteHomeConversionFactors**(*positive_units: DecimalNumber= sentinel*, *negative_units: DecimalNumber= sentinel*)
Bases: `async_v20.definitions.base.Model`

QuoteHomeConversionFactors represents the factors that can be used used to convert quantities of a Price's Instrument's quote currency into the Account's home currency.

positive_units
: `DecimalNumber` The factor used to convert a positive amount of the Price's Instrument's quote currency into a positive amount of the Account's home currency. Conversion is performed by multiplying the quote units by the conversion factor.

negative_units
: `DecimalNumber` The factor used to convert a negative amount of the Price's Instrument's quote currency

into a negative amount of the Account's home currency. Conversion is performed by multiplying the quote units by the conversion factor.

class `async_v20.`**`ClientPrice`**(*bids: ArrayPriceBucket= sentinel*, *asks: ArrayPriceBucket= sentinel*, *closeout_bid: PriceValue= sentinel*, *closeout_ask: PriceValue= sentinel*, *timestamp: DateTime= sentinel*)

Bases: `async_v20.definitions.base.Model`

Client price for an Account.

`bids`

(`PriceBucket`, . . .), The list of prices and liquidity available on the Instrument's bid side. It is possible for this list to be empty if there is no bid liquidity currently available for the Instrument in the Account.

`asks`

(`PriceBucket`, . . .), The list of prices and liquidity available on the Instrument's ask side. It is possible for this list to be empty if there is no ask liquidity currently available for the Instrument in the Account.

`closeout_bid`

`PriceValue` The closeout bid Price. This Price is used when a bid is required to closeout a Position (margin closeout or manual) yet there is no bid liquidity. The closeout bid is never used to open a new position.

`closeout_ask`

`PriceValue` The closeout ask Price. This Price is used when a ask is required to closeout a Position (margin closeout or manual) yet there is no ask liquidity. The closeout ask is never used to open a new position.

`timestamp`

`DateTime` The date/time when the Price was created.

class `async_v20.`**`PricingHeartbeat`**(*type: str= sentinel*, *time: DateTime= sentinel*)

Bases: `async_v20.definitions.base.Model`

A PricingHeartbeat object is injected into the Pricing stream to ensure that the HTTP connection remains active.

`type`

`str` The string "HEARTBEAT"

`time`

`DateTime` The date/time when the Heartbeat was created.

8.11.6 Trade

class `async_v20.`**`Trade`**(*id: TradeID= sentinel*, *instrument: InstrumentName= sentinel*, *price: PriceValue= sentinel*, *open_time: DateTime= sentinel*, *state: TradeState= sentinel*, *initial_units: DecimalNumber= sentinel*, *initial_margin_required: AccountUnits= sentinel*, *current_units: DecimalNumber= sentinel*, *realized_pl: AccountUnits= sentinel*, *unrealized_pl: AccountUnits= sentinel*, *average_close_price: PriceValue= sentinel*, *closing_transaction_ids: ArrayTransactionID= sentinel*, *financing: AccountUnits= sentinel*, *close_time: DateTime= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_order: TakeProfitOrder= sentinel*, *stop_loss_order: StopLossOrder= sentinel*, *trailing_stop_loss_order: TrailingStopLossOrder= sentinel*, *margin_used: AccountUnits= sentinel*)

Bases: `async_v20.definitions.base.Model`

The specification of a Trade within an Account. This includes the full representation of the Trade's dependent Orders in addition to the IDs of those Orders.

id
TradeID The Trade's identifier, unique within the Trade's Account.

instrument
InstrumentName The Trade's Instrument.

price
PriceValue The execution price of the Trade.

open_time
DateTime The date/time when the Trade was opened.

state
TradeState The current state of the Trade.

initial_units
DecimalNumber The initial size of the Trade. Negative values indicate a short Trade, and positive values indicate a long Trade.

initial_margin_required
AccountUnits The margin required at the time the Trade was created. Note, this is the 'pure' margin required, it is not the 'effective' margin used that factors in the trade risk if a GSLO is attached to the trade.

current_units
DecimalNumber The number of units currently open for the Trade. This value is reduced to 0.0 as the Trade is closed.

realized_pl
AccountUnits The total profit/loss realized on the closed portion of the Trade.

unrealized_pl
AccountUnits The unrealized profit/loss on the open portion of the Trade.

average_close_price
PriceValue The average closing price of the Trade. Only present if the Trade has been closed or reduced at least once.

closing_transaction_ids
(*TransactionID*, ...) The IDs of the Transactions that have closed portions of this Trade.

financing
AccountUnits The financing paid/collected for this Trade.

close_time
DateTime The date/time when the Trade was fully closed. Only provided for Trades whose state is CLOSED.

client_extensions
ClientExtensions The client extensions of the Trade.

take_profit_order
TakeProfitOrder Full representation of the Trade's Take Profit Order, only provided if such an Order exists.

stop_loss_order
StopLossOrder Full representation of the Trade's Stop Loss Order, only provided if such an Order exists.

trailing_stop_loss_order
TrailingStopLossOrder Full representation of the Trade's Trailing Stop Loss Order, only provided if such an Order exists.

margin_used
Margin currently used by the Trade.

class async_v20.**TradeSummary**(*id: TradeID= sentinel, instrument: InstrumentName= sentinel, price: PriceValue= sentinel, open_time: DateTime= sentinel, state: TradeState= sentinel, initial_units: DecimalNumber= sentinel, initial_margin_required: AccountUnits= sentinel, current_units: DecimalNumber= sentinel, realized_pl: AccountUnits= sentinel, unrealized_pl: AccountUnits= sentinel, average_close_price: PriceValue= sentinel, closing_transaction_ids: ArrayTransactionID= sentinel, financing: AccountUnits= sentinel, close_time: DateTime= sentinel, client_extensions: ClientExtensions= sentinel, take_profit_order_id: OrderID= sentinel, stop_loss_order_id: OrderID= sentinel, trailing_stop_loss_order_id: OrderID= sentinel, margin_used: AccountUnits= sentinel*)

Bases: `async_v20.definitions.base.Model`

The summary of a Trade within an Account. This representation does not provide the full details of the Trade's dependent Orders.

id
TradeID The Trade's identifier, unique within the Trade's Account.

instrument
InstrumentName The Trade's Instrument.

price
PriceValue The execution price of the Trade.

open_time
DateTime The date/time when the Trade was opened.

state
TradeState The current state of the Trade.

initial_units
DecimalNumber The initial size of the Trade. Negative values indicate a short Trade, and positive values indicate a long Trade.

initial_margin_required
AccountUnits The margin required at the time the Trade was created. Note, this is the 'pure' margin required, it is not the 'effective' margin used that factors in the trade risk if a GSLO is attached to the trade.

current_units
DecimalNumber The number of units currently open for the Trade. This value is reduced to 0.0 as the Trade is closed.

realized_pl
AccountUnits The total profit/loss realized on the closed portion of the Trade.

unrealized_pl
AccountUnits The unrealized profit/loss on the open portion of the Trade.

average_close_price
PriceValue The average closing price of the Trade. Only present if the Trade has been closed or reduced at least once.

closing_transaction_ids
(*TransactionID*, . . .) The IDs of the Transactions that have closed portions of this Trade.

financing
AccountUnits The financing paid/collected for this Trade.

close_time
DateTime The date/time when the Trade was fully closed. Only provided for Trades whose state is CLOSED.

client_extensions
ClientExtensions The client extensions of the Trade.

take_profit_order_id
OrderID ID of the Trade's Take Profit Order, only provided if such an Order exists.

stop_loss_order_id
OrderID ID of the Trade's Stop Loss Order, only provided if such an Order exists.

trailing_stop_loss_order_id
OrderID ID of the Trade's Trailing Stop Loss Order, only provided if such an Order exists.

margin_used
Margin currently used by the Trade.

class async_v20.**CalculatedTradeState**(*id: TradeID= sentinel*, *unrealized_pl: AccountUnits= sentinel*, *margin_used: AccountUnits= sentinel*)

Bases: `async_v20.definitions.base.Model`

The dynamic (calculated) state of an open Trade

id
TradeID The Trade's ID.

unrealized_pl
AccountUnits The Trade's unrealized profit/loss.

