OANDA REST-V20 API Documentation

Release 0.2.2

Feite Brekeveld

November 30, 2016

Contents:

CHAPTER 1

Installation

1.1 Introduction

The OANDA REST-V20 package offers an API to the OANDA V20 REST service. To use the API-service you will need a *token* and an *account*. This applies for both *live* and *practice* accounts. For details check oanda.com.

1.2 Download & Install

Install the package with pip:

```
$ pip install git+https://github.com/hootnot/oanda-api-v20.git
```

You may consider using *virtualenv* to create isolated Python environments. Python 3.4 has *pyvenv* providing the same kind of functionality.

1.2.1 From Github

```
$ git clone https://github.com/hootnot/oanda-api-v20.git
$ cd oanda-api-v20

Run the tests:
$ python setup.py test
$ python setup.py install
```

CHAPTER 2

OANDA REST-V20 API

2.1 Interface to the OANDA V20 REST-service

class oandapyV20.**API**(*access_token*, *environment='practice'*, *headers=None*, *request_params=None*)

Bases: object

API - class to handle APIRequests objects to access API endpoints.

Examples

```
# get a list of trades
from oandapyV20 import API
import oandapyV20.endpoints.trades as trades

api = API(access_token="xxx")
accountID = "101-305-3091856-001"

r = trades.TradesList(accountID)
# show the endpoint as it is constructed for this call
print("REQUEST:{}".format(r))
rv = api.request(r)
print("RESPONSE:\n{}".format(json.dumps(rv, indent=2)))
```

Output:

```
REQUEST:v3/accounts/101-305-3091856-001/trades
RESPONSE:
"trades": [
    {
      "financing": "0.0000",
      "openTime": "2016-07-21T15:47:05.170212014Z",
      "price": "10133.9",
      "unrealizedPL": "8.0000",
      "realizedPL": "0.0000",
      "instrument": "DE30_EUR",
      "state": "OPEN",
      "initialUnits": "-10",
      "currentUnits": "-10",
      "id": "1032"
    },
    {
      "financing": "0.0000",
```

```
        "openTime": "2016-07-21T15:47:04.963590941Z",
        "price": "10134.4",
        "unrealizedPL": "13.0000",
        "realizedPL": "0.0000",
        "instrument": "DE30_EUR",
        "state": "OPEN",
        "initialUnits": "-10",
        "currentUnits": "-10",
        "id": "1030"
      }
    ],
    "lastTransactionID": "1040"
}
```

```
# reduce a trade by it's id
from oandapyV20 import API
import oandapyV20.endpoints.trades as trades

api = API(access_token="...")

accountID = "101-305-3091856-001"
tradeID = "1030"
cfg = {"units": 5}
r = trades.TradeClose(accountID, tradeID=tradeID, data=cfg)
# show the endpoint as it is constructed for this call
print("REQUEST:{}".format(r))
rv = api.request(r)
print("RESPONSE\n{}".format(json.dumps(rv, indent=2)))
```

Output:

```
REQUEST:v3/accounts/101-305-3091856-001/trades/1030/close
RESPONSE: {
  "orderFillTransaction": {
    "orderID": "1041",
    "financing": "-0.1519",
    "instrument": "DE30_EUR",
    "userID": 1435156,
    "price": "10131.6",
    "tradeReduced": {
      "units": "5",
      "financing": "-0.1519",
      "realizedPL": "14.0000",
      "tradeID": "1030"
    },
    "batchID": "1041",
    "accountBalance": "44876.2548",
    "reason": "MARKET_ORDER_TRADE_CLOSE",
    "time": "2016-07-21T17:32:51.361464739Z",
    "units": "5",
    "type": "ORDER_FILL",
    "id": "1042",
    "pl": "14.0000",
    "accountID": "101-305-3091856-001"
  },
  "orderCreateTransaction": {
    "timeInForce": "FOK",
    "positionFill": "REDUCE_ONLY",
```

```
        "userID": 1435156,
        "batchID": "1041",
        "instrument": "DE30_EUR",
        "reason": "TRADE_CLOSE",
        "tradeClose": {
          "units": "5",
          "tradeID": "1030"
        },
        "time": "2016-07-21T17:32:51.361464739Z",
        "units": "5",
        "type": "MARKET_ORDER",
        "id": "1041",
        "accountID": "101-305-3091856-001"
      },
      "relatedTransactionIDs": [
        "1041",
        "1042"
      ],
      "lastTransactionID": "1042"
    }
```

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__init__(*access_token*, *environment='practice'*, *headers=None*, *request_params=None*)
Instantiate an instance of OandaPy's API wrapper.

Parameters

- **access_token** (*string*) – Provide a valid access token.
- **environment** (*string*) – Provide the environment for OANDA's REST api. Valid values: 'practice' or 'live'. Default: 'practice'.
- **headers** (*dict (optional)*) – Provide request headers to be set for a request.

Note: There is no need to set the 'Content-Type: application/json' for the endpoints that require this header. The API-request classes covering those endpoints will take care of the header.

---

request_params [(optional)] parameters to be passed to the request. This can be used to apply for instance a timeout value:

request_params={"timeout": 0.1}

See specs of the requests module for full details of possible parameters.

Warning: parameters belonging to a request need to be set on the requestinstance and are NOT passed via the client.

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

request(*endpoint*)
Perform a request for the APIRequest instance 'endpoint'.

Parameters **endpoint** (APIRequest) – The endpoint parameter contains an instance of an APIRequest containing the endpoint, method and optionally other parameters or body data.

Raises V20Error in case of HTTP response code >= 400

request_params
request_params property.

2.2 Exceptions

class oandapyV20.**V20Error**(*code*, *msg*)
Bases: exceptions.Exception

Generic error class.

In case of HTTP response codes >= 400 this class can be used to raise an exception representing that error.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__getitem__
x.__getitem__(y) <==> x[y]

__getslice__
x.__getslice__(i, j) <==> x[i:j]

Use of negative indices is not supported.

__hash__

__init__(*code*, *msg*)
Instantiate a V20Error.

Parameters

- **code** (*int*) – the HTTP-code of the response
- **msg** (*str*) – the message returned with the response

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

2.3 Logging

The OANDA REST-V20 package has *logging* integrated. Logging can be simply applied by enabling a *logger* . The example below will log INFO-level logging to the file *v20.log*. For details check the *logger* module in the standard Python documentation.

```
# code snippet
from oandapyV20 import API
import oandapyV20.endpoints.orders as orders
from oandapyV20.exceptions import V20Error
from exampleauth import exampleAuth
import logging

logging.basicConfig(
    filename="v20.log",
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(name)s : %(message)s',
)

accountID, token = exampleAuth()
...
```

Resulting loglines:

```
2016-10-22 17:50:37,988 [INFO] oandapyV20.oandapyV20 : setting up API-client for environment practice
2016-10-22 17:50:37,990 [INFO] oandapyV20.oandapyV20 : performing request https://api-fxpractice.oand
2016-10-22 17:50:37,998 [INFO] requests.packages.urllib3.connectionpool : Starting new HTTPS connecti
2016-10-22 17:50:38,866 [INFO] oandapyV20.oandapyV20 : performing request https://api-fxpractice.oand
2016-10-22 17:50:39,066 [ERROR] oandapyV20.oandapyV20 : request https://api-fxpractice.oanda.com/v3/a
```

CHAPTER 3

oandapyV20

3.1 oandapyV20 package

3.1.1 Subpackages

oandapyV20.endpoints package

Submodules

oandapyV20.endpoints.apirequest module

Handling of API requests.

class oandapyV20.endpoints.apirequest.**APIRequest**(*endpoint*, *method='GET'*, *expected_status=200*)

Bases: object

Base Class for API-request classes.

__init__(*endpoint*, *method='GET'*, *expected_status=200*)
Instantiate an API request.

Parameters

- **endpoint** (*string*) – the URL format string
- **method** (*string*) – the method for the request. Default: GET.

__str__()
return the endpoint.

expected_status

response
response - get the response of the request.

status_code

oandapyV20.endpoints.accounts module

Handle account endpoints.

class oandapyV20.endpoints.accounts.**AccountChanges**(*accountID*, *params=None*)

Bases: *oandapyV20.endpoints.accounts.Accounts*

AccountChanges.

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

ENDPOINT = 'v3/accounts/{accountID}/changes'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *params=None*)

Instantiate an AccountChanges request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict (optional)*) – query params to send, check developer.oanda.com for details.

Query Params example:

```
{
  "sinceTransactionID": 2308
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.accounts as accounts
>>> client = oandapyV20.API(access_token=...)
>>> params = ...
>>> r = accounts.AccountChanges(accountID=..., params=params)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "state": {
    "trades": [],
    "marginCloseoutNAV": "33848.2663",
    "marginUsed": "0.0000",
    "marginAvailable": "33848.2663",
    "marginCallPercent": "0.00000",
    "NAV": "33848.2663",
    "marginCloseoutMarginUsed": "0.0000",
    "orders": [],
    "withdrawalLimit": "33848.2663",
    "marginCloseoutPercent": "0.00000",
    "positions": [],
    "unrealizedPL": "0.0000",
    "marginCallMarginUsed": "0.0000",
    "marginCloseoutUnrealizedPL": "0.0000",
    "positionValue": "0.0000"
  },
  "changes": {
    "tradesReduced": [],
    "tradesOpened": [],
    "ordersFilled": [],
    "tradesClosed": [],
```

```
            "transactions": [
              {
                "price": "1.20000",
                "stopLossOnFill": {
                  "timeInForce": "GTC",
                  "price": "1.22000"
                },
                "timeInForce": "GTC",
                "reason": "CLIENT_ORDER",
                "id": "2309",
                "batchID": "2309",
                "triggerCondition": "TRIGGER_DEFAULT",
                "positionFill": "DEFAULT",
                "userID": 1435156,
                "instrument": "EUR_USD",
                "time": "2016-10-25T21:07:21.065554321Z",
                "units": "-100",
                "type": "LIMIT_ORDER",
                "accountID": "101-004-1435156-001"
              }
            ],
            "ordersCreated": [
              {
                "partialFill": "DEFAULT_FILL",
                "price": "1.20000",
                "stopLossOnFill": {
                  "timeInForce": "GTC",
                  "price": "1.22000"
                },
                "timeInForce": "GTC",
                "createTime": "2016-10-25T21:07:21.065554321Z",
                "triggerCondition": "TRIGGER_DEFAULT",
                "positionFill": "POSITION_DEFAULT",
                "id": "2309",
                "instrument": "EUR_USD",
                "state": "PENDING",
                "units": "-100",
                "type": "LIMIT"
              }
            ],
            "positions": [],
            "ordersTriggered": [],
            "ordersCancelled": []
          },
          "lastTransactionID": "2309"
        }
```


class oandapyV20.endpoints.accounts.**AccountConfiguration**(*accountID*, *data*)

Bases: *oandapyV20.endpoints.accounts.Accounts*

Set the client-configurable portions of an Account.

ENDPOINT = 'v3/accounts/{accountID}/configuration'

EXPECTED_STATUS = 200

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PATCH'

__init__(*accountID*, *data*)
Instantiate an AccountConfiguration request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **data** (*dict (required)*) – json body to send

body example:

```
{
  "marginRate": "0.05"
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.accounts as accounts
>>> client = oandapyV20.API(access_token=...)
>>> r = accounts.AccountConfiguration(accountID, data=data)
>>> client.request(r)
>>> print r.response
```

```
{
  "lastTransactionID": "830",
  "clientConfigureTransaction": {
    "userID": 1435156,
    "marginRate": "0.05",
    "batchID": "830",
    "time": "2016-07-12T19:48:11.657494168Z",
    "type": "CLIENT_CONFIGURE",
    "id": "830",
    "accountID": "101-004-1435156-001"
  }
}
```

class oandapyV20.endpoints.accounts.**AccountDetails**(*accountID*)
Bases: *oandapyV20.endpoints.accounts.Accounts*

AccountDetails.

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.

ENDPOINT = 'v3/accounts/{accountID}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*)
Instantiate an AccountDetails request.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.accounts as accounts
>>> client = oandapyV20.API(access_token=...)
>>> r = accounts.AccountDetails(accountID)
>>> client.request(r)
>>> print r.response
```

```
{
  "account": {
    "marginCloseoutNAV": "35454.4740",
    "marginUsed": "10581.5000",
    "currency": "EUR",
    "resettablePL": "-13840.3525",
    "NAV": "35454.4740",
    "marginCloseoutMarginUsed": "10581.5000",
    "marginCloseoutPositionValue": "211630.0000",
    "openTradeCount": 2,
    "id": "101-004-1435156-001",
    "hedgingEnabled": false,
    "marginCloseoutPercent": "0.14923",
    "marginCallMarginUsed": "10581.5000",
    "openPositionCount": 1,
    "positionValue": "211630.0000",
    "pl": "-13840.3525",
    "lastTransactionID": "2123",
    "marginAvailable": "24872.9740",
    "marginRate": "0.05",
    "marginCallPercent": "0.29845",
    "pendingOrderCount": 0,
    "withdrawalLimit": "24872.9740",
    "unrealizedPL": "0.0000",
    "alias": "hootnotv20",
    "createdByUserID": 1435156,
    "marginCloseoutUnrealizedPL": "0.0000",
    "createdTime": "2016-06-24T21:03:50.914647476Z",
    "balance": "35454.4740"
  },
  "lastTransactionID": "2123"
}
```


class oandapyV20.endpoints.accounts.**AccountInstruments**(*accountID*, *params=None*)

Bases: *oandapyV20.endpoints.accounts.Accounts*

AccountInstruments.

Get the list of tradable 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.

ENDPOINT = 'v3/accounts/{accountID}/instruments'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *params=None*)

Instantiate an AccountInstruments request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict (optional)*) – query params to send, check developer.oanda.com for details.

Query Params example:

