
OANDA REST-V20 API Documentation

Release 0.6.0

Feite Brekeveld

Mar 10, 2018

oandapyV20 REST-V20 API wrapper

1	Introduction	3
1.1	Install	3
1.2	Download from Github	3
2	Interface OANDA's REST-V20	5
2.1	The client	5
2.2	Exceptions	8
2.3	Logging	8
3	oandapyV20.endpoints	11
3.1	oandapyV20.endpoints.accounts	11
3.2	oandapyV20.endpoints.forexlabs	20
3.3	oandapyV20.endpoints.instruments	34
3.4	oandapyV20.endpoints.orders	42
3.5	oandapyV20.endpoints.positions	50
3.6	oandapyV20.endpoints.pricing	56
3.7	oandapyV20.endpoints.trades	62
3.8	oandapyV20.endpoints.transactions	70
4	oandapyV20.definitions	79
4.1	oandapyV20.definitions.accounts	79
4.2	oandapyV20.definitions.instruments	80
4.3	oandapyV20.definitions.orders	82
4.4	oandapyV20.definitions.pricing	86
4.5	oandapyV20.definitions.trades	87
4.6	oandapyV20.definitions.transactions	89
5	oandapyV20.types	101
5.1	AccountID	101
5.2	AccountUnits	102
5.3	ClientComment	102
5.4	ClientID	102
5.5	ClientTag	102
5.6	DateTime	102
5.7	OrderID	103
5.8	OrderIdentifier	103
5.9	OrderSpecifier	104

5.10	PriceValue	104
5.11	TradeID	104
5.12	Units	104
6	oandapyV20.contrib	107
6.1	Factories	107
6.2	Generic	108
6.3	Order Classes	109
6.4	support classes	119
7	Examples	125
7.1	Example for trades-endpoints	125
8	Indices and tables	127
	Python Module Index	129

Contents:

The `oandapyV20` package offers an API to the OANDA V20 REST service. To use the REST-API-service you will need a *token* and an *account*. This applies for both *live* and *practice* accounts. For details check oanda.com.

1.1 Install

Install the pypi package with pip:

```
$ pip install oandapyV20
```

Or alternatively install the latest development version from github:

```
$ 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 Download from Github

If you want to run the tests, download the source from github:

```
$ git clone https://github.com/hootnot/oanda-api-v20.git
$ cd oanda-api-v20
$ python setup.py test
$ python setup.py install
```

Interface OANDA's REST-V20

2.1 The client

The `oandapyV20` package contains a client class, `oandapyV20.API`, to communicate with the REST-V20 interface. It processes requests that can be created from the endpoint classes. For its communication it relies on: `requests (requests)`.

The client keeps no state of a requests. The response of a request is assigned to the request instance. The response is also returned as a return value by the client.

```
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"
}

```

__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.

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.

__init__ (*code, msg*)

Instantiate a V20Error.

Parameters

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

2.3 Logging

The oandapyV20 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.oanda.com/v3/accounts/101-004-1435156-001/orders
2016-10-22 17:50:37,998 [INFO] requests.packages.urllib3.connectionpool : Starting_
↳new HTTPS connection (1): api-fxpractice.oanda.com
2016-10-22 17:50:38,866 [INFO] oandapyV20.oandapyV20 : performing request https://api-
↳fxpractice.oanda.com/v3/accounts/101-004-1435156-001/orders
2016-10-22 17:50:39,066 [ERROR] oandapyV20.oandapyV20 : request https://api-
↳fxpractice.oanda.com/v3/accounts/101-004-1435156-001/orders failed [400,{
↳"errorMessage":"Invalid value specified for 'order.instrument'"}]
```


3.1 oandapyV20.endpoints.accounts

3.1.1 AccountChanges

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"
}

```

3.1.2 AccountConfiguration

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"
  }
}

```

3.1.3 AccountDetails

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": {
    "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"
}

```

3.1.4 AccountInstruments

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": "10000000",
  "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"
}

```

3.1.5 AccountList

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

```

```

{
  "accounts": [
    {
      "id": "101-004-1435156-002",
      "tags": []
    },
    {
      "id": "101-004-1435156-001",
      "tags": []
    }
  ]
}

```

3.1.6 AccountSummary

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"
}
```

3.2 oandapyV20.endpoints.forexlabs

3.2.1 Autochartist

class oandapyV20.endpoints.forexlabs.**Autochartist** (*params=None*)

Bases: oandapyV20.endpoints.forexlabs.ForexLabs

Autochartist.