8.11.7 Transaction

class async_v20.**Transaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *type: TransactionType= sentinel*, *extension_number: int= sentinel*, *division_id: int= sentinel*, *site_id: int= sentinel*, *account_user_id: int= sentinel*, *account_number: int= sentinel*, *home_currency: Currency= sentinel*, *alias: str= sentinel*, *margin_rate: DecimalNumber= sentinel*, *reason: Reason= sentinel*, *trade_ids: TradeID= sentinel*, *order_id: OrderID= sentinel*, *client_order_id: ClientID= sentinel*, *replaced_by_order_id: OrderID= sentinel*, *closed_trade_id: OrderID= sentinel*, *trade_close_transaction_id: TransactionID= sentinel*, *client_extensions_modify: ClientExtensions= sentinel*, *trade_client_extensions_modify: ClientExtensions= sentinel*, *financing: AccountUnits= sentinel*, *account_balance: AccountUnits= sentinel*, *account_financing_mode: AccountFinancingMode= sentinel*, *position_financings: ArrayPositionFinancing= sentinel*, *trade_id: TradeID= sentinel*, *client_trade_id: ClientID= sentinel*, *price: PriceValue= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *reject_reason: TransactionRejectReason= sentinel*, *amount: AccountUnits= sentinel*, *funding_reason: FundingReason= sentinel*, *comment: str= sentinel*, *instrument: InstrumentName= sentinel*, *units: DecimalNumber= sentinel*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill= sentinel*, *trade_close: MarketOrderTradeClose= sentinel*, *long_position_closeout: MarketOrderPositionCloseout= sentinel*, *short_position_closeout: MarketOrderPositionCloseout= sentinel*, *margin_closeout: MarketOrderMarginCloseout= sentinel*, *delayed_trade_close: MarketOrderDelayedTradeClose= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *distance: PriceValue= sentinel*, *full_price: ClientPrice= sentinel*, *pl: AccountUnits= sentinel*, *commission: AccountUnits= sentinel*, *trade_opened: TradeOpen= sentinel*, *trades_closed: ArrayTradeReduce= sentinel*, *trade_reduced: TradeReduce= sentinel*, *intended_replaces_order_id: OrderID= sentinel*, *gain_quote_home_conversion_factor: DecimalNumber= sentinel*, *loss_quote_home_conversion_factor: DecimalNumber= sentinel*, *guaranteed_execution_fee: AccountUnits= sentinel*, *half_spread_cost: AccountUnits= sentinel*, *partial_fill: str= sentinel*, *guaranteed: bool= sentinel*, *requested_units: AccountUnits= sentinel*, *full_vwap: DecimalNumber= sentinel*)

Bases: `async_v20.definitions.base.Model`

The base Transaction specification. Contains all possible attributes a transaction may contain.

id

TransactionID The Transaction's Identifier.

time

`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

class async_v20.**CreateTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, division_id: int= sentinel, site_id: int= sentinel, account_user_id: int= sentinel, account_number: int= sentinel, home_currency: Currency= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A CreateTransaction represents the creation of an Account.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

division_id
`TransactionID` The ID of the Division that the Account is in

site_id
`TransactionID` The ID of the Site that the Account was created at

account_user_id
`int` The ID of the user that the Account was created for

account_number
`int` The number of the Account within the site/division/user

home_currency
`Currency` The home currency of the Account

class async_v20.**CloseTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A CloseTransaction represents the closing of an Account.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

class async_v20.**ReopenTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A ReopenTransaction represents the re-opening of a closed Account.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

class async_v20.**ClientConfigureTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *alias: str= sentinel*, *margin_rate: DecimalNumber= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A ClientConfigureTransaction represents the configuration of an Account by a client.

id
TransactionID The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

alias
`str` The client-provided alias for the Account.

margin_rate
`DecimalNumber` The margin rate override for the Account.

class async_v20.**ClientConfigureRejectTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *alias: str= sentinel*, *margin_rate: DecimalNumber= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A ClientConfigureRejectTransaction represents the reject of configuration of an Account by a client.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

alias
`str` The client-provided alias for the Account.

margin_rate
`DecimalNumber` The margin rate override for the Account.

reject_reason
`TransactionRejectReason` The reason that the Reject Transaction was created

class async_v20.**TransferFundsTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *amount: AccountUnits= sentinel*, *funding_reason: FundingReason= sentinel*, *comment: str= sentinel*, *account_balance: AccountUnits= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TransferFundsTransaction represents the transfer of funds in/out of an Account.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

amount
AccountUnits The amount to deposit/withdraw from the Account in the Account's home currency. A positive value indicates a deposit, a negative value indicates a withdrawal.

funding_reason
FundingReason The reason that an Account is being funded.

comment
`str` An optional comment that may be attached to a fund transfer for audit purposes

account_balance
AccountUnits The Account's balance after funds are transferred.

class async_v20.**TransferFundsRejectTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *amount: AccountUnits= sentinel*, *funding_reason: FundingReason= sentinel*, *comment: str= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TransferFundsRejectTransaction represents the rejection of the transfer of funds in/out of an Account.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

amount
`AccountUnits` The amount to deposit/withdraw from the Account in the Account's home currency. A positive value indicates a deposit, a negative value indicates a withdrawal.

funding_reason
`FundingReason` The reason that an Account is being funded.

comment
`str` An optional comment that may be attached to a fund transfer for audit purposes

reject_reason
`TransactionRejectReason` The reason that the Reject Transaction was created

class async_v20.**MarketOrderTransaction**(*instrument: InstrumentName*, *units: DecimalNumber*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *time_in_force: TimeInForce=FOK*, *price_bound: PriceValue= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trade_close: MarketOrderTradeClose= sentinel*, *long_position_closeout: MarketOrderPositionCloseout= sentinel*, *short_position_closeout: MarketOrderPositionCloseout= sentinel*, *margin_closeout: MarketOrderMarginCloseout= sentinel*, *delayed_trade_close: MarketOrderDelayedTradeClose= sentinel*, *reason: MarketOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarketOrderTransaction represents the creation of a Market Order in the user's account. A Market Order is an Order that is filled immediately at the current market price. Market Orders can be specialized when they are created to accomplish a specific tas': 'to' close a Trade, to closeout a Position or to particiate in in a Margin closeout.

instrument
`InstrumentName` The Market Order's Instrument.

id
`TransactionID` The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id
: *TransactionID* The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: *RequestID* The Request ID of the request which generated the transaction.

units
: *DecimalNumber* The quantity requested to be filled by the Market Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

time_in_force
: *TimeInForce* The time-in-force requested for the Market Order. Restricted to FOK or IOC for a MarketOrder.

price_bound
: *PriceValue* The worst price that the client is willing to have the Market Order filled at.

position_fill
: *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.

trade_close
: *MarketOrderTradeClose* Details of the Trade requested to be closed, only provided when the Market Order is being used to explicitly close a Trade.

long_position_closeout
: *MarketOrderPositionCloseout* Details of the long Position requested to be closed out, only provided when a Market Order is being used to explicitly closeout a long Position.

short_position_closeout
: *MarketOrderPositionCloseout* Details of the short Position requested to be closed out, only provided when a Market Order is being used to explicitly closeout a short Position.

margin_closeout
: *MarketOrderMarginCloseout* Details of the Margin Closeout that this Market Order was created for

delayed_trade_close
: *MarketOrderDelayedTradeClose* Details of the delayed Trade close that this Market Order was created for

reason
: *MarketOrderReason* The reason that the Market Order was created

client_extensions
: *ClientExtensions* Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
: *TakeProfitDetails* The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
: *StopLossDetails* The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
: *TrailingStopLossDetails* The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
: *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

class async_v20.**MarketOrderRejectTransaction**(*instrument: InstrumentName= sentinel, units: DecimalNumber= sentinel, id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, time_in_force: TimeInForce=FOK, price_bound: PriceValue= sentinel, position_fill: OrderPositionFill=DEFAULT, trade_close: MarketOrderTradeClose= sentinel, long_position_closeout: MarketOrderPositionCloseout= sentinel, short_position_closeout: MarketOrderPositionCloseout= sentinel, margin_closeout: MarketOrderMarginCloseout= sentinel, delayed_trade_close: MarketOrderDelayedTradeClose= sentinel, reason: MarketOrderReason= sentinel, client_extensions: ClientExtensions= sentinel, take_profit_on_fill: TakeProfitDetails= sentinel, stop_loss_on_fill: StopLossDetails= sentinel, trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel, trade_client_extensions: ClientExtensions= sentinel, reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarketOrderRejectTransaction represents the rejection of the creation of a Market Order.

instrument
: *InstrumentName* The Market Order's Instrument.

id
: *TransactionID* The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id

TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

units
DecimalNumber The quantity requested to be filled by the Market Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

time_in_force
TimeInForce The time-in-force requested for the Market Order. Restricted to FOK or IOC for a MarketOrder.

price_bound
PriceValue The worst price that the client is willing to have the Market Order filled at.

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trade_close
MarketOrderTradeClose Details of the Trade requested to be closed, only provided when the Market Order is being used to explicitly close a Trade.

long_position_closeout
MarketOrderPositionCloseout Details of the long Position requested to be closed out, only provided when a Market Order is being used to explicitly closeout a long Position.

short_position_closeout
MarketOrderPositionCloseout Details of the short Position requested to be closed out, only provided when a Market Order is being used to explicitly closeout a short Position.

margin_closeout
MarketOrderMarginCloseout Details of the Margin Closeout that this Market Order was created for

delayed_trade_close
MarketOrderDelayedTradeClose Details of the delayed Trade close that this Market Order was created for

reason
MarketOrderReason The reason that the Market Order was created

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
TakeProfitDetails The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
StopLossDetails The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
TrailingStopLossDetails The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
: *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

reject_reason
: *TransactionRejectReason* The reason that the Reject Transaction was created

class async_v20.**LimitOrderTransaction**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: LimitOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *partial_fill: OrderPositionFill= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A LimitOrderTransaction represents the creation of a Limit Order in the user's Account.

instrument
: *InstrumentName* The Limit Order's Instrument.

units
: *DecimalNumber* The quantity requested to be filled by the Limit Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
: *PriceValue* The price threshold specified for the Limit Order. The Limit Order will only be filled by a market price that is equal to or better than this price.

id
: *TransactionID* The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id
: *TransactionID* The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: *RequestID* The Request ID of the request which generated the transaction.

time_in_force
: *TimeInForce* The time-in-force requested for the Limit Order.

gtd_time
DateTime The date/time when the Limit Order will be cancelled if its timeInForce is “GTD”.