```
{
  "instruments": "EU50_EUR,EUR_USD,US30_USD,FR40_EUR,EUR_CHF,DE30_EUR"
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.accounts as accounts
>>> client = oandapyV20.API(access_token=...)
>>> params = ...
>>> r = accounts.AccountInstruments(accountID=..., params=params)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "instruments": [
    {
      "marginRate": "0.05",
      "minimumTrailingStopDistance": "5.0",
      "maximumPositionSize": "0",
      "minimumTradeSize": "1",
      "displayName": "Europe 50",
      "name": "EU50_EUR",
      "displayPrecision": 1,
      "maximumTrailingStopDistance": "10000.0",
      "maximumOrderUnits": "3000",
      "tradeUnitsPrecision": 0,
      "pipLocation": 0,
      "type": "CFD"
    },
    {
      "marginRate": "0.05",
      "minimumTrailingStopDistance": "0.00050",
      "maximumPositionSize": "0",
      "minimumTradeSize": "1",
      "displayName": "EUR/USD",
      "name": "EUR_USD",
      "displayPrecision": 5,
      "maximumTrailingStopDistance": "1.00000",
      "maximumOrderUnits": "100000000",
      "tradeUnitsPrecision": 0,
      "pipLocation": -4,
      "type": "CURRENCY"
    },
    {
      "marginRate": "0.05",
      "minimumTrailingStopDistance": "5.0",
      "maximumPositionSize": "0",
      "minimumTradeSize": "1",
      "displayName": "US Wall St 30",
      "name": "US30_USD",
      "displayPrecision": 1,
      "maximumTrailingStopDistance": "10000.0",
      "maximumOrderUnits": "1000",
      "tradeUnitsPrecision": 0,
      "pipLocation": 0,
      "type": "CFD"
    },
```

```
            {
              "marginRate": "0.05",
              "minimumTrailingStopDistance": "5.0",
              "maximumPositionSize": "0",
              "minimumTradeSize": "1",
              "displayName": "France 40",
              "name": "FR40_EUR",
              "displayPrecision": 1,
              "maximumTrailingStopDistance": "10000.0",
              "maximumOrderUnits": "2000",
              "tradeUnitsPrecision": 0,
              "pipLocation": 0,
              "type": "CFD"
            },
            {
              "marginRate": "0.05",
              "minimumTrailingStopDistance": "0.00050",
              "maximumPositionSize": "0",
              "minimumTradeSize": "1",
              "displayName": "EUR/CHF",
              "name": "EUR_CHF",
              "displayPrecision": 5,
              "maximumTrailingStopDistance": "1.00000",
              "maximumOrderUnits": "100000000",
              "tradeUnitsPrecision": 0,
              "pipLocation": -4,
              "type": "CURRENCY"
            },
            {
              "marginRate": "0.05",
              "minimumTrailingStopDistance": "5.0",
              "maximumPositionSize": "0",
              "minimumTradeSize": "1",
              "displayName": "Germany 30",
              "name": "DE30_EUR",
              "displayPrecision": 1,
              "maximumTrailingStopDistance": "10000.0",
              "maximumOrderUnits": "2500",
              "tradeUnitsPrecision": 0,
              "pipLocation": 0,
              "type": "CFD"
            }
          ],
          "lastTransactionID": "2124"
        }
```


class oandapyV20.endpoints.accounts.**AccountList**

Bases: *oandapyV20.endpoints.accounts.Accounts*

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

ENDPOINT = 'v3/accounts'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()

Instantiate an AccountList request.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.accounts as accounts
>>> client = oandapyV20.API(access_token=...)
>>> r = accounts.AccountList()
>>> client.request(r)
>>> print r.response
```


```
{
  "account": {
    "positions": [
      {
        "short": {
          "units": "0",
          "resettablePL": "0.0000",
          "unrealizedPL": "0.0000",
          "pl": "0.0000"
        },
        "unrealizedPL": "0.0000",
        "long": {
          "units": "0",
          "resettablePL": "-3.8046",
          "unrealizedPL": "0.0000",
          "pl": "-3.8046"
        },
        "instrument": "EUR_USD",
        "resettablePL": "-3.8046",
        "pl": "-3.8046"
      },
      {
        "short": {
          "unrealizedPL": "682.0000",
          "units": "-20",
          "resettablePL": "-1744.8000",
          "tradeIDs": [
            "821",
            "823"
          ],
          "averagePrice": "9984.7",
          "pl": "-1744.8000"
        },
        "unrealizedPL": "682.0000",
        "long": {
          "units": "0",
          "resettablePL": "447.6000",
          "unrealizedPL": "0.0000",
          "pl": "447.6000"
        },
        "instrument": "DE30_EUR",
        "resettablePL": "-1297.2000",
        "pl": "-1297.2000"
      }
    ],
    "unrealizedPL": "682.0000",
    "marginCloseoutNAV": "49393.6580",
    "marginUsed": "9948.9000",
    "currency": "EUR",
    "resettablePL": "-1301.0046",
    "NAV": "49377.6580",
```

```
            "marginCloseoutMarginUsed": "9949.8000",
            "id": "101-004-1435156-001",
            "marginCloseoutPositionValue": "198996.0000",
            "openTradeCount": 2,
            "orders": [
              {
                "partialFill": "DEFAULT_FILL",
                "price": "0.87000",
                "stopLossOnFill": {
                  "timeInForce": "GTC",
                  "price": "0.88000"
                },
                "timeInForce": "GTC",
                "clientExtensions": {
                  "comment": "myComment",
                  "id": "myID"
                },
                "id": "204",
                "triggerCondition": "TRIGGER_DEFAULT",
                "replacesOrderID": "200",
                "positionFill": "POSITION_DEFAULT",
                "createTime": "2016-07-08T07:18:47.623211321Z",
                "instrument": "EUR_GBP",
                "state": "PENDING",
                "units": "-50000",
                "type": "LIMIT"
              }
            ],
            "openPositionCount": 1,
            "marginCloseoutPercent": "0.10072",
            "marginCallMarginUsed": "9949.8000",
            "hedgingEnabled": false,
            "positionValue": "198978.0000",
            "pl": "-1301.0046",
            "lastTransactionID": "833",
            "marginAvailable": "39428.7580",
            "marginRate": "0.05",
            "marginCallPercent": "0.20144",
            "pendingOrderCount": 1,
            "withdrawalLimit": "39428.7580",
            "trades": [
              {
                "instrument": "DE30_EUR",
                "financing": "0.0000",
                "openTime": "2016-07-12T09:32:18.062823776Z",
                "initialUnits": "-10",
                "currentUnits": "-10",
                "price": "9984.7",
                "unrealizedPL": "341.0000",
                "realizedPL": "0.0000",
                "state": "OPEN",
                "id": "821"
              },
              {
                "instrument": "DE30_EUR",
                "financing": "0.0000",
                "openTime": "2016-07-12T09:32:18.206929733Z",
                "initialUnits": "-10",
```

```
                "currentUnits": "-10",
                "price": "9984.7",
                "unrealizedPL": "341.0000",
                "realizedPL": "0.0000",
                "state": "OPEN",
                "id": "823"
              }
            ],
            "alias": "hootnotv20",
            "createdByUserID": 1435156,
            "marginCloseoutUnrealizedPL": "698.0000",
            "createdTime": "2016-06-24T21:03:50.914647476Z",
            "balance": "48695.6580"
          },
          "lastTransactionID": "833"
        }
```

class oandapyV20.endpoints.accounts.**AccountSummary**(*accountID*)

Bases: *oandapyV20.endpoints.accounts.Accounts*

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

ENDPOINT = 'v3/accounts/{accountID}/summary'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*)

Instantiate an AccountSummary request.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.accounts as accounts
>>> client = oandapyV20.API(access_token=...)
>>> r = accounts.AccountSummary(accountID)
>>> client.request(r)
>>> print r.response
```

```
{
  "account": {
    "marginCloseoutNAV": "35454.4740",
    "marginUsed": "10581.5000",
    "currency": "EUR",
    "resettablePL": "-13840.3525",
    "NAV": "35454.4740",
    "marginCloseoutMarginUsed": "10581.5000",
    "marginCloseoutPositionValue": "211630.0000",
    "openTradeCount": 2,
    "id": "101-004-1435156-001",
    "hedgingEnabled": false,
    "marginCloseoutPercent": "0.14923",
    "marginCallMarginUsed": "10581.5000",
    "openPositionCount": 1,
    "positionValue": "211630.0000",
    "pl": "-13840.3525",
    "lastTransactionID": "2123",
    "marginAvailable": "24872.9740",
```

```
            "marginRate": "0.05",
            "marginCallPercent": "0.29845",
            "pendingOrderCount": 0,
            "withdrawalLimit": "24872.9740",
            "unrealizedPL": "0.0000",
            "alias": "hootnotv20",
            "createdByUserID": 1435156,
            "marginCloseoutUnrealizedPL": "0.0000",
            "createdTime": "2016-06-24T21:03:50.914647476Z",
            "balance": "35454.4740"
          },
          "lastTransactionID": "2123"
        }
```

class oandapyV20.endpoints.accounts.**Accounts**(*accountID=None*)
Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Accounts - class to handle the accounts endpoints.

ENDPOINT = ‘’

METHOD = ‘GET’

__init__(*accountID=None*)
Instantiate an Accounts APIRequest instance.

Parameters **accountID** (*string (optional)*) – the accountID of the account. Optional when requesting all accounts. For all other requests to the endpoint it is required.

oandapyV20.endpoints.instruments module

Handle instruments endpoints.

class oandapyV20.endpoints.instruments.**Instruments**(*instrument*)
Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Instruments - abstract class to handle instruments endpoint.

ENDPOINT = ‘’

METHOD = ‘GET’

__init__(*instrument*)
Instantiate a Instrument APIRequest instance.

Parameters

- **instrument** (*string (required)*) – the instrument to operate on
- **params** (*dict with query parameters*) –

class oandapyV20.endpoints.instruments.**InstrumentsCandles**(*instrument*, *params=None*)
Bases: *oandapyV20.endpoints.instruments.Instruments*

Get candle data for a specified Instrument.

ENDPOINT = ‘v3/instruments/{instrument}/candles’

EXPECTED_STATUS = 200

METHOD = ‘GET’

__init__(*instrument*, *params=None*)
Instantiate an InstrumentsCandles request.

Parameters

- **instrument** (*string (required)*) – the instrument to fetch candle data for
- **params** (*dict*) – optional request query parameters, check developer.oanda.com for details

Params example:

```
{
  "count": 5,
  "granularity": "M5"
}
```

Candle data example:

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.instruments as instruments
>>> client = oandapyV20.API(access_token=...)
>>> params = ...
>>> r = instruments.InstrumentsCandles(instrument="DE30_EUR",
>>>                                    params=params)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "candles": [
    {
      "volume": 132,
      "mid": {
        "h": "10508.0",
        "c": "10506.0",
        "l": "10503.8",
        "o": "10503.8"
      },
      "complete": true,
      "time": "2016-10-17T19:35:00.000000000Z"
    },
    {
      "volume": 162,
      "mid": {
        "h": "10507.0",
        "c": "10504.9",
        "l": "10502.0",
        "o": "10506.0"
      },
      "complete": true,
      "time": "2016-10-17T19:40:00.000000000Z"
    },
    {
      "volume": 196,
      "mid": {
        "h": "10509.8",
        "c": "10505.0",
        "l": "10502.6",
        "o": "10504.9"
```

```
            },
            "complete": true,
            "time": "2016-10-17T19:45:00.000000000Z"
          },
          {
            "volume": 153,
            "mid": {
              "h": "10510.1",
              "c": "10509.0",
              "l": "10504.2",
              "o": "10505.0"
            },
            "complete": true,
            "time": "2016-10-17T19:50:00.000000000Z"
          },
          {
            "volume": 172,
            "mid": {
              "h": "10509.8",
              "c": "10507.8",
              "l": "10503.2",
              "o": "10509.0"
            },
            "complete": true,
            "time": "2016-10-17T19:55:00.000000000Z"
          }
        ],
        "granularity": "M5",
        "instrument": "DE30/EUR"
      }
```

oandapyV20.endpoints.orders module

Handle orders and pendingOrders endpoints.

class oandapyV20.endpoints.orders.**OrderCancel**(*accountID*, *orderID*)

Bases: *oandapyV20.endpoints.orders.Orders*

Cancel a pending Order in an Account.

ENDPOINT = 'v3/accounts/{accountID}/orders/{orderID}/cancel'

EXPECTED_STATUS = 200

METHOD = 'PUT'

__init__(*accountID*, *orderID*)

Instantiate an OrdersCancel request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **orderID** (*string (required)*) – id of the account to perform the request on.

Example:

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
>>> client = oandapyV20.API(access_token=...)
```

```
>>> r = orders.OrderCancel(accountID= ..., orderID=...)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "orderCancelTransaction": {
    "orderID": "2307",
    "clientOrderID": "myID",
    "reason": "CLIENT_REQUEST",
    "batchID": "2308",
    "time": "2016-10-25T20:53:03.789670387Z",
    "type": "ORDER_CANCEL",
    "userID": 1435156,
    "id": "2308",
    "accountID": "101-004-1435156-001"
  },
  "lastTransactionID": "2308",
  "relatedTransactionIDs": [
    "2308"
  ]
}
```

class oandapyV20.endpoints.orders.**OrderClientExtensions**(*accountID*, *orderID*, *data*)

Bases: *oandapyV20.endpoints.orders.Orders*

Update the Client Extensions for an Order in an Account.

> **Warning:** Do not set, modify or delete clientExtensions if your account is associated with MT4.

ENDPOINT = 'v3/accounts/{accountID}/orders/{orderID}/clientExtensions'

EXPECTED_STATUS = 200

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PUT'

__init__(*accountID*, *orderID*, *data*)

Instantiate an OrderCreate request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **orderID** (*string (required)*) – id of the order to perform the request on.
- **data** (*JSON (required)*) – json orderbody to send

Orderbody example:

```
{
  "clientExtensions": {
    "comment": "myComment",
    "id": "myID"
  }
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
```

```
>>> client = oandapyV20.API(access_token=...)
>>> r = orders.OrderClientExtensions(accountID, orderID, data=data)
>>> client.request(r)
>>> print r.response
```

```
{
  "lastTransactionID": "2305",
  "orderClientExtensionsModifyTransaction": {
    "orderID": "2304",
    "batchID": "2305",
    "clientExtensionsModify": {
      "comment": "myComment",
      "id": "myID"
    },
    "time": "2016-10-25T15:56:43.075594239Z",
    "type": "ORDER_CLIENT_EXTENSIONS_MODIFY",
    "userID": 1435156,
    "id": "2305",
    "accountID": "101-004-1435156-001"
  },
  "relatedTransactionIDs": [
    "2305"
  ]
}
```

class oandapyV20.endpoints.orders.**OrderCreate**(*accountID*, *data*)

Bases: *oandapyV20.endpoints.orders.Orders*

Create an Order for an Account.

ENDPOINT = 'v3/accounts/{accountID}/orders'

EXPECTED_STATUS = 201

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'POST'

__init__(*accountID*, *data*)

Instantiate an OrderCreate request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **data** (*JSON (required)*) – json orderbody to send

Orderbody example:

```
{
  "order": {
    "price": "1.2",
    "stopLossOnFill": {
      "timeInForce": "GTC",
      "price": "1.22"
    },
    "timeInForce": "GTC",
    "instrument": "EUR_USD",
    "units": "-100",
    "type": "LIMIT",
    "positionFill": "DEFAULT"
```

```
    }
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
>>> client = oandapyV20.API(access_token=...)
>>> r = orders.OrderCreate(accountID, data=data)
>>> client.request(r)
>>> print r.response
```

```
{
  "orderCreateTransaction": {
    "price": "1.20000",
    "stopLossOnFill": {
      "timeInForce": "GTC",
      "price": "1.22000"
    },
    "timeInForce": "GTC",
    "reason": "CLIENT_ORDER",
    "id": "2304",
    "batchID": "2304",
    "triggerCondition": "TRIGGER_DEFAULT",
    "positionFill": "DEFAULT",
    "userID": 1435156,
    "instrument": "EUR_USD",
    "time": "2016-10-24T21:48:18.593753865Z",
    "units": "-100",
    "type": "LIMIT_ORDER",
    "accountID": "101-004-1435156-001"
  },
  "lastTransactionID": "2304",
  "relatedTransactionIDs": [
    "2304"
  ]
}
```

class oandapyV20.endpoints.orders.**OrderDetails**(*accountID*, *orderID*)

Bases: *oandapyV20.endpoints.orders.Orders*

Get details for a single Order in an Account.

ENDPOINT = 'v3/accounts/{accountID}/orders/{orderID}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *orderID*)

Instantiate an OrderDetails request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **orderID** (*string (required)*) – id of the order to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
>>> client = oandapyV20.API(access_token=...)
>>> r = orders.OrderDetails(accountID=..., orderID=...)
```

```
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "order": {
    "partialFill": "DEFAULT_FILL",
    "price": "1.20000",
    "stopLossOnFill": {
      "timeInForce": "GTC",
      "price": "1.22000"
    },
    "timeInForce": "GTC",
    "createTime": "2016-10-25T21:07:21.065554321Z",
    "triggerCondition": "TRIGGER_DEFAULT",
    "positionFill": "POSITION_DEFAULT",
    "id": "2309",
    "instrument": "EUR_USD",
    "state": "PENDING",
    "units": "-100",
    "type": "LIMIT"
  },
  "lastTransactionID": "2309"
}
```

class oandapyV20.endpoints.orders.**OrderList**(*accountID*, *params=None*)

Bases: *oandapyV20.endpoints.orders.Orders*

Create an Order for an Account.

ENDPOINT = 'v3/accounts/{accountID}/orders'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *params=None*)

Instantiate an OrderList request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict*) – optional request query parameters, check developer.oanda.com for details

Example:

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
>>> client = oandapyV20.API(access_token=...)
>>> r = orders.OrderList(accountID)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "orders": [
    {
      "partialFill": "DEFAULT_FILL",
```

```
            "price": "1.20000",
            "stopLossOnFill": {
              "timeInForce": "GTC",
              "price": "1.22000"
            },
            "timeInForce": "GTC",
            "createTime": "2016-10-05T10:25:47.627003645Z",
            "triggerCondition": "TRIGGER_DEFAULT",
            "positionFill": "POSITION_DEFAULT",
            "id": "2125",
            "instrument": "EUR_USD",
            "state": "PENDING",
            "units": "-100",
            "type": "LIMIT"
          }
        ],
        "lastTransactionID": "2129"
      }
```

class oandapyV20.endpoints.orders.**OrderReplace**(*accountID*, *orderID*, *data*)

Bases: *oandapyV20.endpoints.orders.Orders*

OrderReplace.

Replace an Order in an Account by simultaneously cancelling it and createing a replacement Order.

ENDPOINT = 'v3/accounts/{accountID}/orders/{orderID}'

EXPECTED_STATUS = 201

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PUT'

__init__(*accountID*, *orderID*, *data*)

Instantiate an OrderReplace request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **orderID** (*string (required)*) – id of the order to perform the request on.
- **data** (*JSON (required)*) – json orderbody to send

Orderbody example:

```
{
  "order": {
    "units": "-500000",
    "instrument": "EUR_USD",
    "price": "1.25000",
    "type": "LIMIT"
  }
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
>>> client = oandapyV20.API(access_token=...)
>>> data =
        {
          "order": {
            "units": "-500000",
```

```
                    "instrument": "EUR_USD",
                    "price": "1.25000",
                    "type": "LIMIT"
                }
            }
```

```
>>> r = orders.OrderReplace(accountID=..., orderID=..., data=data)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "orderCreateTransaction": {
    "price": "1.25000",
    "timeInForce": "GTC",
    "reason": "REPLACEMENT",
    "clientExtensions": {
      "comment": "myComment",
      "id": "myID"
    },
    "id": "2307",
    "batchID": "2306",
    "triggerCondition": "TRIGGER_DEFAULT",
    "replacesOrderID": "2304",
    "positionFill": "DEFAULT",
    "userID": 1435156,
    "instrument": "EUR_USD",
    "time": "2016-10-25T19:45:38.558056359Z",
    "units": "-500000",
    "type": "LIMIT_ORDER",
    "accountID": "101-004-1435156-001"
  },
  "orderCancelTransaction": {
    "orderID": "2304",
    "clientOrderID": "myID",
    "reason": "CLIENT_REQUEST_REPLACED",
    "batchID": "2306",
    "time": "2016-10-25T19:45:38.558056359Z",
    "type": "ORDER_CANCEL",
    "replacedByOrderID": "2307",
    "userID": 1435156,
    "id": "2306",
    "accountID": "101-004-1435156-001"
  },
  "lastTransactionID": "2307",
  "relatedTransactionIDs": [
    "2306",
    "2307"
  ]
}
```

class oandapyV20.endpoints.orders.**Orders**(*accountID*, *orderID=None*)

Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Orders - abstract base class to handle the orders endpoints.

ENDPOINT = ''

EXPECTED_STATUS = 0

METHOD = 'GET'

__init__(*accountID*, *orderID=None*)
Instantiate an Orders request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **orderID** (*string*) – id of the order to perform the request for.

class oandapyV20.endpoints.orders.**OrdersPending**(*accountID*)
Bases: *oandapyV20.endpoints.orders.Orders*

List all pending Orders in an Account.

ENDPOINT = 'v3/accounts/{accountID}/pendingOrders'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*)
Instantiate an OrdersPending request.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

Example:

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.orders as orders
>>> client = oandapyV20.API(access_token=...)
>>> r = orders.OrdersPending(accountID)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "orders": [
    {
      "partialFill": "DEFAULT_FILL",
      "price": "1.20000",
      "stopLossOnFill": {
        "timeInForce": "GTC",
        "price": "1.22000"
      },
      "timeInForce": "GTC",
      "clientExtensions": {
        "comment": "myComment",
        "id": "myID"
      },
      "id": "2304",
      "triggerCondition": "TRIGGER_DEFAULT",
      "positionFill": "POSITION_DEFAULT",
      "createTime": "2016-10-24T21:48:18.593753865Z",
      "instrument": "EUR_USD",
      "state": "PENDING",
      "units": "-100",
      "type": "LIMIT"
    }
  ],
```

```
        "lastTransactionID": "2305"
    }
```

oandapyV20.endpoints.positions module

Handle position endpoints.

class oandapyV20.endpoints.positions.**OpenPositions**(*accountID*)
Bases: *oandapyV20.endpoints.positions.Positions*

OpenPositions.

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

ENDPOINT = 'v3/accounts/{accountID}/openPositions'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*)
Instantiate an OpenPositions request.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.positions as positions
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> r = positions.OpenPositions(accountID=accountID)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "positions": [
    {
      "short": {
        "units": "0",
        "resettablePL": "-14164.3000",
        "unrealizedPL": "0.0000",
        "pl": "-14164.3000"
      },
      "unrealizedPL": "-284.0000",
      "long": {
        "unrealizedPL": "-284.0000",
        "tradeIDs": [
          "2315"
        ],
        "resettablePL": "404.5000",
        "units": "10",
        "averagePrice": "10678.3",
        "pl": "404.5000"
      },
      "instrument": "DE30_EUR",
      "resettablePL": "-13759.8000",
      "pl": "-13759.8000"
```

```
            },
            {
              "short": {
                "unrealizedPL": "-0.0738",
                "tradeIDs": [
                  "2323"
                ],
                "resettablePL": "0.0000",
                "units": "-100",
                "averagePrice": "1.09843",
                "pl": "0.0000"
              },
              "unrealizedPL": "-0.0738",
              "long": {
                "units": "0",
                "resettablePL": "-44.6272",
                "unrealizedPL": "0.0000",
                "pl": "-44.6272"
              },
              "instrument": "EUR_USD",
              "resettablePL": "-44.6272",
              "pl": "-44.6272"
            }
          ],
          "lastTransactionID": "2327"
        }
```

class oandapyV20.endpoints.positions.**PositionClose**(*accountID*, *instrument*, *data*)

Bases: *oandapyV20.endpoints.positions.Positions*

Closeout the open Position regarding instrument in an Account.

ENDPOINT = 'v3/accounts/{accountID}/positions/{instrument}/close'

EXPECTED_STATUS = 200

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PUT'

__init__(*accountID*, *instrument*, *data*)

Instantiate a PositionClose request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **instrument** (*string (required)*) – instrument to close partially or fully.
- **data** (*dict (required)*) – closeout specification data to send, check developer.oanda.com for details.

Data body example:

```
        {
          "longUnits": "ALL"
        }
```

```
        >>> import oandapyV20
        >>> import oandapyV20.endpoints.positions as positions
        >>> accountID = ...
        >>> instrument = ...
```

```
>>> client = oandapyV20.API(access_token=...)
>>> data =
        {
          "longUnits": "ALL"
        }
```

```
>>> r = positions.PositionClose(accountID=accountID,
>>>                             instrument=instrument,
>>>                             data=data)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "longOrderCreateTransaction": {
    "longPositionCloseout": {
      "units": "ALL",
      "instrument": "EUR_USD"
    },
    "batchID": "6390",
    "reason": "POSITION_CLOSEOUT",
    "id": "6390",
    "timeInForce": "FOK",
    "positionFill": "REDUCE_ONLY",
    "userID": "<USERID>",
    "instrument": "EUR_USD",
    "time": "2016-06-22T18:41:35.034041665Z",
    "units": "-251",
    "type": "MARKET_ORDER",
    "accountID": "<ACCOUNT>"
  },
  "relatedTransactionIDs": [
    "6390",
    "6391"
  ],
  "lastTransactionID": "6391",
  "longOrderFillTransaction": {
    "price": "1.13018",
    "batchID": "6390",
    "accountBalance": "43650.69807",
    "reason": "MARKET_ORDER_POSITION_CLOSEOUT",
    "tradesClosed": [
      {
        "units": "-1",
        "financing": "0.00000",
        "realizedPL": "-0.00013",
        "tradeID": "6383"
      },
      {
        "units": "-250",
        "financing": "0.00000",
        "realizedPL": "-0.03357",
        "tradeID": "6385"
      }
    ],
    "id": "6391",
    "orderID": "6390",
```

```
            "financing": "0.00000",
            "userID": "<USERID>",
            "instrument": "EUR_USD",
            "time": "2016-06-22T18:41:35.034041665Z",
            "units": "-251",
            "type": "ORDER_FILL",
            "pl": "-0.03370",
            "accountID": "<ACCOUNT>"
          }
        }
```

class oandapyV20.endpoints.positions.**PositionDetails**(*accountID*, *instrument*)

Bases: *oandapyV20.endpoints.positions.Positions*

PositionDetails.

Get the details of a single instrument's position in an Account. The position may be open or not.

ENDPOINT = 'v3/accounts/{accountID}/positions/{instrument}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *instrument*)

Instantiate a PositionDetails request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **instrument** (*string (required)*) – id of the instrument to get the position details for.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.positions as positions
>>> accountID = ...
>>> instrument = ...
>>> client = oandapyV20.API(access_token=...)
>>> r = positions.PositionDetails(accountID=accountID, instrument)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "position": {
    "short": {
      "unrealizedPL": "-0.0738",
      "tradeIDs": [
        "2323"
      ],
      "resettablePL": "0.0000",
      "units": "-100",
      "averagePrice": "1.09843",
      "pl": "0.0000"
    },
    "unrealizedPL": "-0.0738",
    "long": {
      "units": "0",
      "resettablePL": "-44.6272",
      "unrealizedPL": "0.0000",
```

```
            "pl": "-44.6272"
          },
          "instrument": "EUR_USD",
          "resettablePL": "-44.6272",
          "pl": "-44.6272"
        },
        "lastTransactionID": "2327"
      }
```

class oandapyV20.endpoints.positions.**PositionList**(*accountID*)

Bases: *oandapyV20.endpoints.positions.Positions*

PositionList.