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
LimitOrderReason The reason that the Limit Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
TakeProfitDetails The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
StopLossDetails The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
TrailingStopLossDetails The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

replaces_order_id
OrderID The ID of the Order that this Order replaces (only provided if this Order replaces an existing Order).

cancelling_transaction_id
TransactionID The ID of the Transaction that cancels the replaced Order (only provided if this Order replaces an existing Order).

partial_fill
= positionFill (seems to be a mismatch in Oanda documentation)

class async_v20.**LimitOrderRejectTransaction**(*instrument: InstrumentName= sentinel*, *units: DecimalNumber= sentinel*, *price: PriceValue= sentinel*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: LimitOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *intended_replaces_order_id: OrderID= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A LimitOrderRejectTransaction represents the rejection of the creation of a Limit Order.

instrument
: `InstrumentName` The Limit Order's Instrument.

units
: `DecimalNumber` The quantity requested to be filled by the Limit Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
: `PriceValue` The price threshold specified for the Limit Order. The Limit Order will only be filled by a market price that is equal to or better than this price.

id
: `TransactionID` The Transaction's Identifier.

time
: `DateTime` The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: `AccountID` The ID of the Account the Transaction was created for.

batch_id
: `TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: `RequestID` The Request ID of the request which generated the transaction.

time_in_force
: `TimeInForce` The time-in-force requested for the Limit Order.

gtd_time
: `DateTime` The date/time when the Limit Order will be cancelled if its timeInForce is "GTD".

position_fill
: *OrderPositionFill* Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
: *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
: *LimitOrderReason* The reason that the Limit Order was initiated

client_extensions
: *ClientExtensions* Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
: *TakeProfitDetails* The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
: *StopLossDetails* The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
: *TrailingStopLossDetails* The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
: *ClientExtensions* Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

intended_replaces_order_id
: *OrderID* The ID of the Order that this Order was intended to replace (only provided if this Order was intended to replace an existing Order).

reject_reason
: *TransactionRejectReason* The reason that the Reject Transaction was created

class async_v20.**StopOrderTransaction**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *partial_fill: str= sentinel*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: StopOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A StopOrderTransaction represents the creation of a Stop Order in the user's Account.

instrument
InstrumentName The Stop Order's Instrument.

units
DecimalNumber The quantity requested to be filled by the Stop Order. A positive number of units results in a long Order, and a negative number of units results in a short Order.

price
PriceValue The price threshold specified for the Stop Order. The Stop Order will only be filled by a market price that is equal to or worse than this price.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

price_bound
PriceValue The worst market price that may be used to fill this Stop Order. If the market gaps and crosses through both the price and the priceBound, the Stop Order will be cancelled instead of being filled.

time_in_force
TimeInForce The time-in-force requested for the Stop Order.

gtd_time
DateTime The date/time when the Stop Order will be cancelled if its timeInForce is "GTD".

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
StopOrderReason The reason that the Stop Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
TakeProfitDetails The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
StopLossDetails The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
TrailingStopLossDetails The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

replaces_order_id
OrderID The ID of the Order that this Order replaces (only provided if this Order replaces an existing Order).

cancelling_transaction_id
TransactionID The ID of the Transaction that cancels the replaced Order (only provided if this Order replaces an existing Order).

class async_v20.**StopOrderRejectTransaction**(*instrument: InstrumentName= sentinel*, *units: DecimalNumber= sentinel*, *price: PriceValue= sentinel*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: StopOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *intended_replaces_order_id: OrderID= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A StopOrderRejectTransaction represents the rejection of the creation of a Stop Order.

instrument
InstrumentName The Stop Order's Instrument.

units
DecimalNumber The quantity requested to be filled by the Stop Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
PriceValue The price threshold specified for the Stop Order. The Stop Order will only be filled by a market price that is equal to or worse than this price.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

price_bound
PriceValue The worst market price that may be used to fill this Stop Order. If the market gaps and crosses through both the price and the priceBound, the Stop Order will be cancelled instead of being filled.

time_in_force
TimeInForce The time-in-force requested for the Stop Order.

gtd_time
DateTime The date/time when the Stop Order will be cancelled if its timeInForce is "GTD".

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
StopOrderReason The reason that the Stop Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
TakeProfitDetails The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
StopLossDetails The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
TrailingStopLossDetails The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

intended_replaces_order_id
OrderID The ID of the Order that this Order was intended to replace (only provided if this Order was intended to replace an existing Order).

reject_reason
TransactionRejectReason The reason that the Reject Transaction was created

class async_v20.**MarketIfTouchedOrderTransaction**(*instrument: InstrumentName*, *units: DecimalNumber*, *price: PriceValue*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *price_bound: PriceValue= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *position_fill: OrderPositionFill=DEFAULT*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: MarketIfTouchedOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *take_profit_on_fill: TakeProfitDetails= sentinel*, *stop_loss_on_fill: StopLossDetails= sentinel*, *trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel*, *trade_client_extensions: ClientExtensions= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarketIfTouchedOrderTransaction represents the creation of a MarketIfTouched Order in the user's Account.

instrument
`InstrumentName` The MarketIfTouched Order's Instrument.

units
`DecimalNumber` The quantity requested to be filled by the MarketIfTouched Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price
`PriceValue` The price threshold specified for the MarketIfTouched Order. The MarketIfTouched Order will only be filled by a market price that crosses this price from the direction of the market price at the time when the Order was created (the initialMarketPrice). Depending on the value of the Order's price and initialMarketPrice, the MarketIfTouchedOrder will behave like a Limit or a Stop Order.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

price_bound
PriceValue The worst market price that may be used to fill this MarketIfTouched Order.

time_in_force
TimeInForce The time-in-force requested for the MarketIfTouched Order. Restricted to “GTC”, “GFD” and “GTD” for MarketIfTouched Orders.

gtd_time
DateTime The date/time when the MarketIfTouched Order will be cancelled if its timeInForce is “GTD”.

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
MarketIfTouchedOrderReason The reason that the Market-if-touched Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
TakeProfitDetails The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
StopLossDetails The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
TrailingStopLossDetails The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

replaces_order_id
OrderID The ID of the Order that this Order replaces (only provided if this Order replaces an existing Order).

cancelling_transaction_id
TransactionID The ID of the Transaction that cancels the replaced Order (only provided if this Order replaces an existing Order).

class async_v20.**MarketIfTouchedOrderRejectTransaction**(*instrument: InstrumentName= sentinel, units: DecimalNumber= sentinel, price: PriceValue= sentinel, id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, price_bound: PriceValue= sentinel, time_in_force: TimeInForce=GTC, gtd_time: DateTime= sentinel, position_fill: OrderPositionFill=DEFAULT, trigger_condition: OrderTriggerCondition=DEFAULT, reason: MarketIfTouchedOrderReason= sentinel, client_extensions: ClientExtensions= sentinel, take_profit_on_fill: TakeProfitDetails= sentinel, stop_loss_on_fill: StopLossDetails= sentinel, trailing_stop_loss_on_fill: TrailingStopLossDetails= sentinel, trade_client_extensions: ClientExtensions= sentinel, intended_replaces_order_id: OrderID= sentinel, reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarketIfTouchedOrderRejectTransaction represents the rejection of the creation of a MarketIfTouched Order.

instrument

InstrumentName The MarketIfTouched Order's Instrument.

units

DecimalNumber The quantity requested to be filled by the MarketIfTouched Order. A posititive number of units results in a long Order, and a negative number of units results in a short Order.

price

PriceValue The price threshold specified for the MarketIfTouched Order. The MarketIfTouched Order will only be filled by a market price that crosses this price from the direction of the market price at the time when the Order was created (the initialMarketPrice). Depending on the value of the Order's price and initialMarketPrice, the MarketIfTouchedOrder will behave like a Limit or a Stop Order.

id

TransactionID The Transaction's Identifier.

time

DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

price_bound
PriceValue The worst market price that may be used to fill this MarketIfTouched Order.

time_in_force
TimeInForce The time-in-force requested for the MarketIfTouched Order. Restricted to "GTC", "GFD" and "GTD" for MarketIfTouched Orders.

gtd_time
DateTime The date/time when the MarketIfTouched Order will be cancelled if its timeInForce is "GTD".

position_fill
OrderPositionFill Specification of how Positions in the Account are modified when the Order is filled.

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
MarketIfTouchedOrderReason The reason that the Market-if-touched Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

take_profit_on_fill
TakeProfitDetails The specification of the Take Profit Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

stop_loss_on_fill
StopLossDetails The specification of the Stop Loss Order that should be created for a Trade opened when the Order is filled (if such a Trade is created).

trailing_stop_loss_on_fill
TrailingStopLossDetails The specification of the Trailing Stop Loss Order that should be created for a Trade that is opened when the Order is filled (if such a Trade is created).

trade_client_extensions
ClientExtensions Client Extensions to add to the Trade created when the Order is filled (if such a Trade is created). Do not set, modify, delete tradeClientExtensions if your account is associated with MT4.

intended_replaces_order_id
OrderID The ID of the Order that this Order was intended to replace (only provided if this Order was intended to replace an existing Order).

reject_reason
TransactionRejectReason The reason that the Reject Transaction was created

class async_v20.**TakeProfitOrderTransaction**(*trade_id: TradeID*, *price: PriceValue*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: TakeProfitOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TakeProfitOrderTransaction represents the creation of a TakeProfit Order in the user's Account.