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

ENDPOINT = 'v3/accounts/{accountID}/positions'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*)

Instantiate a PositionList request.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.positions as positions
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> r = positions.PositionList(accountID=accountID)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "positions": [
    {
      "short": {
        "units": "0",
        "resettablePL": "-272.6805",
        "unrealizedPL": "0.0000",
        "pl": "-272.6805"
      },
      "unrealizedPL": "0.0000",
      "long": {
        "units": "0",
        "resettablePL": "0.0000",
        "unrealizedPL": "0.0000",
        "pl": "0.0000"
      },
      "instrument": "EUR_GBP",
      "resettablePL": "-272.6805",
      "pl": "-272.6805"
    },
    {
      "short": {
```

```
                "unrealizedPL": "870.0000",
                "tradeIDs": [
                  "2121",
                  "2123"
                ],
                "resettablePL": "-13959.3000",
                "units": "-20",
                "averagePrice": "10581.5",
                "pl": "-13959.3000"
              },
              "unrealizedPL": "870.0000",
              "long": {
                "units": "0",
                "resettablePL": "404.5000",
                "unrealizedPL": "0.0000",
                "pl": "404.5000"
              },
              "instrument": "DE30_EUR",
              "resettablePL": "-13554.8000",
              "pl": "-13554.8000"
            },
            {
              "short": {
                "units": "0",
                "resettablePL": "0.0000",
                "unrealizedPL": "0.0000",
                "pl": "0.0000"
              },
              "unrealizedPL": "0.0000",
              "long": {
                "units": "0",
                "resettablePL": "-12.8720",
                "unrealizedPL": "0.0000",
                "pl": "-12.8720"
              },
              "instrument": "EUR_USD",
              "resettablePL": "-12.8720",
              "pl": "-12.8720"
            }
          ],
          "lastTransactionID": "2124"
        }
```


class oandapyV20.endpoints.positions.**Positions**(*accountID*, *instrument=None*)

Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Positions - abstractbase class to handle the 'positions' endpoints.

ENDPOINT = ''

METHOD = 'GET'

__init__(*accountID*, *instrument=None*)

Instantiate a Positions APIRequest instance.

Parameters

- **accountID** (*string (required)*) – the id of the account to perform the request on.
- **instrument** (*string (optional)*) – the instrument for the Positions request

oandapyV20.endpoints.pricing module

Handle pricing endpoints.

class oandapyV20.endpoints.pricing.**Pricing**(*accountID*)
Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Pricing - class to handle pricing endpoint.

ENDPOINT = ''

METHOD = 'GET'

__init__(*accountID*)
Instantiate a Pricing APIRequest instance.

Parameters **accountID** (*string (required)*) – the accountID of the account.

class oandapyV20.endpoints.pricing.**PricingInfo**(*accountID*, *params=None*)
Bases: *oandapyV20.endpoints.pricing.Pricing*

Pricing.

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

ENDPOINT = 'v3/accounts/{accountID}/pricing'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *params=None*)
Instantiate a PricingStream APIRequest instance.

Parameters

- **accountID** (*string (required)*) – the accountID of the account.
- **params** (*dict (required)*) – parameters for the request, check developer.oanda.com for details.

Example

```
>>> import oandapyV20
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.pricing as pricing
>>> accountID = "..."
>>> api = API(access_token="...")
>>> params =
        {
          "instruments": "EUR_USD,EUR_JPY"
        }
```

```
>>> r = pricing.PricingInfo(accountID=accountID, params=params)
>>> rv = api.request(r)
>>> print r.response
```

Output:

```
{
  "prices": [
    {
```

```
      "status": "tradeable",
      "instrument": "EUR_USD",
      "quoteHomeConversionFactors": {
        "negativeUnits": "0.89160730",
        "positiveUnits": "0.89150397"
      },
      "asks": [
        {
          "price": "1.12170",
          "liquidity": 10000000
        },
        {
          "price": "1.12172",
          "liquidity": 10000000
        }
      ],
      "time": "2016-10-05T05:28:16.729643492Z",
      "closeoutAsk": "1.12174",
      "bids": [
        {
          "price": "1.12157",
          "liquidity": 10000000
        },
        {
          "price": "1.12155",
          "liquidity": 10000000
        }
      ],
      "closeoutBid": "1.12153",
      "unitsAvailable": {
        "default": {
          "short": "506246",
          "long": "506128"
        },
        "reduceOnly": {
          "short": "0",
          "long": "0"
        },
        "openOnly": {
          "short": "506246",
          "long": "506128"
        },
        "reduceFirst": {
          "short": "506246",
          "long": "506128"
        }
      }
    },
    {
      "status": "tradeable",
      "instrument": "EUR_JPY",
      "quoteHomeConversionFactors": {
        "negativeUnits": "0.00867085",
        "positiveUnits": "0.00866957"
      },
      "asks": [
        {
          "price": "115.346",
```

```
                "liquidity": 1000000
              },
              {
                "price": "115.347",
                "liquidity": 2000000
              },
              {
                "price": "115.348",
                "liquidity": 5000000
              },
              {
                "price": "115.350",
                "liquidity": 10000000
              }
            ],
            "time": "2016-10-05T05:28:15.621238671Z",
            "closeoutAsk": "115.350",
            "bids": [
              {
                "price": "115.329",
                "liquidity": 1000000
              },
              {
                "price": "115.328",
                "liquidity": 2000000
              },
              {
                "price": "115.327",
                "liquidity": 5000000
              },
              {
                "price": "115.325",
                "liquidity": 10000000
              }
            ],
            "closeoutBid": "115.325",
            "unitsAvailable": {
              "default": {
                "short": "506262",
                "long": "506112"
              },
              "reduceOnly": {
                "short": "0",
                "long": "0"
              },
              "openOnly": {
                "short": "506262",
                "long": "506112"
              },
              "reduceFirst": {
                "short": "506262",
                "long": "506112"
              }
            }
          }
        ]
      }
```

class oandapyV20.endpoints.pricing.**PricingStream**(*accountID*, *params=None*)
Bases: *oandapyV20.endpoints.pricing.Pricing*

PricingStream.

Get realtime pricing information for a specified list of Instruments.

ENDPOINT = 'v3/accounts/{accountID}/pricing/stream'

EXPECTED_STATUS = 200

METHOD = 'GET'

STREAM = True

__init__(*accountID*, *params=None*)
Instantiate a PricingStream APIRequest instance.

Parameters

- **accountID** (*string (required)*) – the accountID of the account.
- **params** (*dict (required)*) – parameters for the request, check developer.oanda.com for details.

Example

```
>>> import oandapyV20
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.pricing as pricing
>>> accountID = "..."
>>> api = API(access_token="...")
>>> params =
        {
          "instruments": "EUR_USD,EUR_JPY"
        }
```

```
>>> r = pricing.PricingStream(accountID=accountID, params=params)
>>> rv = api.request(r)
>>> maxrecs = 100
>>> for ticks in r:
>>>     print json.dumps(R, indent=4),","
>>>     if maxrecs == 0:
>>>         r.terminate("maxrecs records received")
```

Output:

```
{
  "status": "tradeable",
  "asks": [
    {
      "price": "114.312",
      "liquidity": 1000000
    },
    {
      "price": "114.313",
      "liquidity": 2000000
    },
    {
      "price": "114.314",
      "liquidity": 5000000
```

```
            },
            {
              "price": "114.316",
              "liquidity": 10000000
            }
          ],
          "closeoutBid": "114.291",
          "bids": [
            {
              "price": "114.295",
              "liquidity": 1000000
            },
            {
              "price": "114.294",
              "liquidity": 2000000
            },
            {
              "price": "114.293",
              "liquidity": 5000000
            },
            {
              "price": "114.291",
              "liquidity": 10000000
            }
          ],
          "instrument": "EUR_JPY",
          "time": "2016-10-27T08:38:43.094548890Z",
          "closeoutAsk": "114.316"
        },
        {
          "type": "HEARTBEAT",
          "time": "2016-10-27T08:38:44.327443673Z"
        },
        {
          "status": "tradeable",
          "asks": [
            {
              "price": "1.09188",
              "liquidity": 10000000
            },
            {
              "price": "1.09190",
              "liquidity": 10000000
            }
          ],
          "closeoutBid": "1.09173",
          "bids": [
            {
              "price": "1.09177",
              "liquidity": 10000000
            },
            {
              "price": "1.09175",
              "liquidity": 10000000
            }
          ],
          "instrument": "EUR_USD",
          "time": "2016-10-27T08:38:45.664613867Z",
```

```
      "closeoutAsk": "1.09192"
    },
    {
      "status": "tradeable",
      "asks": [
        {
          "price": "114.315",
          "liquidity": 1000000
        },
        {
          "price": "114.316",
          "liquidity": 2000000
        },
        {
          "price": "114.317",
          "liquidity": 5000000
        },
        {
          "price": "114.319",
          "liquidity": 10000000
        }
      ],
      "closeoutBid": "114.294",
      "bids": [
        {
          "price": "114.298",
          "liquidity": 1000000
        },
        {
          "price": "114.297",
          "liquidity": 2000000
        },
        {
          "price": "114.296",
          "liquidity": 5000000
        },
        {
          "price": "114.294",
          "liquidity": 10000000
        }
      ],
      "instrument": "EUR_JPY",
      "time": "2016-10-27T08:38:45.681572782Z",
      "closeoutAsk": "114.319"
    }
```


terminate(*message=''*)
terminate the stream.

Calling this method will stop the generator yielding tickrecords. A message can be passed optionally.

oandapyV20.endpoints.trades module

Handle trades endpoints.

class oandapyV20.endpoints.trades.**OpenTrades**(*accountID*)
Bases: *oandapyV20.endpoints.trades.Trades*

Get the list of open Trades for an Account.

ENDPOINT = 'v3/accounts/{accountID}/openTrades'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*)
Instantiate an OpenTrades request.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.trades as trades
>>> client = oandapyV20.API(access_token=...)
>>> r = trades.OpenTrades(accountID=...)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "trades": [
    {
      "instrument": "DE30_EUR",
      "financing": "0.0000",
      "openTime": "2016-10-28T14:28:05.231759081Z",
      "initialUnits": "10",
      "currentUnits": "10",
      "price": "10678.3",
      "unrealizedPL": "136.0000",
      "realizedPL": "0.0000",
      "state": "OPEN",
      "id": "2315"
    }
  ],
  "lastTransactionID": "2317"
}
```

class oandapyV20.endpoints.trades.**TradeCRCDO**(*accountID*, *tradeID*, *data*)
Bases: *oandapyV20.endpoints.trades.Trades*

Trade Create Replace Cancel Dependent Orders.

ENDPOINT = 'v3/accounts/{accountID}/trades/{tradeID}/orders'

EXPECTED_STATUS = 200

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PUT'

__init__(*accountID*, *tradeID*, *data*)
Instantiate a TradeClientExtensions request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **tradeID** (*string (required)*) – id of the trade to update client extensions for.

- **data** (*dict (required)*) – clientextension data to send, check developer.oanda.com for details.

Data body example:

```
{
  "takeProfit": {
    "timeInForce": "GTC",
    "price": "1.05"
  },
  "stopLoss": {
    "timeInForce": "GTC",
    "price": "1.10"
  }
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.trades as trades
>>> accountID = ...
>>> tradeID = ...
>>> client = oandapyV20.API(access_token=...)
>>> data =
        {
          "takeProfit": {
            "timeInForce": "GTC",
            "price": "1.05"
          },
          "stopLoss": {
            "timeInForce": "GTC",
            "price": "1.10"
          }
        }
```

```
>>> r = trades.TradeCRCDO(accountID=accountID,
>>>                       tradeID=tradeID,
>>>                       data=data)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "lastTransactionID": "2327",
  "stopLossOrderCancelTransaction": {
    "orderID": "2324",
    "batchID": "2325",
    "reason": "CLIENT_REQUEST_REPLACED",
    "time": "2016-10-28T21:00:19.978476830Z",
    "type": "ORDER_CANCEL",
    "replacedByOrderID": "2327",
    "userID": 1435156,
    "id": "2326",
    "accountID": "101-004-1435156-001"
  },
  "stopLossOrderTransaction": {
    "tradeID": "2323",
    "price": "1.10000",
    "timeInForce": "GTC",
    "reason": "REPLACEMENT",
```

```
            "id": "2327",
            "batchID": "2325",
            "triggerCondition": "TRIGGER_DEFAULT",
            "replacesOrderID": "2324",
            "userID": 1435156,
            "time": "2016-10-28T21:00:19.978476830Z",
            "cancellingTransactionID": "2326",
            "type": "STOP_LOSS_ORDER",
            "accountID": "101-004-1435156-001"
          },
          "relatedTransactionIDs": [
            "2325",
            "2326",
            "2327"
          ],
          "takeProfitOrderTransaction": {
            "tradeID": "2323",
            "price": "1.05000",
            "timeInForce": "GTC",
            "reason": "CLIENT_ORDER",
            "id": "2325",
            "batchID": "2325",
            "triggerCondition": "TRIGGER_DEFAULT",
            "userID": 1435156,
            "time": "2016-10-28T21:00:19.978476830Z",
            "type": "TAKE_PROFIT_ORDER",
            "accountID": "101-004-1435156-001"
          }
        }
```

class oandapyV20.endpoints.trades.**TradeClientExtensions**(*accountID*, *tradeID*, *data=None*)

Bases: *oandapyV20.endpoints.trades.Trades*

TradeClientExtensions.

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

ENDPOINT = 'v3/accounts/{accountID}/trades/{tradeID}/clientExtensions'

EXPECTED_STATUS = 200

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PUT'

__init__(*accountID*, *tradeID*, *data=None*)

Instantiate a TradeClientExtensions request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **tradeID** (*string (required)*) – id of the trade to update client extensions for.
- **data** (*dict (required)*) – clientextension data to send, check developer.oanda.com for details.

Data body example:

```
{
  "clientExtensions": {
    "comment": "myComment",
    "id": "myID2315"
  }
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.trades as trades
>>> accountID = ...
>>> tradeID = ...
>>> client = oandapyV20.API(access_token=...)
>>> data =
        {
          "clientExtensions": {
            "comment": "myComment",
            "id": "myID2315"
          }
        }
```

```
>>> r = trades.TradeClientExtensions(accountID=accountID,
>>>                                  tradeID=tradeID,
>>>                                  data=data)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "tradeClientExtensionsModifyTransaction": {
    "batchID": "2319",
    "tradeID": "2315",
    "time": "2016-10-28T20:32:39.356516787Z",
    "tradeClientExtensionsModify": {
      "comment": "myComment",
      "id": "myID2315"
    },
    "type": "TRADE_CLIENT_EXTENSIONS_MODIFY",
    "userID": 1435156,
    "id": "2319",
    "accountID": "101-004-1435156-001"
  },
  "lastTransactionID": "2319",
  "relatedTransactionIDs": [
    "2319"
  ]
}
```

class oandapyV20.endpoints.trades.**TradeClose**(*accountID*, *tradeID*, *data=None*)

Bases: *oandapyV20.endpoints.trades.Trades*

TradeClose.

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

ENDPOINT = 'v3/accounts/{accountID}/trades/{tradeID}/close'

EXPECTED_STATUS = 200

HEADERS = {'Content-Type': 'application/json'}

METHOD = 'PUT'

__init__(*accountID*, *tradeID*, *data=None*)
Instantiate a TradeClose request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **tradeID** (*string (required)*) – id of the trade to close.
- **data** (*dict (optional)*) – data to send, use this to close a trade partially. Check developer.oanda.com for details.

Data body example:

```
{
  "units": 100
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.trades as trades
>>> client = oandapyV20.API(access_token=...)
>>> data =
        {
          "units": 100
        }
```

```
>>> r = trades.TradeClose(accountID=..., data=data)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "orderFillTransaction": {
    "price": "1.09289",
    "batchID": "2316",
    "accountBalance": "33848.1208",
    "reason": "MARKET_ORDER_TRADE_CLOSE",
    "tradesClosed": [
      {
        "units": "-100",
        "financing": "0.0000",
        "realizedPL": "-0.1455",
        "tradeID": "2313"
      }
    ],
    "id": "2317",
    "orderID": "2316",
    "financing": "0.0000",
    "userID": 1435156,
    "instrument": "EUR_USD",
    "time": "2016-10-28T15:11:58.023004583Z",
    "units": "-100",
    "type": "ORDER_FILL",
    "pl": "-0.1455",
    "accountID": "101-004-1435156-001"
  },
  "orderCreateTransaction": {
    "timeInForce": "FOK",
```

```
            "reason": "TRADE_CLOSE",
            "tradeClose": {
              "units": "100",
              "tradeID": "2313"
            },
            "id": "2316",
            "batchID": "2316",
            "positionFill": "REDUCE_ONLY",
            "userID": 1435156,
            "instrument": "EUR_USD",
            "time": "2016-10-28T15:11:58.023004583Z",
            "units": "-100",
            "type": "MARKET_ORDER",
            "accountID": "101-004-1435156-001"
          },
          "lastTransactionID": "2317",
          "relatedTransactionIDs": [
            "2316",
            "2317"
          ]
        }
```

class oandapyV20.endpoints.trades.**TradeDetails**(*accountID*, *tradeID*)

Bases: *oandapyV20.endpoints.trades.Trades*

Get the details of a specific Trade in an Account.

ENDPOINT = 'v3/accounts/{accountID}/trades/{tradeID}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *tradeID*)

Instantiate a TradeDetails request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **tradeID** (*string (required)*) – id of the trade.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.trades as trades
>>> client = oandapyV20.API(access_token=...)
>>> r = accounts.TradeDetails(accountID=..., tradeID=...)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "lastTransactionID": "2317",
  "trade": {
    "instrument": "DE30_EUR",
    "financing": "0.0000",
    "openTime": "2016-10-28T14:28:05.231759081Z",
    "initialUnits": "10",
    "currentUnits": "10",
    "price": "10678.3",
    "unrealizedPL": "226.0000",
    "realizedPL": "0.0000",
```

```
            "state": "OPEN",
            "id": "2315"
          }
        }
```

class oandapyV20.endpoints.trades.**Trades**(*accountID*, *tradeID=None*)
Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Trades - abstract baseclass to handle the trades endpoints.

ENDPOINT = ‘’

METHOD = ‘GET’

__init__(*accountID*, *tradeID=None*)
Instantiate a Trades APIRequest instance.

Parameters

- **accountID** (*string*) – the account_id of the account.
- **tradeID** (*string*) – ID of the trade

class oandapyV20.endpoints.trades.**TradesList**(*accountID*, *params=None*)
Bases: *oandapyV20.endpoints.trades.Trades*

Get a list of trades for an Account.

ENDPOINT = ‘v3/accounts/{accountID}/trades’

EXPECTED_STATUS = 200

METHOD = ‘GET’

__init__(*accountID*, *params=None*)
Instantiate a TradesList request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict (optional)*) – query params to send, check developer.oanda.com for details.

Query Params example:

```
{
  "instrument": "DE30_EUR,EUR_USD"
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.trades as trades
>>> client = oandapyV20.API(access_token=...)
>>> params =
        {
          "instrument": "DE30_EUR,EUR_USD"
        }
```

```
>>> r = trades.TradesList(accountID=..., params=params)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "trades": [
    {
      "instrument": "DE30_EUR",
      "financing": "0.0000",
      "openTime": "2016-10-28T14:28:05.231759081Z",
      "initialUnits": "10",
      "currentUnits": "10",
      "price": "10678.3",
      "unrealizedPL": "25.0000",
      "realizedPL": "0.0000",
      "state": "OPEN",
      "id": "2315"
    },
    {
      "instrument": "EUR_USD",
      "financing": "0.0000",
      "openTime": "2016-10-28T14:27:19.011002322Z",
      "initialUnits": "100",
      "currentUnits": "100",
      "price": "1.09448",
      "unrealizedPL": "-0.0933",
      "realizedPL": "0.0000",
      "state": "OPEN",
      "id": "2313"
    }
  ],
  "lastTransactionID": "2315"
}
```

oandapyV20.endpoints.transactions module

Handle transactions endpoints.

class oandapyV20.endpoints.transactions.**TransactionDetails**(*accountID*, *transactionID*)

Bases: *oandapyV20.endpoints.transactions.Transactions*

Get the details of a single Account Transaction.

ENDPOINT = 'v3/accounts/{accountID}/transactions/{transactionID}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *transactionID*)

Instantiate a TransactionDetails request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **transactionID** (*string (required)*) – id of the transaction

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.transactions as trans
>>> client = oandapyV20.API(access_token=...)
>>> r = trans.TransactionDetails(accountID=..., transactionID=...)
```

```
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "transaction": {
    "price": "1.20000",
    "stopLossOnFill": {
      "timeInForce": "GTC",
      "price": "1.22000"
    },
    "timeInForce": "GTC",
    "reason": "CLIENT_ORDER",
    "id": "2304",
    "batchID": "2304",
    "triggerCondition": "TRIGGER_DEFAULT",
    "positionFill": "DEFAULT",
    "userID": 1435156,
    "instrument": "EUR_USD",
    "time": "2016-10-24T21:48:18.593753865Z",
    "units": "-100",
    "type": "LIMIT_ORDER",
    "accountID": "101-004-1435156-001"
  },
  "lastTransactionID": "2311"
}
```

class oandapyV20.endpoints.transactions.**TransactionIDRange**(*accountID*, *params=None*)

Bases: *oandapyV20.endpoints.transactions.Transactions*

TransactionIDRange.

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

ENDPOINT = 'v3/accounts/{accountID}/transactions/idrange'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *params=None*)

Instantiate an TransactionIDRange request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict (required)*) – query params to send, check developer.oanda.com for details.

Query Params example:

```
{
  "to": 2306,
  "from": 2304
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.transactions as trans
>>> client = oandapyV20.API(access_token=...)
```

```
>>> params =
        {
          "to": 2306,
          "from": 2304
        }
```

```
>>> r = trans.TransactionIDRange(accountID=..., params=params)
>>> client.request(r)
>>> print r.response
```

Output:


```
{
  "lastTransactionID": "2311",
  "transactions": [
    {
      "price": "1.20000",
      "stopLossOnFill": {
        "timeInForce": "GTC",
        "price": "1.22000"
      },
      "timeInForce": "GTC",
      "reason": "CLIENT_ORDER",
      "id": "2304",
      "batchID": "2304",
      "triggerCondition": "TRIGGER_DEFAULT",
      "positionFill": "DEFAULT",
      "userID": 1435156,
      "instrument": "EUR_USD",
      "time": "2016-10-24T21:48:18.593753865Z",
      "units": "-100",
      "type": "LIMIT_ORDER",
      "accountID": "101-004-1435156-001"
    },
    {
      "orderID": "2304",
      "batchID": "2305",
      "clientExtensionsModify": {
        "comment": "myComment",
        "id": "myID"
      },
      "time": "2016-10-25T15:56:43.075594239Z",
      "type": "ORDER_CLIENT_EXTENSIONS_MODIFY",
      "userID": 1435156,
      "id": "2305",
      "accountID": "101-004-1435156-001"
    },
    {
      "orderID": "2304",
      "clientOrderID": "myID",
      "reason": "CLIENT_REQUEST_REPLACED",
      "batchID": "2306",
      "time": "2016-10-25T19:45:38.558056359Z",
      "type": "ORDER_CANCEL",
      "replacedByOrderID": "2307",
      "userID": 1435156,
      "id": "2306",
      "accountID": "101-004-1435156-001"
    }
```

```
    ]
}
```

class `oandapyV20.endpoints.transactions.`**`TransactionList`**(*accountID*, *params=None*)
Bases: `oandapyV20.endpoints.transactions.Transactions`

TransactionList.

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

`ENDPOINT` = **'v3/accounts/{accountID}/transactions'**

`EXPECTED_STATUS` = **200**

`METHOD` = **'GET'**

`__init__`(*accountID*, *params=None*)
Instantiate a TransactionList request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict (optional)*) – query params to send, check developer.oanda.com for details.

Query Params example:

```
{
  "pageSize": 200
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.transactions as trans
>>> client = oandapyV20.API(access_token=...)
>>> r = trans.TransactionList(accountID)  # params optional
>>> client.request(r)
>>> print r.response
```

Output:


```
{
  "count": 2124,
  "from": "2016-06-24T21:03:50.914647476Z",
  "lastTransactionID": "2124",
  "pageSize": 100,
  "to": "2016-10-05T06:54:14.025946546Z",
  "pages": [
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
```

```
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
        "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/transactions/idrange?f
      ]
    }
```

class oandapyV20.endpoints.transactions.**Transactions**(*accountID*, *transactionID=None*)

Bases: *oandapyV20.endpoints.apirequest.APIRequest*

Transactions - abstract baseclass to handle transaction endpoints.

ENDPOINT = ''

METHOD = 'GET'

__init__(*accountID*, *transactionID=None*)

Instantiate a Transactions APIRequest instance.

Parameters

- **accountID** (*string (required)*) – the id of the account.
- **transactionID** (*string*) – the id of the transaction

class oandapyV20.endpoints.transactions.**TransactionsSinceID**(*accountID*, *params=None*)

Bases: *oandapyV20.endpoints.transactions.Transactions*

TransactionsSinceID.

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

ENDPOINT = 'v3/accounts/{accountID}/transactions/sinceid'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*accountID*, *params=None*)

Instantiate an TransactionsSince request.

Parameters

- **accountID** (*string (required)*) – id of the account to perform the request on.
- **params** (*dict (required)*) – query params to send, check developer.oanda.com for details.

Query Params example:

```
{
  "id": 2306
}
```

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.transactions as trans
>>> client = oandapyV20.API(access_token=...)
>>> params =
        {
```

```
                "id": 2306
            }
```

```
>>> r = trans.TransactionsSinceID(accountID=..., params=params)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "lastTransactionID": "2311",
  "transactions": [
    {
      "price": "1.25000",
      "timeInForce": "GTC",
      "reason": "REPLACEMENT",
      "clientExtensions": {
        "comment": "myComment",
        "id": "myID"
      },
      "id": "2307",
      "batchID": "2306",
      "triggerCondition": "TRIGGER_DEFAULT",
      "replacesOrderID": "2304",
      "positionFill": "DEFAULT",
      "userID": 1435156,
      "instrument": "EUR_USD",
      "time": "2016-10-25T19:45:38.558056359Z",
      "units": "-500000",
      "type": "LIMIT_ORDER",
      "accountID": "101-004-1435156-001"
    },
    {
      "orderID": "2307",
      "clientOrderID": "myID",
      "reason": "CLIENT_REQUEST",
      "batchID": "2308",
      "time": "2016-10-25T20:53:03.789670387Z",
      "type": "ORDER_CANCEL",
      "userID": 1435156,
      "id": "2308",
      "accountID": "101-004-1435156-001"
    },
    {
      "price": "1.20000",
      "stopLossOnFill": {
        "timeInForce": "GTC",
        "price": "1.22000"
      },
      "timeInForce": "GTC",
      "reason": "CLIENT_ORDER",
      "id": "2309",
      "batchID": "2309",
      "triggerCondition": "TRIGGER_DEFAULT",
      "positionFill": "DEFAULT",
      "userID": 1435156,
      "instrument": "EUR_USD",
      "time": "2016-10-25T21:07:21.065554321Z",
```

```
            "units": "-100",
            "type": "LIMIT_ORDER",
            "accountID": "101-004-1435156-001"
          },
          {
            "userID": 1435156,
            "marginRate": "0.01",
            "batchID": "2310",
            "time": "2016-10-26T13:28:00.507651360Z",
            "type": "CLIENT_CONFIGURE",
            "id": "2310",
            "accountID": "101-004-1435156-001"
          },
          {
            "userID": 1435156,
            "marginRate": "0.01",
            "batchID": "2311",
            "time": "2016-10-26T13:28:13.597103123Z",
            "type": "CLIENT_CONFIGURE",
            "id": "2311",
            "accountID": "101-004-1435156-001"
          }
        ]
      }
```

class oandapyV20.endpoints.transactions.**TransactionsStream**(*accountID*, *params=None*)

Bases: *oandapyV20.endpoints.transactions.Transactions*

TransactionsStream.

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

ENDPOINT = 'v3/accounts/{accountID}/transactions/stream'

EXPECTED_STATUS = 200

METHOD = 'GET'

STREAM = True

__init__(*accountID*, *params=None*)

Instantiate an TransactionsStream request.

Performing this request will result in a generator yielding transactions.

Parameters **accountID** (*string (required)*) – id of the account to perform the request on.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.transactions as trans
>>> client = oandapyV20.API(access_token=...)
>>> r = trans.TransactionsStream(accountID=...)
>>> rv = client.request(r)
>>> maxrecs = 5
>>> try:
>>>     for T in r.response:  # or rv ...
>>>         print json.dumps(R, indent=4), ","
>>>         maxrecs -= 1
>>>         if maxrecs == 0:
>>>             r.terminate("Got them all")
```

```
>>> except StreamTerminated as e:
>>>     print("Finished: {msg}".format(msg=e))
```

Output:

```
        {
          "type": "HEARTBEAT",
          "lastTransactionID": "2311",
          "time": "2016-10-28T11:56:12.002855862Z"
        },
        {
          "type": "HEARTBEAT",
          "lastTransactionID": "2311",
          "time": "2016-10-28T11:56:17.059535527Z"
        },
        {
          "type": "HEARTBEAT",
          "lastTransactionID": "2311",
          "time": "2016-10-28T11:56:22.142256403Z"
        },
        {
          "type": "HEARTBEAT",
          "lastTransactionID": "2311",
          "time": "2016-10-28T11:56:27.238853774Z"
        },
        {
          "type": "HEARTBEAT",
          "lastTransactionID": "2311",
          "time": "2016-10-28T11:56:32.289316796Z"
        }



Finished: Got them all
```

terminate(*message=''*)
terminate the stream.

Calling this method will stop the generator yielding transaction records. A message can be passed optionally.

3.1.2 Submodules

3.1.3 oandapyV20.exceptions module

Exceptions.

exception oandapyV20.exceptions.**StreamTerminated**
Bases: exceptions.Exception

StreamTerminated.

exception oandapyV20.exceptions.**V20Error**(*code*, *msg*)
Bases: exceptions.Exception

Generic error class.

In case of HTTP response codes >= 400 this class can be used to raise an exception representing that error.

3.1.4 oandapyV20.oandapyV20 module

OANDA API wrapper for OANDA's REST-V20 API.

class oandapyV20.oandapyV20.**API**(*access_token*, *environment='practice'*, *headers=None*, *request_params=None*)

Bases: object

API - class to handle APIRequests objects to access API endpoints.