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

price
PriceValue The price threshold specified for the TakeProfit Order. The associated Trade will be closed by a market price that is equal to or better than this threshold.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

client_trade_id
ClientID The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
TimeInForce The time-in-force requested for the TakeProfit Order. Restricted to "GTC", "GFD" and "GTD" for TakeProfit Orders.

gtd_time
DateTime The date/time when the TakeProfit Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
TakeProfitOrderReason The reason that the Take Profit Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

order_fill_transaction_id
TransactionID The ID of the OrderFill Transaction that caused this Order to be created (only provided if this Order was created automatically when another Order was filled).

replaces_order_id
OrderID The ID of the Order that this Order replaces (only provided if this Order replaces an existing Order).

cancelling_transaction_id
TransactionID The ID of the Transaction that cancels the replaced Order (only provided if this Order replaces an existing Order).

class async_v20.**TakeProfitOrderRejectTransaction**(*trade_id: TradeID= sentinel*, *price: PriceValue= sentinel*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: TakeProfitOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *intended_replaces_order_id: OrderID= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TakeProfitOrderRejectTransaction represents the rejection of the creation of a TakeProfit Order.

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

price
PriceValue The price threshold specified for the TakeProfit Order. The associated Trade will be closed by a market price that is equal to or better than this threshold.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch

are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

client_trade_id
`TradeID` The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
`TimeInForce` The time-in-force requested for the TakeProfit Order. Restricted to "GTC", "GFD" and "GTD" for TakeProfit Orders.

gtd_time
`DateTime` The date/time when the TakeProfit Order will be cancelled if its timeInForce is "GTD".

trigger_condition
`OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
`TakeProfitOrderReason` The reason that the Take Profit Order was initiated

client_extensions
`ClientExtensions` Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

order_fill_transaction_id
`TransactionID` The ID of the OrderFill Transaction that caused this Order to be created (only provided if this Order was created automatically when another Order was filled).

intended_replaces_order_id
`OrderID` The ID of the Order that this Order was intended to replace (only provided if this Order was intended to replace an existing Order).

reject_reason
`TransactionRejectReason` The reason that the Reject Transaction was created

class async_v20.**StopLossOrderTransaction**(*trade_id: TradeID*, *price: PriceValue*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: StopLossOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*, *guaranteed: bool= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A StopLossOrderTransaction represents the creation of a StopLoss Order in the user's Account.

trade_id
`TradeID` The ID of the Trade to close when the price threshold is breached.

id
`TransactionID` The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id
: *TransactionID* The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: *RequestID* The Request ID of the request which generated the transaction.

client_trade_id
: *TradeID* The client ID of the Trade to be closed when the price threshold is breached.

price
: *PriceValue* The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.

time_in_force
: *TimeInForce* The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.

gtd_time
: *DateTime* The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
: *OrderTriggerCondition* Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
: *StopLossOrderReason* The reason that the Stop Loss Order was initiated

client_extensions
: *ClientExtensions* Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

order_fill_transaction_id
: *TransactionID* The ID of the OrderFill Transaction that caused this Order to be created (only provided if this Order was created automatically when another Order was filled).

replaces_order_id
: *OrderID* The ID of the Order that this Order replaces (only provided if this Order replaces an existing Order).

cancelling_transaction_id
: *TransactionID* The ID of the Transaction that cancels the replaced Order (only provided if this Order replaces an existing Order).

guaranteed
: `bool` Flag indicating that the Stop Loss Order is guaranteed. The default value depends on the GuaranteedStopLossOrderMode of the account, if it is REQUIRED, the default will be true, for DISABLED or ENABLED the default is false.

class async_v20.**StopLossOrderRejectTransaction**(*trade_id: TradeID= sentinel*, *price: PriceValue= sentinel*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: StopLossOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *intended_replaces_order_id: OrderID= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A StopLossOrderRejectTransaction represents the rejection of the creation of a StopLoss Order.

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

price
PriceValue The price threshold specified for the StopLoss Order. The associated Trade will be closed by a market price that is equal to or worse than this threshold.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

client_trade_id
TradeID The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
TimeInForce The time-in-force requested for the StopLoss Order. Restricted to "GTC", "GFD" and "GTD" for StopLoss Orders.

gtd_time
DateTime The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
`StopLossOrderReason` The reason that the Stop Loss Order was initiated

client_extensions
`ClientExtensions` Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

order_fill_transaction_id
`TransactionID` The ID of the OrderFill Transaction that caused this Order to be created (only provided if this Order was created automatically when another Order was filled).

intended_replaces_order_id
`OrderID` The ID of the Order that this Order was intended to replace (only provided if this Order was intended to replace an existing Order).

reject_reason
`TransactionRejectReason` The reason that the Reject Transaction was created

class async_v20.**TrailingStopLossOrderTransaction**(*trade_id: TradeID*, *distance: PriceValue*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: TrailingStopLossOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *replaces_order_id: OrderID= sentinel*, *cancelling_transaction_id: TransactionID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TrailingStopLossOrderTransaction represents the creation of a TrailingStopLoss Order in the user's Account.

trade_id
`TradeID` The ID of the Trade to close when the price threshold is breached.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

client_trade_id
TradeID The client ID of the Trade to be closed when the price threshold is breached.

distance
PriceValue The price distance specified for the TrailingStopLoss Order.

time_in_force
TimeInForce The time-in-force requested for the TrailingStopLoss Order. Restricted to "GTC", "GFD" and "GTD" for TrailingStopLoss Orders.

gtd_time
DateTime The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
OrderTriggerCondition Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
TrailingStopLossOrderReason The reason that the Trailing Stop Loss Order was initiated

client_extensions
ClientExtensions Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

order_fill_transaction_id
TransactionID The ID of the OrderFill Transaction that caused this Order to be created (only provided if this Order was created automatically when another Order was filled).

replaces_order_id
OrderID The ID of the Order that this Order replaces (only provided if this Order replaces an existing Order).

cancelling_transaction_id
TransactionID The ID of the Transaction that cancels the replaced Order (only provided if this Order replaces an existing Order).

class async_v20.**TrailingStopLossOrderRejectTransaction**(*trade_id: TradeID= sentinel*, *distance: PriceValue= sentinel*, *id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *client_trade_id: ClientID= sentinel*, *time_in_force: TimeInForce=GTC*, *gtd_time: DateTime= sentinel*, *trigger_condition: OrderTriggerCondition=DEFAULT*, *reason: TrailingStopLossOrderReason= sentinel*, *client_extensions: ClientExtensions= sentinel*, *order_fill_transaction_id: TransactionID= sentinel*, *intended_replaces_order_id: OrderID= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TrailingStopLossOrderRejectTransaction represents the rejection of the creation of a TrailingStopLoss Order.

trade_id
TradeID The ID of the Trade to close when the price threshold is breached.

distance
PriceValue The price distance specified for the TrailingStopLoss Order.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

client_trade_id
TradeID The client ID of the Trade to be closed when the price threshold is breached.

time_in_force
`TimeInForce` The time-in-force requested for the TrailingStopLoss Order. Restricted to "GTC", "GFD" and "GTD" for TrailingStopLoss Orders.

gtd_time
`DateTime` The date/time when the StopLoss Order will be cancelled if its timeInForce is "GTD".

trigger_condition
`OrderTriggerCondition` Specification of what component of a price should be used for comparison when determining if the Order should be filled.

reason
`TrailingStopLossOrderReason` The reason that the Trailing Stop Loss Order was initiated

client_extensions
`ClientExtensions` Client Extensions to add to the Order (only provided if the Order is being created with client extensions).

order_fill_transaction_id
`TransactionID` The ID of the OrderFill Transaction that caused this Order to be created (only provided if this Order was created automatically when another Order was filled).

intended_replaces_order_id
`OrderID` The ID of the Order that this Order was intended to replace (only provided if this Order was intended to replace an existing Order).

reject_reason
`TransactionRejectReason` The reason that the Reject Transaction was created

class `async_v20.`**OrderFillTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, order_id: OrderID= sentinel, client_order_id: ClientID= sentinel, instrument: InstrumentName= sentinel, units: DecimalNumber= sentinel, price: PriceValue= sentinel, full_price: ClientPrice= sentinel, reason: OrderFillReason= sentinel, pl: AccountUnits= sentinel, financing: AccountUnits= sentinel, commission: AccountUnits= sentinel, account_balance: AccountUnits= sentinel, trade_opened: TradeOpen= sentinel, trades_closed: ArrayTradeReduce= sentinel, trade_reduced: TradeReduce= sentinel, gain_quote_home_conversion_factor: DecimalNumber= sentinel, loss_quote_home_conversion_factor: DecimalNumber= sentinel, guaranteed_execution_fee: AccountUnits= sentinel, half_spread_cost: AccountUnits= sentinel, requested_units: AccountUnits= sentinel, full_vwap: DecimalNumber= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

An OrderFillTransaction represents the filling of an Order in the client's Account.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id
: *TransactionID* The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: *RequestID* The Request ID of the request which generated the transaction.

order_id
: *OrderID* The ID of the Order filled.

client_order_id
: *ClientID* The client Order ID of the Order filled (only provided if the client has assigned one).

instrument
: *InstrumentName* The name of the filled Order's instrument.

units
: *DecimalNumber* The number of units filled by the Order.