Examples

```
# get a list of trades
from oandapyV20 import API
import oandapyV20.endpoints.trades as trades

api = API(access_token="xxx")
accountID = "101-305-3091856-001"

r = trades.TradesList(accountID)
# show the endpoint as it is constructed for this call
print("REQUEST:{}".format(r))
rv = api.request(r)
print("RESPONSE:\n{}".format(json.dumps(rv, indent=2)))
```

Output:

```
REQUEST:v3/accounts/101-305-3091856-001/trades
RESPONSE:
"trades": [
    {
      "financing": "0.0000",
      "openTime": "2016-07-21T15:47:05.170212014Z",
      "price": "10133.9",
      "unrealizedPL": "8.0000",
      "realizedPL": "0.0000",
      "instrument": "DE30_EUR",
      "state": "OPEN",
      "initialUnits": "-10",
      "currentUnits": "-10",
      "id": "1032"
    },
    {
      "financing": "0.0000",
      "openTime": "2016-07-21T15:47:04.963590941Z",
      "price": "10134.4",
      "unrealizedPL": "13.0000",
      "realizedPL": "0.0000",
      "instrument": "DE30_EUR",
      "state": "OPEN",
      "initialUnits": "-10",
      "currentUnits": "-10",
      "id": "1030"
    }
  ],
  "lastTransactionID": "1040"
}
```

```
# reduce a trade by it's id
from oandapyV20 import API
import oandapyV20.endpoints.trades as trades

api = API(access_token="...")

accountID = "101-305-3091856-001"
tradeID = "1030"
cfg = {"units": 5}
r = trades.TradeClose(accountID, tradeID=tradeID, data=cfg)
# show the endpoint as it is constructed for this call
print("REQUEST:{}".format(r))
rv = api.request(r)
print("RESPONSE\n{}".format(json.dumps(rv, indent=2)))
```

Output:

```
REQUEST:v3/accounts/101-305-3091856-001/trades/1030/close
RESPONSE: {
  "orderFillTransaction": {
    "orderID": "1041",
    "financing": "-0.1519",
    "instrument": "DE30_EUR",
    "userID": 1435156,
    "price": "10131.6",
    "tradeReduced": {
      "units": "5",
      "financing": "-0.1519",
      "realizedPL": "14.0000",
      "tradeID": "1030"
    },
    "batchID": "1041",
    "accountBalance": "44876.2548",
    "reason": "MARKET_ORDER_TRADE_CLOSE",
    "time": "2016-07-21T17:32:51.361464739Z",
    "units": "5",
    "type": "ORDER_FILL",
    "id": "1042",
    "pl": "14.0000",
    "accountID": "101-305-3091856-001"
  },
  "orderCreateTransaction": {
    "timeInForce": "FOK",
    "positionFill": "REDUCE_ONLY",
    "userID": 1435156,
    "batchID": "1041",
    "instrument": "DE30_EUR",
    "reason": "TRADE_CLOSE",
    "tradeClose": {
      "units": "5",
      "tradeID": "1030"
    },
    "time": "2016-07-21T17:32:51.361464739Z",
    "units": "5",
    "type": "MARKET_ORDER",
    "id": "1041",
    "accountID": "101-305-3091856-001"
  },
```

```
    "relatedTransactionIDs": [
      "1041",
      "1042"
    ],
    "lastTransactionID": "1042"
  }
```

request(*endpoint*)
Perform a request for the APIRequest instance 'endpoint'.

Parameters **endpoint** (APIRequest) – The endpoint parameter contains an instance of an APIRequest containing the endpoint, method and optionally other parameters or body data.

Raises V20Error in case of HTTP response code >= 400

request_params
request_params property.

3.1.5 Module contents

CHAPTER 4

oandapyV20.definitions module

4.1 oandapyV20.definitions.accounts

class oandapyV20.definitions.accounts.**AccountFinancingMode**
Bases: object

Definition representation of AccountFinancingMode

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.accounts as defaccounts
>>> print defaccounts.AccountFinancingMode.SECOND_BY_SECOND
SECOND_BY_SECOND
>>> c = defaccounts.AccountFinancingMode()
>>> print c[c.SECOND_BY_SECOND]
Second-by-second financing is paid/charged for open Trades in the Account, both daily and when t
```

DAILY = 'DAILY'

NO_FINANCING = 'NO_FINANCING'

SECOND_BY_SECOND = 'SECOND_BY_SECOND'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.accounts.**PositionAggregationMode**
Bases: object

Definition representation of PositionAggregationMode

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.accounts as defaccounts
>>> print defaccounts.PositionAggregationMode.NET_SUM
NET_SUM
>>> c = defaccounts.PositionAggregationMode()
>>> print c[c.NET_SUM]
The units for each side (long and short) of the Position are netted together and the resulting v
```

ABSOLUTE_SUM = 'ABSOLUTE_SUM'

MAXIMAL_SIDE = 'MAXIMAL_SIDE'

NET_SUM = 'NET_SUM'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.2 oandapyV20.definitions.instruments

class oandapyV20.definitions.instruments.**CandlestickGranularity**
Bases: object

Definition representation of CandlestickGranularity

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.instruments as definstruments
>>> print definstruments.CandlestickGranularity.H4
H4
>>> c = definstruments.CandlestickGranularity()
>>> print c[c.H4]
4 hour candlesticks, day alignment
```

D = 'D'

H1 = 'H1'

H12 = 'H12'

H2 = 'H2'

H3 = 'H3'

H4 = 'H4'

H6 = 'H6'

H8 = 'H8'

M = 'M'

M1 = 'M1'

M15 = 'M15'

M2 = 'M2'

M30 = 'M30'

M4 = 'M4'

M5 = 'M5'

S10 = 'S10'

S15 = 'S15'

S30 = 'S30'

S5 = 'S5'

W = 'W'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.instruments.**WeeklyAlignment**
Bases: object

Definition representation of WeeklyAlignment

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.instruments as definstruments
>>> print definstruments.WeeklyAlignment.Monday
Monday
>>> c = definstruments.WeeklyAlignment()
>>> print c[c.Monday]
Monday
```

Friday = 'Friday'

Monday = 'Monday'

Saturday = 'Saturday'

Sunday = 'Sunday'

Thursday = 'Thursday'

Tuesday = 'Tuesday'

Wednesday = 'Wednesday'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.3 oandapyV20.definitions.orders

class oandapyV20.definitions.orders.**OrderType**
Bases: object

Definition representation of OrderType

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.orders as deforders
>>> print deforders.OrderType.MARKET_IF_TOUCHED
MARKET_IF_TOUCHED
>>> c = deforders.OrderType()
>>> print c[c.MARKET_IF_TOUCHED]
A Market-if-touched Order
```

LIMIT = 'LIMIT'

MARKET = 'MARKET'

MARKET_IF_TOUCHED = 'MARKET_IF_TOUCHED'

STOP = 'STOP'

STOP_LOSS = 'STOP_LOSS'

TAKE_PROFIT = 'TAKE_PROFIT'

TRAILING_STOP_LOSS = 'TRAILING_STOP_LOSS'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.orders.**OrderState**
Bases: object

Definition representation of OrderState

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.orders as deforders
>>> print deforders.OrderState.CANCELLED
CANCELLED
>>> c = deforders.OrderState()
>>> print c[c.CANCELLED]
The Order has been cancelled
```

CANCELLED = 'CANCELLED'

FILLED = 'FILLED'

PENDING = 'PENDING'

TRIGGERED = 'TRIGGERED'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.orders.**TimeInForce**
Bases: object

Definition representation of TimeInForce

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.orders as deforders
>>> print deforders.TimeInForce.IOC
IOC
>>> c = deforders.TimeInForce()
>>> print c[c.IOC]
The Order must be "Immediately partially filled Or Killed"
```

FOK = 'FOK'

GFD = 'GFD'

GTC = 'GTC'

GTD = 'GTD'

IOC = 'IOC'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.orders.**OrderPositionFill**
Bases: object

Definition representation of OrderPositionFill

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.orders as deforders
>>> print deforders.OrderPositionFill.REDUCE_ONLY
REDUCE_ONLY
>>> c = deforders.OrderPositionFill()
>>> print c[c.REDUCE_ONLY]
When the Order is filled, only reduce an existing Position.
```

DEFAULT = 'DEFAULT'

OPEN_ONLY = 'OPEN_ONLY'

REDUCE_FIRST = 'REDUCE_FIRST'

REDUCE_ONLY = 'REDUCE_ONLY'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.4 oandapyV20.definitions.pricing

class oandapyV20.definitions.pricing.**PriceStatus**
Bases: object

Definition representation of PriceStatus

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.pricing as defpricing
>>> print defpricing.PriceStatus.non_tradeable
non-tradeable
>>> c = defpricing.PriceStatus()
>>> print c[c.non_tradeable]
The Instrument's price is not tradeable.
```

Note: attribute name *non-tradeable* is renamed to *non_tradeable*, value stil is *non-tradeable*. This means that a lookup stil applies.

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

invalid = 'invalid'

non_tradeable = 'non-tradeable'

tradeable = 'tradeable'

4.5 oandapyV20.definitions.trades

class oandapyV20.definitions.trades.**TradeState**
Bases: object

Definition representation of TradeState

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.trades as deftrades
>>> print deftrades.TradeState.CLOSE_WHEN_TRADABLE
CLOSE_WHEN_TRADABLE
>>> c = deftrades.TradeState()
>>> print c[c.CLOSE_WHEN_TRADABLE]
The Trade will be closed as soon as the trade's instrument becomes tradeable
```

CLOSED = 'CLOSED'

CLOSE_WHEN_TRADABLE = 'CLOSE_WHEN_TRADABLE'

OPEN = 'OPEN'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.6 oandapyV20.definitions.transactions

class oandapyV20.definitions.transactions.**TransactionType**
Bases: object

Definition representation of TransactionType

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.TransactionType.STOP_LOSS_ORDER
STOP_LOSS_ORDER
>>> c = deftransactions.TransactionType()
>>> print c[c.STOP_LOSS_ORDER]
Stop Loss Order Transaction
```

CLIENT_CONFIGURE = 'CLIENT_CONFIGURE'

CLIENT_CONFIGURE_REJECT = 'CLIENT_CONFIGURE_REJECT'

CLOSE = 'CLOSE'

CREATE = 'CREATE'

DAILY_FINANCING = 'DAILY_FINANCING'

LIMIT_ORDER = 'LIMIT_ORDER'

LIMIT_ORDER_REJECT = 'LIMIT_ORDER_REJECT'

MARGIN_CALL_ENTER = 'MARGIN_CALL_ENTER'

MARGIN_CALL_EXIT = 'MARGIN_CALL_EXIT'

MARGIN_CALL_EXTEND = 'MARGIN_CALL_EXTEND'

MARKET_IF_TOUCHED_ORDER = 'MARKET_IF_TOUCHED_ORDER'

MARKET_IF_TOUCHED_ORDER_REJECT = 'MARKET_IF_TOUCHED_ORDER_REJECT'

MARKET_ORDER = 'MARKET_ORDER'

MARKET_ORDER_REJECT = 'MARKET_ORDER_REJECT'

ORDER_CANCEL = 'ORDER_CANCEL'

ORDER_CLIENT_EXTENSIONS_MODIFY = 'ORDER_CLIENT_EXTENSIONS_MODIFY'

ORDER_CLIENT_EXTENSIONS_MODIFY_REJECT = 'ORDER_CLIENT_EXTENSIONS_MODIFY_REJECT'

ORDER_FILL = 'ORDER_FILL'

REOPEN = 'REOPEN'

RESET_RESETTABLE_PL = 'RESET_RESETTABLE_PL'

STOP_LOSS_ORDER = 'STOP_LOSS_ORDER'

STOP_LOSS_ORDER_REJECT = 'STOP_LOSS_ORDER_REJECT'

STOP_ORDER = 'STOP_ORDER'

STOP_ORDER_REJECT = 'STOP_ORDER_REJECT'

TAKE_PROFIT_ORDER = 'TAKE_PROFIT_ORDER'

TAKE_PROFIT_ORDER_REJECT = 'TAKE_PROFIT_ORDER_REJECT'

TRADE_CLIENT_EXTENSIONS_MODIFY = 'TRADE_CLIENT_EXTENSIONS_MODIFY'

TRADE_CLIENT_EXTENSIONS_MODIFY_REJECT = 'TRADE_CLIENT_EXTENSIONS_MODIFY_REJECT'

TRAILING_STOP_LOSS_ORDER = 'TRAILING_STOP_LOSS_ORDER'

TRAILING_STOP_LOSS_ORDER_REJECT = 'TRAILING_STOP_LOSS_ORDER_REJECT'

TRANSFER_FUNDS = 'TRANSFER_FUNDS'

TRANSFER_FUNDS_REJECT = 'TRANSFER_FUNDS_REJECT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**FundingReason**
Bases: object

Definition representation of FundingReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.FundingReason.ACCOUNT_TRANSFER
ACCOUNT_TRANSFER
>>> c = deftransactions.FundingReason()
>>> print c[c.ACCOUNT_TRANSFER]
Funds are being transfered between two Accounts.
```

ACCOUNT_TRANSFER = 'ACCOUNT_TRANSFER'

ADJUSTMENT = 'ADJUSTMENT'

CLIENT_FUNDING = 'CLIENT_FUNDING'

DIVISION_MIGRATION = 'DIVISION_MIGRATION'

SITE_MIGRATION = 'SITE_MIGRATION'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**MarketOrderReason**
Bases: object

Definition representation of MarketOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.MarketOrderReason.TRADE_CLOSE
TRADE_CLOSE
>>> c = deftransactions.MarketOrderReason()
>>> print c[c.TRADE_CLOSE]
The Market Order was created to close a Trade at the request of a client
```

CLIENT_ORDER = 'CLIENT_ORDER'

DELAYED_TRADE_CLOSE = 'DELAYED_TRADE_CLOSE'

MARGIN_CLOSEOUT = 'MARGIN_CLOSEOUT'

POSITION_CLOSEOUT = 'POSITION_CLOSEOUT'

TRADE_CLOSE = 'TRADE_CLOSE'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**LimitOrderReason**
Bases: object

Definition representation of LimitOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.LimitOrderReason.CLIENT_ORDER
CLIENT_ORDER
>>> c = deftransactions.LimitOrderReason()
>>> print c[c.CLIENT_ORDER]
The Limit Order was initiated at the request of a client
```

CLIENT_ORDER = 'CLIENT_ORDER'

REPLACEMENT = 'REPLACEMENT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**StopOrderReason**
Bases: object

Definition representation of StopOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.StopOrderReason.CLIENT_ORDER
CLIENT_ORDER
>>> c = deftransactions.StopOrderReason()
>>> print c[c.CLIENT_ORDER]
The Stop Order was initiated at the request of a client
```

CLIENT_ORDER = 'CLIENT_ORDER'

REPLACEMENT = 'REPLACEMENT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**MarketIfTouchedOrderReason**
Bases: object

Definition representation of MarketIfTouchedOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.MarketIfTouchedOrderReason.CLIENT_ORDER
CLIENT_ORDER
>>> c = deftransactions.MarketIfTouchedOrderReason()
>>> print c[c.CLIENT_ORDER]
The Market-if-touched Order was initiated at the request of a client
```

CLIENT_ORDER = 'CLIENT_ORDER'

REPLACEMENT = 'REPLACEMENT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**TakeProfitOrderReason**
Bases: object

Definition representation of TakeProfitOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.TakeProfitOrderReason.ON_FILL
ON_FILL
>>> c = deftransactions.TakeProfitOrderReason()
>>> print c[c.ON_FILL]
The Take Profit Order was initiated automatically when an Order was filled that opened a new Tra
```

CLIENT_ORDER = 'CLIENT_ORDER'

ON_FILL = 'ON_FILL'

REPLACEMENT = 'REPLACEMENT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**StopLossOrderReason**
Bases: object

Definition representation of StopLossOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.StopLossOrderReason.ON_FILL
ON_FILL
>>> c = deftransactions.StopLossOrderReason()
>>> print c[c.ON_FILL]
The Stop Loss Order was initiated automatically when an Order was filled that opened a new Trade
```

CLIENT_ORDER = 'CLIENT_ORDER'

ON_FILL = 'ON_FILL'

REPLACEMENT = 'REPLACEMENT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**TrailingStopLossOrderReason**
Bases: object

Definition representation of TrailingStopLossOrderReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.TrailingStopLossOrderReason.ON_FILL
ON_FILL
>>> c = deftransactions.TrailingStopLossOrderReason()
>>> print c[c.ON_FILL]
The Trailing Stop Loss Order was initiated automatically when an Order was filled that opened a
```

CLIENT_ORDER = 'CLIENT_ORDER'

ON_FILL = 'ON_FILL'

REPLACEMENT = 'REPLACEMENT'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**OrderFillReason**
Bases: object

Definition representation of OrderFillReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.OrderFillReason.STOP_ORDER
STOP_ORDER
>>> c = deftransactions.OrderFillReason()
>>> print c[c.STOP_ORDER]
The Order filled was a Stop Order
```

LIMIT_ORDER = 'LIMIT_ORDER'

MARKET_IF_TOUCHED_ORDER = 'MARKET_IF_TOUCHED_ORDER'

MARKET_ORDER = 'MARKET_ORDER'

MARKET_ORDER_DELAYED_TRADE_CLOSE = 'MARKET_ORDER_DELAYED_TRADE_CLOSE'

MARKET_ORDER_MARGIN_CLOSEOUT = 'MARKET_ORDER_MARGIN_CLOSEOUT'

MARKET_ORDER_POSITION_CLOSEOUT = 'MARKET_ORDER_POSITION_CLOSEOUT'

MARKET_ORDER_TRADE_CLOSE = 'MARKET_ORDER_TRADE_CLOSE'

STOP_LOSS_ORDER = 'STOP_LOSS_ORDER'

STOP_ORDER = 'STOP_ORDER'

TAKE_PROFIT_ORDER = 'TAKE_PROFIT_ORDER'

TRAILING_STOP_LOSS_ORDER = 'TRAILING_STOP_LOSS_ORDER'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class oandapyV20.definitions.transactions.**OrderCancelReason**
Bases: object

Definition representation of OrderCancelReason

Definitions used in requests and responses. This class provides the ID and the description of the definitions.

```
>>> import oandapyV20.definitions.transactions as deftransactions
>>> print deftransactions.OrderCancelReason.LINKED_TRADE_CLOSED
LINKED_TRADE_CLOSED
>>> c = deftransactions.OrderCancelReason()
>>> print c[c.LINKED_TRADE_CLOSED]
The Order is linked to an open Trade that was closed.
```

ACCOUNT_LOCKED = 'ACCOUNT_LOCKED'

ACCOUNT_NEW_POSITIONS_LOCKED = 'ACCOUNT_NEW_POSITIONS_LOCKED'

ACCOUNT_ORDER_CREATION_LOCKED = 'ACCOUNT_ORDER_CREATION_LOCKED'

ACCOUNT_ORDER_FILL_LOCKED = 'ACCOUNT_ORDER_FILL_LOCKED'

BOUNDS_VIOLATION = 'BOUNDS_VIOLATION'

CLIENT_REQUEST = 'CLIENT_REQUEST'

CLIENT_REQUEST_REPLACED = 'CLIENT_REQUEST_REPLACED'

CLIENT_TRADE_ID_ALREADY_EXISTS = 'CLIENT_TRADE_ID_ALREADY_EXISTS'

FIFO_VIOLATION = 'FIFO_VIOLATION'

INSUFFICIENT_LIQUIDITY = 'INSUFFICIENT_LIQUIDITY'

INSUFFICIENT_MARGIN = 'INSUFFICIENT_MARGIN'

INTERNAL_SERVER_ERROR = 'INTERNAL_SERVER_ERROR'

LINKED_TRADE_CLOSED = 'LINKED_TRADE_CLOSED'

LOSING_TAKE_PROFIT = 'LOSING_TAKE_PROFIT'

MARKET_HALTED = 'MARKET_HALTED'

MIGRATION = 'MIGRATION'

OPEN_TRADES_ALLOWED_EXCEEDED = 'OPEN_TRADES_ALLOWED_EXCEEDED'

PENDING_ORDERS_ALLOWED_EXCEEDED = 'PENDING_ORDERS_ALLOWED_EXCEEDED'

POSITION_CLOSEOUT_FAILED = 'POSITION_CLOSEOUT_FAILED'

POSITION_SIZE_EXCEEDED = 'POSITION_SIZE_EXCEEDED'

STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS = 'STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_A

STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST'

STOP_LOSS_ON_FILL_LOSS = 'STOP_LOSS_ON_FILL_LOSS'

TAKE_PROFIT_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS = 'TAKE_PROFIT_ON_FILL_CLIENT_ORDER

TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST

TAKE_PROFIT_ON_FILL_LOSS = 'TAKE_PROFIT_ON_FILL_LOSS'

TIME_IN_FORCE_EXPIRED = 'TIME_IN_FORCE_EXPIRED'

TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS = 'TRAILING_STOP_LOSS_ON_FILL

TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TRAILING_STOP_LOSS_ON_FILL_GTD_TIMI

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

CHAPTER 5

oandapyV20.contrib module

5.1 oandapyV20.contrib.requests

5.1.1 Support classes

The requests package contains several classes that can be used optional when creating Order Requests. When creating an order to create a position, it is possible to create dependant orders that will be triggered when the position gets filled. This goes typically for *Take Profit* and *Stop Loss*.

class oandapyV20.contrib.requests.**TakeProfitDetails**(*price*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Bases: oandapyV20.contrib.requests.onfill.OnFill

Representation of the specification for a TakeProfitOrder.

It is typically used to specify 'take profit details' for the 'takeProfitOnFill' parameter of an OrderRequest. This way one can create the Take Profit Order as a dependency when an order gets filled.

The other way to create a TakeProfitOrder is to create it afterwards on an existing trade. In that case you use TakeProfitOrderRequest on the trade.

__init__(*price*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Instantiate TakeProfitDetails.

Parameters

- **price** (*float or string (required)*) – the price to trigger take profit order
- **timeInForce** (*TimeInForce (required), default TimeInForce.GTC*) – the time in force
- **gtdTime** (*DateTime (optional)*) – gtdTime is required in case timeInForce == TimeInForce.GTD

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import (
>>>     MarketOrderRequest, TakeProfitDetails)
>>>
```

```
>>> accountID = "..."
>>> client = API(access_token=...)
>>> # at time of writing EUR_USD = 1.0740
>>> # let us take profit at 1.10, GoodTillCancel (default)
>>> takeProfitOnFillOrder = TakeProfitDetails(price=1.10)
>>> print(takeProfitOnFillOrder)
{
    "timeInForce": "GTC",
    "price": "1.10000"
}
>>> ordr = MarketOrderRequest(
>>>     instrument="EUR_USD",
>>>     units=10000,
>>>     takeProfitOnFill=takeProfitOnFillOrder.data
>>> )
>>> # or as shortcut ...
>>> #   takeProfitOnFill=TakeProfitDetails(price=1.10).data
>>> print(json.dumps(ordr.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

class oandapyV20.contrib.requests.**StopLossDetails**(*price*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Bases: oandapyV20.contrib.requests.onfill.OnFill

Representation of the specification for a StopLossOrder.

It is typically used to specify 'stop loss details' for the 'stopLossOnFill' parameter of an OrderRequest. This way one can create the Stop Loss Order as a dependency when an order gets filled.

The other way to create a StopLossOrder is to create it afterwards on an existing trade. In that case you use StopLossOrderRequest on the trade.

__init__(*price*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Instantiate TakeProfitDetails.

Parameters

- **price** (*float or string (required)*) – the price to trigger take profit order
- **timeInForce** (*TimeInForce (required), default TimeInForce.GTC*) – the time in force
- **gtdTime** (*DateTime (optional)*) – gtdTime is required in case timeInForce == TimeInForce.GTD
- **clientExtensions** (*ClientExtensions (optional)*) –

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import (
>>>     MarketOrderRequest, StopLossDetails)
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> # at time of writing EUR_USD = 1.0740
```

```
>>> # let us take profit at 1.10, GoodTillCancel (default)
>>> stopLossOnFill = StopLossDetails(price=1.06)
>>> print(stopLossOnFill)
{
    "timeInForce": "GTC",
    "price": "1.10000"
}
>>> ordr = MarketOrderRequest(
>>>     instrument="EUR_USD",
>>>     units=10000,
>>>     stopLossOnFill=stopLossOnFill.data
>>> )
>>> # or as shortcut ...
>>> #   stopLossOnFill=StopLossDetails(price=1.06).data
>>> print(json.dumps(ordr.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

class oandapyV20.contrib.requests.**TrailingStopLossDetails**(*distance*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Bases: oandapyV20.contrib.requests.onfill.OnFill

Representation of the specification for a TrailinStopLossOrder.

It is typically used to specify 'trainling stop loss details' for the 'trailingStopLossOnFill' parameter of an OrderRequest. This way one can create the Trailing Stop Loss Order as a dependency when an order gets filled.

The other way to create a TrailinStopLossOrder is to create it afterwards on an existing trade. In that case you use TrailingStopLossOrderRequest on the trade.

__init__(*distance*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Instantiate TakeProfitDetails.

Parameters

- **distance** (*float or string (required)*) – the price to trigger take profit order
- **timeInForce** (*TimeInForce (required), default TimeInForce.GTC*) – the time in force
- **gtdTime** (*DateTime (optional)*) – gtdTime is required in case timeInForce == TimeInForce.GTD
- **clientExtensions** (*ClientExtensions (optional)*) –

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import (
>>>     MarketOrderRequest, TrailingStopLossDetails)
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
```

```
>>> # at time of writing EUR_USD = 1.0740
>>> # let us take profit at 1.10, GoodTillCancel (default)
>>> trailingStopLossOnFill = TrailingStopLossDetails(price=1.06)
>>> print(trailingStopLossOnFill)
{
    "timeInForce": "GTC",
    "price": "1.10000"
}
>>> ordr = MarketOrderRequest(
>>>     instrument="EUR_USD",
>>>     units=10000,
>>>     trailingStopLossOnFill=trailingStopLossOnFill.data
>>> )
>>> # or as shortcut ...
>>> #   ...OnFill=trailingStopLossDetails(price=1.06).data
>>> print(json.dumps(ordr.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

Client Extensions

Client extensions can be used optionally on Order Requests. It allows a client to set a custom ID, Tag and/or Comment.

class oandapyV20.contrib.requests.**ClientExtensions**(*clientID=None*, *clientTag=None*, *clientComment=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

Representation of the ClientExtensions.

__init__(*clientID=None*, *clientTag=None*, *clientComment=None*)

Instantiate ClientExtensions.

Parameters

- **clientID** (*clientID (required)*) – the clientID
- **clientTag** (*clientTag (required)*) – the clientTag
- **clientComment** (*clientComment (required)*) – the clientComment

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import (
...     MarketOrderRequest, TakeProfitDetails, ClientExtensions)
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> # at time of writing EUR_USD = 1.0740
>>> # let us take profit at 1.10, GoodTillCancel (default)
>>> # add clientExtensions to it also
>>> takeProfitOnFillOrder = TakeProfitDetails(
...     price=1.10,
...     clientExtensions=ClientExtensions(clientTag="mytag").data)
```

```
>>> print(takeProfitOnFillOrder.data)
{
    'timeInForce': 'GTC',
    'price": '1.10000',
    'clientExtensions': {'tag': 'mytag'}
}
>>> ordr = MarketOrderRequest(
...     instrument="EUR_USD",
...     units=10000,
...     takeProfitOnFill=takeProfitOnFillOrder.data
... )
>>> # or as shortcut ...
>>> #   takeProfitOnFill=TakeProfitDetails(price=1.10).data
>>> print(json.dumps(ordr.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

5.1.2 Order classes

class oandapyV20.contrib.requests.**MarketOrderRequest**(*instrument*, *units*, *priceBound=None*, *positionFill='DEFAULT'*, *clientExtensions=None*, *takeProfitOnFill=None*, *timeInForce='FOK'*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a MarketOrderRequest.

MarketOrderRequest is used to build the body for a MarketOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*instrument*, *units*, *priceBound=None*, *positionFill='DEFAULT'*, *clientExtensions=None*, *takeProfitOnFill=None*, *timeInForce='FOK'*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)

Instantiate a MarketOrderRequest.