gain_quote_home_conversion_factor
: This is the conversion factor in effect for the Account at the time of the OrderFill for converting any gains realized in Instrument quote units into units of the Account's home currency.

loss_quote_home_conversion_factor
: This is the conversion factor in effect for the Account at the time of the OrderFill for converting any losses realized in Instrument quote units into units of the Account's home currency.

price
: *PriceValue* The average market price that the Order was filled at.

full_price
: *PriceValue* The price in effect for the account at the time of the Order fill.

reason
: *OrderFillReason* The reason that an Order was filled

pl
: *AccountUnits* The profit or loss incurred when the Order was filled.

financing
: *AccountUnits* The financing paid or collected when the Order was filled.

commission
: *AccountUnits* The commission charged in the Account's home currency as a result of filling the Order. The commission is always represented as a positive quantity of the Account's home currency, however it reduces the balance in the Account.

guaranteed_execution_fee
: The total guaranteed execution fees charged for all Trades opened, closed or reduced with guaranteed Stop Loss Orders.

account_balance
: *AccountUnits* The Account's balance after the Order was filled.

trade_opened
: *TradeOpen* The Trade that was opened when the Order was filled (only provided if filling the Order resulted in a new Trade).

trades_closed
(`ArrayTradeReduce` The Trades that were closed when the Order was filled (only provided if filling the Order resulted in a closing open Trades).

trade_reduced
`TradeReduce` The Trade that was reduced when the Order was filled (only provided if filling the Order resulted in reducing an open Trade).

half_spread_cost
The half spread cost for the OrderFill, which is the sum of the halfSpreadCost values in the tradeOpened, tradesClosed and tradeReduced fields. This can be a positive or negative value and is represented in the home currency of the Account.

class async_v20.**OrderCancelTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *order_id: OrderID= sentinel*, *client_order_id: ClientID= sentinel*, *reason: OrderCancelReason= sentinel*, *replaced_by_order_id: OrderID= sentinel*, *closed_trade_id: OrderID= sentinel*, *trade_close_transaction_id: TransactionID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

An OrderCancelTransaction represents the cancellation of an Order in the client's Account.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

order_id
`TransactionID` The ID of the Order cancelled

client_order_id
`ClientID` The reason that the Order was cancelled.

reason
`OrderCancelReason` The reason that the Order was cancelled.

replaced_by_order_id
`TransactionID` The ID of the Order that replaced this Order (only provided if this Order was cancelled for replacement).

class async_v20.**OrderCancelRejectTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *order_id: OrderID= sentinel*, *client_order_id: ClientID= sentinel*, *reason: OrderCancelReason= sentinel*, *reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

An OrderCancelRejectTransaction represents the rejection of the cancellation of an Order in the client's Account.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

order_id
OrderID The ID of the Order intended to be cancelled

client_order_id
OrderID The client ID of the Order intended to be cancelled (only provided if the Order has a client Order ID).

reason
OrderCancelReason The reason that the Order was to be cancelled.

reject_reason
TransactionRejectReason The reason that the Reject Transaction was created

class async_v20.**OrderClientExtensionsModifyTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *order_id: OrderID= sentinel*, *client_order_id: ClientID= sentinel*, *client_extensions_modify: ClientExtensions= sentinel*, *trade_client_extensions_modify: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A OrderClientExtensionsModifyTransaction represents the modification of an Order's Client Extensions.

id
: *TransactionID* The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id
: *TransactionID* The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: *RequestID* The Request ID of the request which generated the transaction.

order_id
: *OrderID* The ID of the Order who's client extensions are to be modified.

client_order_id
: *OrderID* The original Client ID of the Order who's client extensions are to be modified.

client_extensions_modify
: *ClientExtensions* The new Client Extensions for the Order.

trade_client_extensions_modify
: *ClientExtensions* The new Client Extensions for the Order's Trade on fill.

class async_v20.**OrderClientExtensionsModifyRejectTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, order_id: OrderID= sentinel, client_order_id: ClientID= sentinel, client_extensions_modify: ClientExtensions= sentinel, trade_client_extensions_modify: ClientExtensions= sentinel, reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A OrderClientExtensionsModifyRejectTransaction represents the rejection of the modification of an Order's Client Extensions.

id
: *TransactionID* The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: *AccountID* The ID of the Account the Transaction was created for.

batch_id
: *TransactionID* The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: *RequestID* The Request ID of the request which generated the transaction.

order_id
: *OrderID* The ID of the Order who's client extensions are to be modified.

client_order_id
: *OrderID* The original Client ID of the Order who's client extensions are to be modified.

client_extensions_modify
: *ClientExtensions* The new Client Extensions for the Order.

trade_client_extensions_modify
: *ClientExtensions* The new Client Extensions for the Order's Trade on fill.

reject_reason
: *TransactionRejectReason* The reason that the Reject Transaction was created

class async_v20.**TradeClientExtensionsModifyTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, trade_id: TradeID= sentinel, client_trade_id: ClientID= sentinel, trade_client_extensions_modify: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TradeClientExtensionsModifyTransaction represents the modification of a Trade's Client Extensions.

id
: *TransactionID* The Transaction's Identifier.

time
: *DateTime* The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
    AccountID The ID of the Account the Transaction was created for.

batch_id
    TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
    RequestID The Request ID of the request which generated the transaction.

trade_id
    TradeID The ID of the Trade who's client extensions are to be modified.

client_trade_id
    TradeID The original Client ID of the Trade who's client extensions are to be modified.

trade_client_extensions_modify
    ClientExtensions The new Client Extensions for the Trade.

class async_v20.**TradeClientExtensionsModifyRejectTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, trade_id: TradeID= sentinel, client_trade_id: ClientID= sentinel, trade_client_extensions_modify: ClientExtensions= sentinel, reject_reason: TransactionRejectReason= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A TradeClientExtensionsModifyRejectTransaction represents the rejection of the modification of a Trade's Client Extensions.

id
    TransactionID The Transaction's Identifier.

time
    DateTime The date/time when the Transaction was created.

user_id
    `int` The ID of the user that initiated the creation of the Transaction.

account_id
    AccountID The ID of the Account the Transaction was created for.

batch_id
    TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: `RequestID` The Request ID of the request which generated the transaction.

trade_id
: `TradeID` The ID of the Trade who's client extensions are to be modified.

client_trade_id
: `ClientID` The original Client ID of the Trade who's client extensions are to be modified.

trade_client_extensions_modify
: `ClientExtensions` The new Client Extensions for the Trade.

reject_reason
: `TransactionRejectReason` The reason that the Reject Transaction was created

class async_v20.**MarginCallEnterTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarginCallEnterTransaction is created when an Account enters the margin call state.

id
: `TransactionID` The Transaction's Identifier.

time
: `DateTime` The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
: `AccountID` The ID of the Account the Transaction was created for.

batch_id
: `TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
: `RequestID` The Request ID of the request which generated the transaction.

class async_v20.**MarginCallExtendTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *extension_number: int= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarginCallExtendTransaction is created when the margin call state for an Account has been extended.

id
: `TransactionID` The Transaction's Identifier.

time
: `DateTime` The date/time when the Transaction was created.

user_id
: `int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

extension_number
`int` The number of the extensions to the Account's current margin call that have been applied. This value will be set to 1 for the first MarginCallExtend Transaction

class `async_v20.`**MarginCallExitTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A MarginCallExitnterTransaction is created when an Account leaves the margin call state.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
`AccountID` The ID of the Account the Transaction was created for.

batch_id
`TransactionID` The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
`RequestID` The Request ID of the request which generated the transaction.

class `async_v20.`**DelayedTradeClosureTransaction**(*id: TransactionID= sentinel, time: DateTime= sentinel, user_id: int= sentinel, account_id: AccountID= sentinel, batch_id: TransactionID= sentinel, request_id: RequestID= sentinel, reason: MarketOrderReason= sentinel, trade_ids: TradeID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A DelayedTradeClosure Transaction is created administratively to indicate open trades that should have been closed but weren't because the open trades' instruments were untradeable at the time. Open trades listed in this transaction will be closed once their respective instruments become tradeable.

id
`TransactionID` The Transaction's Identifier.

time
`DateTime` The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

reason
MarketOrderReason The reason for the delayed trade closure

trade_ids
TradeID List of Trade ID's identifying the open trades that will be closed when their respective instruments become tradeable

class async_v20.**DailyFinancingTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*, *financing: AccountUnits= sentinel*, *account_balance: AccountUnits= sentinel*, *account_financing_mode: AccountFinancingMode= sentinel*, *position_financings: ArrayPositionFinancing= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A DailyFinancingTransaction represents the daily payment/collection of financing for an Account.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

financing
AccountUnits The amount of financing paid/collected for the Account.

account_balance
AccountUnits The Account's balance after daily financing.

account_financing_mode
AccountFinancingMode The account financing mode at the time of the daily financing.

position_financings
(*PositionFinancing*, ...) The financing paid/collected for each Position in the Account.

class async_v20.**ResetResettablePLTransaction**(*id: TransactionID= sentinel*, *time: DateTime= sentinel*, *user_id: int= sentinel*, *account_id: AccountID= sentinel*, *batch_id: TransactionID= sentinel*, *request_id: RequestID= sentinel*)

Bases: `async_v20.definitions.types.Transaction`

A ResetResettablePLTransaction represents the resetting of the Account's resettable PL counters.

id
TransactionID The Transaction's Identifier.

time
DateTime The date/time when the Transaction was created.

user_id
`int` The ID of the user that initiated the creation of the Transaction.

account_id
AccountID The ID of the Account the Transaction was created for.

batch_id
TransactionID The ID of the "batch" that the Transaction belongs to. Transactions in the same batch are applied to the Account simultaneously.

request_id
RequestID The Request ID of the request which generated the transaction.

class async_v20.**ClientExtensions**(*id: ClientID= sentinel*, *tag: ClientTag= sentinel*, *comment: ClientComment= sentinel*)

Bases: `async_v20.definitions.base.Model`

A ClientExtensions object allows a client to attach a clientID, tag and comment to Orders and Trades in their Account. Do not set, modify, or delete this field if your account is associated with MT4.

id
ClientID The Client ID of the Order/Trade

tag
ClientTag A tag associated with the Order/Trade

comment
ClientComment A comment associated with the Order/Trade

class async_v20.**TakeProfitDetails**(*price: PriceValue= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.base.Model`

TakeProfitDetails specifies the details of a Take Profit Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Take Profit, or when a Trade's dependent Take Profit Order is modified directly through the Trade.

price
PriceValue The price that the Take Profit Order will be triggered at.

time_in_force
TimeInForce The time in force for the created Take Profit Order. This may only be GTC, GTD or GFD.

gtd_time
DateTime The date when the Take Profit Order will be cancelled on if timeInForce is GTD.

client_extensions
ClientExtensions The Client Extensions to add to the Take Profit Order when created.

class async_v20.**StopLossDetails**(*price: PriceValue= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.base.Model`

StopLossDetails specifies the details of a Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Stop Loss, or when a Trade's dependent Stop Loss Order is modified directly through the Trade.

price

PriceValue The price that the Stop Loss Order will be triggered at.

time_in_force

TimeInForce The time in force for the created Stop Loss Order. This may only be GTC, GTD or GFD.

gtd_time

DateTime The date when the Stop Loss Order will be cancelled on if timeInForce is GTD.

client_extensions

ClientExtensions The Client Extensions to add to the Stop Loss Order when created.

class async_v20.**TrailingStopLossDetails**(*distance: PriceValue= sentinel*, *time_in_force: TimeInForce= sentinel*, *gtd_time: DateTime= sentinel*, *client_extensions: ClientExtensions= sentinel*)

Bases: `async_v20.definitions.base.Model`

TrailingStopLossDetails specifies the details of a Trailing Stop Loss Order to be created on behalf of a client. This may happen when an Order is filled that opens a Trade requiring a Trailing Stop Loss, or when a Trade's dependent Trailing Stop Loss Order is modified directly through the Trade.

distance

PriceValue The distance (in price units) from the Trade's fill price that the Trailing Stop Loss Order will be triggered at.

time_in_force

TimeInForce The time in force for the created Trailing Stop Loss Order. This may only be GTC, GTD or GFD.

gtd_time

DateTime The date when the Trailing Stop Loss Order will be cancelled on if timeInForce is GTD.

client_extensions

ClientExtensions The Client Extensions to add to the Trailing Stop Loss Order when created.

class async_v20.**TradeOpen**(*price: DecimalNumber= sentinel*, *trade_id: TradeID= sentinel*, *units: DecimalNumber= sentinel*, *client_extensions: ClientExtensions= sentinel*, *guaranteed_execution_fee: AccountUnits= sentinel*, *half_spread_cost: AccountUnits= sentinel*, *initial_margin_required: AccountUnits= sentinel*)

Bases: `async_v20.definitions.base.Model`

A TradeOpen object represents a Trade for an instrument that was opened in an Account. It is found embedded in Transactions that affect the position of an instrument in the Account, specifically the OrderFill Transaction.

trade_id

TradeID The ID of the Trade that was opened

units

DecimalNumber The number of units opened by the Trade

client_extensions

ClientExtensions The client extensions for the newly opened Trade

initial_margin_required
AccountUnits The margin required at the time the Trade was created. Note, this is the 'pure' margin required, it is not the 'effective' margin used that factors in the trade risk if a GSLO is attached to the trade.

class async_v20.**TradeReduce**(*trade_id: TradeID= sentinel*, *units: DecimalNumber= sentinel*, *realized_pl: AccountUnits= sentinel*, *financing: AccountUnits= sentinel*, *price: DecimalNumber= sentinel*, *guaranteed_execution_fee: AccountUnits= sentinel*, *half_spread_cost: AccountUnits= sentinel*, *client_trade_id: ClientID= sentinel*)
Bases: `async_v20.definitions.base.Model`

A TradeReduce object represents a Trade for an instrument that was reduced (either partially or fully) in an Account. It is found embedded in Transactions that affect the position of an instrument in the account, specifically the OrderFill Transaction.

trade_id
TradeID The ID of the Trade that was reduced or closed

units
DecimalNumber The number of units that the Trade was reduced by

realized_pl
AccountUnits The PL realized when reducing the Trade

financing
AccountUnits The financing paid/collected when reducing the Trade

client_trade_id
ClientID The ID specified by the client (undocumented by Oanda)

class async_v20.**MarketOrderTradeClose**(*trade_id: TradeID= sentinel*, *client_trade_id: str= sentinel*, *units: str= sentinel*)
Bases: `async_v20.definitions.base.Model`

A MarketOrderTradeClose specifies the extensions to a Market Order that has been created specifically to close a Trade.

trade_id
TradeID The ID of the Trade requested to be closed

client_trade_id
`str` *TradeID* The client ID of the Trade requested to be closed

units
`str` Indication of how much of the Trade to close. Either "ALL", or a DecimalNumber reflection a partial close of the Trade.

class async_v20.**MarketOrderMarginCloseout**(*reason: MarketOrderMarginCloseoutReason= sentinel*)
Bases: `async_v20.definitions.base.Model`

Details for the Market Order extensions specific to a Market Order placed that is part of a Market Order Margin Closeout in a client's account

reason
MarketOrderMarginCloseoutReason The reason the Market Order was created to perform a margin closeout

class async_v20.**MarketOrderDelayedTradeClose**(*trade_id: TradeID= sentinel*, *client_trade_id: TradeID= sentinel*, *source_transaction_id: TransactionID= sentinel*)
Bases: `async_v20.definitions.base.Model`

Details for the Market Order extensions specific to a Market Order placed with the intent of fully closing a specific open trade that should have already been closed but wasn't due to halted market conditions

trade_id
`TradeID` The ID of the Trade being closed

client_trade_id
`TradeID` The Client ID of the Trade being closed

source_transaction_id
`TransactionID` The Transaction ID of the DelayedTradeClosure transaction to which this Delayed Trade Close belongs to

class `async_v20.`**MarketOrderPositionCloseout**(*instrument: InstrumentName= sentinel*, *units: str= sentinel*)
Bases: `async_v20.definitions.base.Model`

A MarketOrderPositionCloseout specifies the extensions to a Market Order when it has been created to closeout a specific Position.

instrument
`InstrumentName` The instrument of the Position being closed out.

units
`str` Indication of how much of the Position to close. Either "ALL", or a DecimalNumber reflection a partial close of the Trade. The DecimalNumber must always be positive, and represent a number that doesn't exceed the absolute size of the Position.

class `async_v20.`**VWAPReceipt**(*units: DecimalNumber= sentinel*, *price: PriceValue= sentinel*)
Bases: `async_v20.definitions.base.Model`

A VWAP Receipt provides a record of how the price for an Order fill is constructed. If the Order is filled with multiple buckets in a depth of market, each bucket will be represented with a VWAP Receipt.

units
`DecimalNumber` The number of units filled

price
`PriceValue` The price at which the units were filled

class `async_v20.`**LiquidityRegenerationSchedule**(*steps: ArrayLiquidityRegenerationScheduleStep= sentinel*)
Bases: `async_v20.definitions.base.Model`

A LiquidityRegenerationSchedule indicates how liquidity that is used when filling an Order for an instrument is regenerated following the fill. A liquidity regeneration schedule will be in effect until the timestamp of its final step, but may be replaced by a schedule created for an Order of the same instrument that is filled while it is still in effect.

steps
(`LiquidityRegenerationScheduleStep`, . . .) The steps in the Liquidity Regeneration Schedule

class `async_v20.`**LiquidityRegenerationScheduleStep**(*timestamp: DateTime= sentinel*, *bid_liquidity_used: DecimalNumber= sentinel*, *ask_liquidity_used: DecimalNumber= sentinel*)
Bases: `async_v20.definitions.base.Model`

A liquidity regeneration schedule Step indicates the amount of bid and ask liquidity that is used by the Account at a certain time. These amounts will only change at the timestamp of the following step.

timestamp
`DateTime` The timestamp of the schedule step.

bid_liquidity_used
`DecimalNumber` The amount of bid liquidity used at this step in the schedule.

ask_liquidity_used
`DecimalNumber` The amount of ask liquidity used at this step in the schedule.

class `async_v20.`**OpenTradeFinancing**(*trade_id: TradeID= sentinel*, *financing: AccountUnits= sentinel*)
Bases: `async_v20.definitions.base.Model`

OpenTradeFinancing is used to pay/collect daily financing charge for an open Trade within an Account

trade_id
`TradeID` The ID of the Trade that financing is being paid/collected for.

financing
`AccountUnits` The amount of financing paid/collected for the Trade.

class `async_v20.`**PositionFinancing**(*instrument: InstrumentName= sentinel*, *financing: AccountUnits= sentinel*, *open_trade_financings: ArrayOpenTradeFinancing= sentinel*)
Bases: `async_v20.definitions.base.Model`

OpenTradeFinancing is used to pay/collect daily financing charge for a Position within an Account

instrument
`InstrumentName` The instrument of the Position that financing is being paid/collected for.

financing
`AccountUnits` The amount of financing paid/collected for the Position.

open_trade_financings
(`OpenTradeFinancing`, . . .) The financing paid/collecte for each open Trade within the Position.

class `async_v20.`**TransactionHeartbeat**(*type: str= sentinel*, *last_transaction_id: TransactionID= sentinel*, *time: DateTime= sentinel*)
Bases: `async_v20.definitions.base.Model`

A TransactionHeartbeat object is injected into the Transaction stream to ensure that the HTTP connection remains active.

type
`str` The string "HEARTBEAT"

last_transaction_id
`TransactionID` The ID of the most recent Transaction created for the Account

time
`DateTime` The date/time when the TransactionHeartbeat was created.