Parameters

- **instrument** (*string (required)*) – the instrument to create the order for
- **units** (*integer (required)*) – the number of units. If positive the order results in a LONG order. If negative the order results in a SHORT order

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import MarketOrderRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> mo = MarketOrderRequest(instrument="EUR_USD", units=10000)
```

```
>>> print(json.dumps(mo.data, indent=4))
{
    "order": {
        "type": "MARKET",
        "positionFill": "DEFAULT",
        "instrument": "EUR_USD",
        "timeInForce": "FOK",
        "units": "10000"
    }
}
>>> # now we have the order specification, create the order request
>>> r = orders.OrderCreate(accountID, data=mo.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(rv)
>>> print(json.dumps(rv, indent=4))
{
    "orderFillTransaction": {
        "reason": "MARKET_ORDER",
        "pl": "0.0000",
        "accountBalance": "97864.8813",
        "units": "10000",
        "instrument": "EUR_USD",
        "accountID": "101-004-1435156-001",
        "time": "2016-11-11T19:59:43.253587917Z",
        "type": "ORDER_FILL",
        "id": "2504",
        "financing": "0.0000",
        "tradeOpened": {
            "tradeID": "2504",
            "units": "10000"
        },
        "orderID": "2503",
        "userID": 1435156,
        "batchID": "2503",
        "price": "1.08463"
    },
    "lastTransactionID": "2504",
    "relatedTransactionIDs": [
        "2503",
        "2504"
    ],
    "orderCreateTransaction": {
        "type": "MARKET_ORDER",
        "reason": "CLIENT_ORDER",
        "id": "2503",
        "timeInForce": "FOK",
        "units": "10000",
        "time": "2016-11-11T19:59:43.253587917Z",
        "positionFill": "DEFAULT",
        "accountID": "101-004-1435156-001",
        "instrument": "EUR_USD",
        "batchID": "2503",
        "userID": 1435156
    }
}
>>>
```

data
data property.

return the JSON body.

class oandapyV20.contrib.requests.**LimitOrderRequest**(*instrument*, *units*, *price*, *positionFill='DEFAULT'*, *clientExtensions=None*, *takeProfitOnFill=None*, *timeInForce='GTC'*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)
Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a LimitOrderRequest.

LimitOrderRequest is used to build the body for a LimitOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*instrument*, *units*, *price*, *positionFill='DEFAULT'*, *clientExtensions=None*, *takeProfitOnFill=None*, *timeInForce='GTC'*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)
Instantiate a LimitOrderRequest.

Parameters

- **instrument** (*string (required)*) – the instrument to create the order for
- **units** (*integer (required)*) – the number of units. If positive the order results in a LONG order. If negative the order results in a SHORT order
- **price** (*float (required)*) – the price indicating the limit.

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import LimitOrderRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> mo = LimitOrderRequest(instrument="EUR_USD",
>>>                        units=10000, price=1.08)
>>> print(json.dumps(mo.data, indent=4))
>>> ...
```

data
data property.

return the JSON order body

class oandapyV20.contrib.requests.**MITOrderRequest**(*instrument*, *units*, *price*, *priceBound=None*, *positionFill='DEFAULT'*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*, *takeProfitOnFill=None*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a MarketIfTouched OrderRequest.

MITOrderRequest is used to build the body for a MITOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*instrument*, *units*, *price*, *priceBound=None*, *positionFill='DEFAULT'*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*, *takeProfitOnFill=None*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)

Instantiate an MITOrderRequest.

Parameters

- **instrument** (*string (required)*) – the instrument to create the order for
- **units** (*integer (required)*) – the number of units. If positive the order results in a LONG order. If negative the order results in a SHORT order
- **price** (*float (required)*) – the price indicating the limit.

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import MITOrderRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> mo = MITOrderRequest(instrument="EUR_USD",
>>>                      units=10000, price=1.08)
>>> print(json.dumps(mo.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=mo.data)
>>> rv = client.request(r)
>>> ...
```

data

data property.

return the JSON order body

class oandapyV20.contrib.requests.**TakeProfitOrderRequest**(*tradeID*, *price*, *clientTradeID=None*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a TakeProfit OrderRequest.

TakeProfitOrderRequest is used to build the body for a TakeProfitOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*tradeID*, *price*, *clientTradeID=None*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Instantiate a TakeProfitOrderRequest.

Parameters

- **tradeID** (*string (required)*) – the tradeID of an existing trade

- **price** (*float (required)*) – the price indicating the target price to close the order.

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import TakeProfitOrderRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> ordr = TakeProfitOrderRequest(tradeID="1234",
>>>                               price=1.22)
>>> print(json.dumps(ordr.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

data
data property.

return the JSON order body

class oandapyV20.contrib.requests.**StopLossOrderRequest**(*tradeID*, *price*, *clientTradeID=None*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a StopLossOrderRequest.

StopLossOrderRequest is used to build the body for a MarketOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*tradeID*, *price*, *clientTradeID=None*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)
Instantiate a StopLossOrderRequest.

Parameters

- **tradeID** (*string (required)*) – the tradeID of an existing trade
- **price** (*float (required)*) – the treshold price indicating the price to close the order

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import StopLossOrderRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> ordr = StopLossOrderRequest(tradeID="1234", price=1.07)
>>> print(json.dumps(ordr.data, indent=4))
{
    "order": {
```

```
                "type": "STOP_LOSS",
                "tradeID": "1234",
                "price": "1.07000",
                "timeInForce": "GTC",
            }
        }
        >>> # now we have the order specification, create the order request
        >>> r = orders.OrderCreate(accountID, data=ordr.data)
        >>> # perform the request
        >>> rv = client.request(r)
        >>> print(rv)
        >>> print(json.dumps(rv, indent=4))
        >>> ...
```

data

data property.

return the JSON body.

class oandapyV20.contrib.requests.**TrailingStopLossOrderRequest**(*tradeID*, *distance*, *clientTradeID=None*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a TrailingStopLossOrderRequest.

TrailingStopLossOrderRequest is used to build the body for a TrailingStopLossOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*tradeID*, *distance*, *clientTradeID=None*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*)

Instantiate a TrailingStopLossOrderRequest.

Parameters

- **tradeID** (*string (required)*) – the tradeID of an existing trade
- **distance** (*float (required)*) – the price distance

Example

```
        >>> import json
        >>> from oandapyV20 import API
        >>> import oandapyV20.endpoints.orders as orders
        >>> from oandapyV20.contrib.requests import TrailingStopLossOrderRequest
        >>>
        >>> accountID = "..."
        >>> client = API(access_token=...)
        >>> ordr = TrailingStopLossOrderRequest(tradeID="1234", distance=20)
        >>> print(json.dumps(ordr.data, indent=4))
        {
            "order": {
                "type": "TRAILING_STOP_LOSS",
                "tradeID": "1234",
```

```
                "timeInForce": "GTC",
                "distance": "20.00000"
            }
        }
        >>> # now we have the order specification, create the order request
        >>> r = orders.OrderCreate(accountID, data=ordr.data)
        >>> # perform the request
        >>> rv = client.request(r)
        >>> print(rv)
        >>> print(json.dumps(rv, indent=4))
        >>> ...
```

data
data property.

return the JSON body.

class oandapyV20.contrib.requests.**StopOrderRequest**(*instrument*, *units*, *price*, *priceBound=None*, *positionFill='DEFAULT'*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*, *takeProfitOnFill=None*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a StopOrderRequest.

StopOrderRequest is used to build the body for an StopOrder. The body can be used to pass to the OrderCreate endpoint.

__init__(*instrument*, *units*, *price*, *priceBound=None*, *positionFill='DEFAULT'*, *timeInForce='GTC'*, *gtdTime=None*, *clientExtensions=None*, *takeProfitOnFill=None*, *stopLossOnFill=None*, *trailingStopLossOnFill=None*, *tradeClientExtensions=None*)

Instantiate a StopOrderRequest.

Parameters

- **instrument** (*string (required)*) – the instrument to create the order for
- **units** (*integer (required)*) – the number of units. If positive the order results in a LONG order. If negative the order results in a SHORT order
- **price** (*float (required)*) – the treshold price indicating the price to activate the order

Example

```
        >>> import json
        >>> from oandapyV20 import API
        >>> import oandapyV20.endpoints.orders as orders
        >>> from oandapyV20.contrib.requests import StopOrderRequest
        >>>
        >>> accountID = "..."
        >>> client = API(access_token=...)
        >>> ordr = StopOrderRequest(instrument="EUR_USD",
        >>>                         units=10000, price=1.07)
        >>> print(json.dumps(ordr.data, indent=4))
```

```
        {
            "order": {
                "type": "STOP",
                "price": "1.07000",
                "positionFill": "DEFAULT",
                "instrument": "EUR_USD",
                "timeInForce": "GTC",
                "units": "10000"
            }
        }
        >>> # now we have the order specification, create the order request
        >>> r = orders.OrderCreate(accountID, data=ordr.data)
        >>> # perform the request
        >>> rv = client.request(r)
        >>> print(rv)
        >>> print(json.dumps(rv, indent=4))
        >>> ...
```

data
data property.

return the JSON body.

class oandapyV20.contrib.requests.**PositionCloseRequest**(*longUnits=None*, *longClientExtensions=None*, *shortUnits=None*, *shortClientExtensions=None*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a PositionCloseRequest.

PositionCloseRequest is used to build the body to close a position. The body can be used to pass to the PositionClose endpoint.

__init__(*longUnits=None*, *longClientExtensions=None*, *shortUnits=None*, *shortClientExtensions=None*)
Instantiate a PositionCloseRequest.

Parameters

- **longUnits** (*integer (optional)*) – the number of long units to close
- **longClientExtensions** (*dict (optional)*) – dict representing longClientExtensions
- **shortUnits** (*integer (optional)*) – the number of short units to close
- **shortClientExtensions** (*dict (optional)*) – dict representing shortClientExtensions
- **of the parameters or both must be supplied.** (*One*) –

Example

```
        >>> import json
        >>> from oandapyV20 import API
        >>> import oandapyV20.endpoints.positions as positions
        >>> from oandapyV20.contrib.requests import PositionCloseRequest
        >>>
        >>> accountID = "..."
```

```
>>> client = API(access_token=...)
>>> ordr = PositionCloseRequest(longUnits=10000)
>>> print(json.dumps(ordr.data, indent=4))
{
   "longUnits": "10000"
}
>>> # now we have the order specification, create the order request
>>> r = position.PositionClose(accountID,
>>>                            instrument="EUR_USD", data=ordr.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(rv)
>>> ...
```

class oandapyV20.contrib.requests.**TradeCloseRequest**(*units='ALL'*)

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a TradeCloseRequest.

TradeCloseRequest is used to build the body to close a trade. The body can be used to pass to the TradeClose endpoint.

__init__(*units='ALL'*)

Instantiate a TradeCloseRequest.

Parameters **units** (*integer (optional)*) – the number of units to close. Default it is set to "ALL".

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.trades as trades
>>> from oandapyV20.contrib.requests import TradeCloseRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> ordr = TradeCloseRequest(units=10000)
>>> print(json.dumps(ordr.data, indent=4))
{
   "units": "10000"
}
>>> # now we have the order specification, create the order request
>>> r = trades.TradeClose(accountID, tradeID=1234,
>>>                       data=ordr.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(rv)
>>> ...
```

CHAPTER 6

oandapyV20.types

class oandapyV20.types.**AccountID**(*accountID*)
representation of an AccountID, string value of an Account Identifier.

Parameters **accountID** (*string (required)*) – the accountID of a v20 account

Example

```
>>> print AccountID("001-011-5838423-001").value
```

A ValueError exception is raised in case of an incorrect value.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**OrderID**(*orderID*)
representation of an orderID, string value of an integer.

Parameters **orderID** (*integer or string (required)*) – the orderID as a positive integer or as a string

Example

```
>>> print OrderID(1234).value
```

A ValueError exception is raised in case of a negative integer value

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**TradeID**(*tradeID*)
representation of a tradeID, string value of an integer.

Parameters **tradeID** (*integer or string (required)*) – the tradeID as a positive integer or as a string

Example

```
>>> print TradeID(1234).value
```

A ValueError exception is raised in case of a negative integer value

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**AccountUnits**(*units*)
representation AccountUnits, string value of a float.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**PriceValue**(*priceValue*)
representation PriceValue, string value of a float.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**Units**(*units*)
representation Units, string value of an integer.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**ClientID**(*clientID*)
representation of ClientID, a string value of max 128 chars.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**ClientTag**(*clientTag*)
representation of ClientTag, a string value of max 128 chars.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**ClientComment**(*clientComment*)
representation of ClientComment, a string value of max 128 chars.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**OrderIdentifier**(*orderID*, *clientID*)
representation of the OrderIdentifier object.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

class oandapyV20.types.**OrderSpecifier**(*specifier*)
representation of the OrderSpecifier.

__delattr__
x.__delattr__('name') <==> del x.name

__format__()
default object formatter

__getattribute__
x.__getattribute__('name') <==> x.name

__hash__

__reduce__()
helper for pickle

__reduce_ex__()
helper for pickle

__repr__

__setattr__
x.__setattr__('name', value) <==> x.name = value

__sizeof__() → int
size of object in memory, in bytes

__str__

value
value property.

CHAPTER 7

Examples

7.1 Example for trades-endpoints

Take the script below and name it 'trades.py'. From the shell:

```
hootnot@dev:~/test$ python trades.py list
hootnot@dev:~/test$ python trades.py open
hootnot@dev:~/test$ python trades.py details <id1> [<id2> ...]
hootnot@dev:~/test$ python trades.py close <id1> <numunits> [<id2> <numunits>...]
hootnot@dev:~/test$ python trades.py clext <id1> [<id2> ...]
hootnot@dev:~/test$ python trades.py crc_do <id1> <takeprofit> <stoploss> [<id2> ...]
```

```
# use of the Trades{..} classes
import json
import requests
from oandapyV20 import API


import oandapyV20.endpoints.trades as trades
import sys

access_token = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx-yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy"
accountID = "zzz-zzzz-zzzzz"

api = API(access_token=access_token)

if chc == 'list':
   r = trades.TradesList(accountID)
   rv = api.request(r)
   print("RESP:\n{} ".format(json.dumps(rv, indent=2)))

if chc == 'open':
   r = trades.OpenTrades(accountID)
   rv = api.request(r)
   print("RESP:\n{} ".format(json.dumps(rv, indent=2)))
   tradeIDs = [o["id"] for o in rv["trades"]]
   print("TRADE IDS: {}".format(tradeIDs))

if chc == 'details':
   for O in sys.argv[2:]:
       r = trades.TradeDetails(accountID, tradeID=O)
       rv = api.request(r)
       print("RESP:\n{} ".format(json.dumps(rv, indent=2)))
```

```
if chc == 'close':
   X = iter(sys.argv[2:])
   for O in X:
       cfg = { "units": X.next() }
       r = trades.TradeClose(accountID, tradeID=O, data=cfg)
       rv = api.request(r)
       print("RESP:\n{} ".format(json.dumps(rv, indent=2)))

if chc == 'cltext':
   for O in sys.argv[2:]:  # tradeIDs
       cfg = { "clientExtensions": {
                "id": "myID{}".format(O),
                "comment": "myComment",
            }
         }
       r = trades.TradeClientExtensions(accountID, tradeID=O, data=cfg)
       rv = api.request(r)
       print("RESP:\n{} ".format(json.dumps(rv, indent=2)))

if chc == 'crc_do':
   X = iter(sys.argv[2:])
   for O in X:
       cfg = {
                "takeProfit": {
                  "timeInForce": "GTC",
                  "price": X.next(),
                },
                "stopLoss": {
                  "timeInForce": "GTC",
                  "price": X.next()
                }
         }
       r = trades.TradeCRCDO(accountID, tradeID=O, data=cfg)
       rv = api.request(r)
       print("RESP:\n{} ".format(json.dumps(rv, indent=2)))
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

O

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W