8.12 Primitives

class `async_v20.`**AcceptDatetimeFormat**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

DateTime header

class `async_v20.`**AccountFinancingMode**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The financing mode of an Account

class async_v20.**AccountID**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The string representation of an Account Identifier.

class async_v20.**AccountUnits**
Bases: `float`, `async_v20.definitions.primitives.Primitive`

The string representation of a quantity of an Account's home currency.

class async_v20.**CancellableOrderType**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The type of the Order.

class async_v20.**CandlestickGranularity**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The granularity of a candlestick

class async_v20.**ClientComment**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

A client-provided comment that can contain any data and may be assigned to their Orders or Trades. Comments are typically used to provide extra context or meaning to an Order or Trade.

class async_v20.**ClientID**
Bases: `str`, `async_v20.definitions.primitives.Primitive`, `async_v20.definitions.primitives.Specifier`

A client-provided identifier, used by clients to refer to their Orders or Trades with an identifier that they have provided.

class async_v20.**ClientTag**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

A client-provided tag that can contain any data and may be assigned to their Orders or Trades. Tags are typically used to associate groups of Trades and/or Orders together.

class async_v20.**Currency**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

Currency name identifier. Used by clients to refer to currencies.

class async_v20.**DateTime**
Bases: `async_v20.definitions.primitives.Primitive`

A date and time value using either RFC3339 or UNIX time representation.

class async_v20.**DecimalNumber**
Bases: `float`, `async_v20.definitions.primitives.Primitive`

The string representation of a decimal number.

class async_v20.**Direction**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

In the context of an Order or a Trade, defines whether the units are positive or negative.

class async_v20.**FundingReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that an Account is being funded.

class async_v20.**InstrumentName**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

Instrument name identifier. Used by clients to refer to an Instrument.

class async_v20.**InstrumentType**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The type of an Instrument.

class async_v20.**LimitOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Limit Order was initiated

class async_v20.**MarketIfTouchedOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Market-if-touched Order was initiated

class async_v20.**MarketOrderMarginCloseoutReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Market Order was created to perform a margin closeout

class async_v20.**MarketOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Market Order was created

class async_v20.**OrderCancelReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that an Order was cancelled.

class async_v20.**OrderFillReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that an Order was filled

class async_v20.**OrderID**
Bases: `int`, `async_v20.definitions.primitives.Primitive`, `async_v20.definitions.primitives.Specifier`

The Order's identifier, unique within the Order's Account.

class async_v20.**OrderPositionFill**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

Specification of how Positions in the Account are modified when the Order is filled.

class async_v20.**OrderSpecifier**
Bases: `str`, `async_v20.definitions.primitives.Primitive`, `async_v20.definitions.primitives.Specifier`

The specification of an Order as referred to by clients

class async_v20.**OrderState**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The current state of the Order.

class async_v20.**OrderStateFilter**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The state to filter the requested Orders by.

class async_v20.**OrderTriggerCondition**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

Specification of which price component should be used when determining if an Order should be triggered and filled. This allows Orders to be triggered based on the bid, ask, mid, default (ask for buy, bid for sell) or inverse (ask for sell, bid for buy) price depending on the desired behaviour. Orders are always filled using their default price component. This feature is only provided through the REST API. Clients who choose to specify a non-default trigger condition will not see it reflected in any of OANDA's proprietary or partner trading platforms, their transaction history or their account statements. OANDA platforms always assume that an Order's trigger condition is set to the default value when indicating the distance from an Order's trigger price, and will always provide the default trigger condition when creating or modifying an Order.

class async_v20.**OrderType**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The type of the Order.

class async_v20.**PositionAggregationMode**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The way that position values for an Account are calculated and aggregated.

class async_v20.**PriceComponent**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

class async_v20.**PriceStatus**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The status of the Price.

class async_v20.**PriceValue**
Bases: `float`, `async_v20.definitions.primitives.Primitive`

The string representation of a Price for an Instrument.

class async_v20.**Reason**
Bases: `async_v20.definitions.primitives.Primitive`, `str`

Generic reason for any transaction that may occur

class async_v20.**RequestID**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The request identifier.

class async_v20.**StopLossOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Stop Loss Order was initiated

class async_v20.**StopOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Stop Order was initiated

class async_v20.**TakeProfitOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Take Profit Order was initiated

class async_v20.**TimeInForce**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The time-in-force of an Order. TimeInForce describes how long an Order should remain pending before being automatically cancelled by the execution system.

class async_v20.**TradeID**
Bases: `int`, `async_v20.definitions.primitives.Primitive`, `async_v20.definitions.primitives.Specifier`

The Trade's identifier, unique within the Trade's Account.

class async_v20.**TradePL**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The classification of TradePLs.

class async_v20.**TradeSpecifier**
Bases: `str`, `async_v20.definitions.primitives.Primitive`, `async_v20.definitions.primitives.Specifier`

The identification of a Trade as referred to by clients

class async_v20.**TradeState**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The current state of the Trade.

class async_v20.**TradeStateFilter**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The state to filter the Trades by

class async_v20.**TrailingStopLossOrderReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that the Trailing Stop Loss Order was initiated

class async_v20.**TransactionFilter**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

A filter that can be used when fetching Transactions

class async_v20.**TransactionID**
Bases: `int`, `async_v20.definitions.primitives.Primitive`, `async_v20.definitions.primitives.Specifier`

The unique Transaction identifier within each Account.

class async_v20.**TransactionRejectReason**
Bases: `async_v20.definitions.primitives.Reason`

The reason that a Transaction was rejected.

class async_v20.**TransactionType**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The possible types of a Transaction

class async_v20.**WeeklyAlignment**
Bases: `str`, `async_v20.definitions.primitives.Primitive`

The day of the week to use for candlestick granularities with weekly alignment.

8.13 Annotations

Note: The class' in `async_v20.endpoints.annotations` are used as parameter annotations for some *OandaClient API* calls, in order to correctly map passed arguments to the correct endpoint parameter

class `async_v20.endpoints.annotations.`**`Alias`**
Bases: `str`

class `async_v20.endpoints.annotations.`**`AlignmentTimezone`**
Bases: `str`

The timezone to use for the dailyAlignment parameter. Candlesticks with daily alignment will be aligned to the dailyAlignment hour within the alignmentTimezone. [default=America/New_York]

class `async_v20.endpoints.annotations.`**`Authorization`**
Bases: `str`

Contains OANDA's v20 API authorization token

class `async_v20.endpoints.annotations.`**`Bool`**
Bases: `object`

class `async_v20.endpoints.annotations.`**`Count`**
Bases: `int`

The number of candlesticks to return in the reponse. Count should not be specified if both the start and end parameters are provided, as the time range combined with the graularity will determine the number of candlesticks to return. [default=500, maximum=5000]

class `async_v20.endpoints.annotations.`**`DailyAlignment`**
Bases: `int`

The hour of the day (in the specified timezone) to use for granularities that have daily alignments. [default=17, minimum=0, maximum=23]

class `async_v20.endpoints.annotations.`**`End`**
Bases: `str`

Only show events which started before this date, inclusive. Suggested format RFC 2822 or RFC 1123

class `async_v20.endpoints.annotations.`**`EventSid`**
Bases: `str`

The SID of the event to get

class `async_v20.endpoints.annotations.`**`FromTime`**
Bases: `async_v20.definitions.primitives.DateTime`

A DateTime to be used as the starting period of a query

class `async_v20.endpoints.annotations.`**`FromTransactionID`**
Bases: `async_v20.definitions.primitives.TransactionID`

A TransactionID to be used as the starting period of a query

class `async_v20.endpoints.annotations.`**`Ids`**
Bases: `str`

class `async_v20.endpoints.annotations.`**`IncludeFirstQuery`**
Bases: *`async_v20.endpoints.annotations.Bool`*

A flag that controls whether the candlestick that is covered by the from time should be included in the results. This flag enables clients to use the timestamp of the last completed candlestick received to poll for future candlesticks but avoid receiving the previous candlestick repeatedly. [default=True]

class `async_v20.endpoints.annotations.`**`Instruments`**

Bases: `str`

class `async_v20.endpoints.annotations.`**`LastTransactionID`**

Bases: `async_v20.definitions.primitives.TransactionID`

Contains the most recent TransactionID

class `async_v20.endpoints.annotations.`**`LongClientExtensions`**(*id: ClientID= sentinel*, *tag: ClientTag= sentinel*, *comment: ClientComment= sentinel*)

Bases: `async_v20.definitions.types.ClientExtensions`

The client extensions to add to the MarketOrder used to close the long position

class `async_v20.endpoints.annotations.`**`LongUnits`**

Bases: `str`

Indication of how much of the long Position to closeout. Either the string "ALL", the string "NONE", or a DecimalNumber representing how many units of the long position to close using a PositionCloseout MarketOrder. The units specified must always be positive.

class `async_v20.endpoints.annotations.`**`PageSize`**

Bases: `int`

The number of Transactions to include in each page of the results. [default=100, maximum=1000]

class `async_v20.endpoints.annotations.`**`ServiceID`**

Bases: `str`

The specifier of the service to get

class `async_v20.endpoints.annotations.`**`ServiceListID`**

Bases: `str`

Identification string of service list to get

class `async_v20.endpoints.annotations.`**`ShortClientExtensions`**(*id: ClientID= sentinel*, *tag: ClientTag= sentinel*, *comment: ClientComment= sentinel*)

Bases: `async_v20.definitions.types.ClientExtensions`

The client extensions to add to the MarketOrder used to close the short position

class `async_v20.endpoints.annotations.`**`ShortUnits`**

Bases: `str`

Indication of how much of the short Position to closeout. Either the string "ALL", the string "NONE", or a DecimalNumber representing how many units of the short position to close using a PositionCloseout MarketOrder. The units specified must always be positive.

class `async_v20.endpoints.annotations.`**`SinceTransactionID`**

Bases: `async_v20.definitions.primitives.TransactionID`

The account changes to get Since LastTransactionID for account_changes() method

class `async_v20.endpoints.annotations.`**`Smooth`**

Bases: *`async_v20.endpoints.annotations.Bool`*

A flag that controls whether the candlestick is 'smoothed' or not. A smoothed candlestick uses the previous candle's close price as its open price, while an unsmoothed candlestick uses the first price from its time range as its open price. [default=False]

class `async_v20.endpoints.annotations.`**`Snapshot`**
Bases: *`async_v20.endpoints.annotations.Bool`*

Flag that enables/disables the sending of a pricing snapshot when initially connecting to the stream. [default=True]

class `async_v20.endpoints.annotations.`**`Start`**
Bases: `str`

Only show events which started after this date, inclusive. Suggested format RFC 2822 or RFC 1123

class `async_v20.endpoints.annotations.`**`StatusID`**
Bases: `str`

The ID of the status to get

class `async_v20.endpoints.annotations.`**`ToTime`**
Bases: `async_v20.definitions.primitives.DateTime`

A DateTime to be used as the ending period of a query

class `async_v20.endpoints.annotations.`**`ToTransactionID`**
Bases: `async_v20.definitions.primitives.TransactionID`

A TransactionID to be used as the ending period of a query

class `async_v20.endpoints.annotations.`**`TradeClientExtensions`**(*id: ClientID= sentinel*, *tag: ClientTag= sentinel*, *comment: ClientComment= sentinel*)
Bases: `async_v20.definitions.types.ClientExtensions`

None

class `async_v20.endpoints.annotations.`**`Type`**
Bases: `str`

class `async_v20.endpoints.annotations.`**`Units`**
Bases: `str`

Indication of how much of the Trade to close. Either the string "ALL" (indicating that all of the Trade should be closed), or a DecimalNumber representing the number of units of the open Trade to Close using a TradeClose MarketOrder. The units specified must always be positive, and the magnitude of the value cannot exceed the magnitude of the Trade's open units

class `async_v20.endpoints.annotations.`**`UserSpecifier`**
Bases: `str`

8.14 Exceptions

class `async_v20.exceptions.`**`AsyncV20Exception`**
Bases: `Exception`

A base exception for all exceptions in the async_v20 package

class async_v20.exceptions.**CloseAllTradesFailure**
Bases: *async_v20.exceptions.AsyncV20Exception*

Failed to close all trades

class async_v20.exceptions.**FailedToCreatePath**
Bases: *async_v20.exceptions.AsyncV20Exception*

Unable to construct the path for the requested endpoint

class async_v20.exceptions.**IncompatibleValue**
Bases: *async_v20.exceptions.AsyncV20Exception*

A supplied argument is different than the predefined value

class async_v20.exceptions.**InitializationFailure**
Bases: *async_v20.exceptions.AsyncV20Exception*

OandaClient Failed to initialize

class async_v20.exceptions.**InstantiationFailure**
Bases: *async_v20.exceptions.AsyncV20Exception*

async_v20 was unable to create an object from the passed arguments

class async_v20.exceptions.**InvalidFormatArguments**
Bases: *async_v20.exceptions.AsyncV20Exception*

Arguments to format a DecimalNumber or PriceValue are invalid

class async_v20.exceptions.**InvalidOrderRequest**
Bases: *async_v20.exceptions.AsyncV20Exception*

The order request is not with in the instruments specification the order is for

class async_v20.exceptions.**InvalidValue**
Bases: *async_v20.exceptions.AsyncV20Exception*

A supplied value does not meet the specification of valid values

class async_v20.exceptions.**ResponseTimeout**
Bases: *async_v20.exceptions.AsyncV20Exception*

The server took to long to respond

class async_v20.exceptions.**UnexpectedStatus**
Bases: *async_v20.exceptions.AsyncV20Exception*

The server returned an unexpected HTTP status

8.15 Warnings

class async_v20.exceptions.**UnknownKeywordArgument**
Bases: async_v20.exceptions.AsyncV20Warning

A passed keyword argument is not in the objects __init__ signature

8.16 Logging

async_v20 employs the *logging.getLogger(__name__)* standard for all it's modules. Hence the base logger for async_v20 is 'async_v20'

OandaClient has a *debug* attribute for enabling the logging of debug level messages.

Example

```
>>> from async_v20 import OandaClient
>>> import asyncio
>>> import logging
>>> loop = asyncio.get_event_loop()
>>> run = loop.run_until_complete
>>> import logging
>>> logger = logging.getLogger('async_v20')
>>> handler = logging.StreamHandler()
>>> logger.addHandler(handler)
>>> logger.setLevel(logging.INFO)
>>> client = OandaClient()
>>> rsp = run(client.close_all_trades())
# close_all_trades()
# Initializing client
# Initializing session
# list_services(args=(), kwargs={})
# list_accounts(args=(), kwargs={})
# get_account_details(args=(), kwargs={})
# account_instruments(args=(), kwargs={})
# list_open_trades(args=(), kwargs={})
# list_open_trades(args=(), kwargs={})
```

8.17 Glossary

aiodns An python package for asynchronous DNS resolution.

https://github.com/saghul/aiodns

aiohttp http client/server framework *async_v20* use's to communicate asynchronously.

http://aiohttp.readthedocs.io/en/stable/

annotation Used to describe what *type* an argument takes

https://www.python.org/dev/peps/pep-0526/

arguments are values passed to a function

async_v20 The name of the package this documentation is documenting

https://github.com/jamespeterschinner/async_v20

asyncio The library for writing single-threaded concurrent code using coroutines, multiplexing I/O access over sockets and other resources, running network clients and servers, and other related primitives.

Reference implementation of **PEP 3156**

https://pypi.python.org/pypi/asyncio/

attribute An attribute is the term used to describe variables associated with a *class* that contains information about that class' state.

attributes plural of *attribute*

await Python syntax to execute a *coroutine* inside an asynchronous function

callable Any object that can be called. Use `callable()` to check that.

camelCase A style of text that uses capital letters to separate words

cchardet cChardet is high speed universal character encoding detector - binding to charsetdetect.

https://pypi.python.org/pypi/cchardet/

class Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state.

https://docs.python.org/3/tutorial/classes.html

concurrent The process by which a single threaded application switches between different tasks. Typically done to prevent waiting for I/O bound operations

context manager A Python programming concept that defines *enter* and *exit* methods. Used to handle set-up and tear-down tasks

coroutine A python object that supports asynchronous execution

coroutines plural of *coroutine*

docstring text placed below a function or class definition that documents the function or class

https://www.python.org/dev/peps/pep-0257/

environment variable An environment variable is a dynamic-named value that can affect the way running processes will behave on a computer. They are part of the environment in which a process runs

http Stands for Hyper Text Transfer Protocol

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

JSON JavaScript Object Notation

http://www.json.org/

key The value used to index a dictionary

metaclass A python *class* definition that inherits from *type*. It allows programmers to customize class creation as if they typed the entire class

https://stackoverflow.com/questions/100003/what-is-a-metaclass-in-python

OANDA The name of the Foreign Exchange (FOREX) broker *async_v20* communicates with

OandaClient The client class definition *async_v20* exposes

pandas a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive

http://pandas.pydata.org/

python Programming language

https://www.python.org/

snake_case a style of text that uses underscores to separate words

status When prefixed with http. Refers to a 3 digit number with pre-assigned meaning.

https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Status_codes

token A 65 character string used to uniquely identify and authorise access to the OANDA v20 API

Example: *810492ace47473fa9f72c0eeecd33657-1eae8a55f01431bdd370206f69071e5f*

type In computer science and computer programming, a data type or simply type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data.

https://en.wikipedia.org/wiki/Data_type

virtual environment A self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

https://docs.python.org/3/tutorial/venv.html

Index