Get the 'autochartist data'.

ENDPOINT = 'labs/v1/signal/autochartist'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)

Instantiate an Autochartist request.

Parameters **params** (*dict (optional)*) – query params to send, check developer.oanda.com for details.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.forexlabs as labs
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> params =
    {
        "instrument": "EUR_JPY"
    }
```

```
>>> r = labs.Autochartist(params=params)
>>> client.request(r)
>>> print(r.response)
```

Output:

```
{
  "signals": [
    {
      "data": {
        "points": {
          "support": {
            "y1": 0.72456,
            "y0": 0.725455,
            "x0": 1520420400,
            "x1": 1520503200
          },
          "resistance": {
            "y1": 0.729755,
            "y0": 0.731095,
            "x0": 1520323200,
            "x1": 1520463600
          }
        },
        "patternendtime": 1520589600,
        "prediction": {
          "pricelow": 0.7316,
          "timefrom": 1520589600,
          "pricehigh": 0.7349,
          "timeto": 1520773200
        }
      },
      "meta": {
        "direction": 1,
        "completed": 1,
        "probability": 72.36,
        "scores": {
          "clarity": 7,
          "breakout": 10,
          "quality": 8,
          "initialtrend": 10,
          "uniformity": 6
        }
      }
    }
  ]
}
```

```

    },
    "pattern": "Channel Down",
    "historicalstats": {
      "hourofday": {
        "total": 1909,
        "percent": 71.08,
        "correct": 1357
      },
      "pattern": {
        "total": 3361,
        "percent": 73.61,
        "correct": 2474
      },
      "symbol": {
        "total": 429,
        "percent": 65.5,
        "correct": 281
      }
    },
    "interval": 60,
    "trendtype": "Continuation",
    "length": 73
  },
  "type": "chartpattern",
  "id": 458552738,
  "instrument": "NZD_USD"
}
1,
"provider": "autochartist"
}

```

3.2.2 Calendar

class oandapyV20.endpoints.forexlabs.**Calendar** (*params*)

Bases: oandapyV20.endpoints.forexlabs.ForexLabs

Calendar.

Get calendar information.

ENDPOINT = 'labs/v1/calendar'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__ (*params*)

Instantiate a Calendar request.

Parameters **params** (*dict (required)*) – query params to send, check developer.oanda.com for details.

```

>>> import oandapyV20
>>> import oandapyV20.endpoints.forexlabs as labs
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> params =
    {
        "instrument": "EUR_USD",

```

```
    "period": 86400
  }
```

```
>>> r = labs.Calendar(params=params)
>>> client.request(r)
>>> print(r.response)
```

Output:

```
[
  {
    "impact": 3,
    "currency": "USD",
    "actual": "60.8",
    "market": "58.7",
    "title": "ISM Manufacturing",
    "timestamp": 1519916400,
    "region": "americas",
    "previous": "59.1",
    "unit": "index",
    "forecast": "59.5"
  }
]
```

3.2.3 CommitmentsOfTraders

class oandapyV20.endpoints.forexlabs.**CommitmentsOfTraders** (*params*)

Bases: oandapyV20.endpoints.forexlabs.ForexLabs

CommitmentsOfTraders.

Get the ‘commitments of traders’ information for an instrument.

ENDPOINT = 'labs/v1/commitments_of_traders'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__ (*params*)

Instantiate a CommitmentsOfTraders request.

Parameters **params** (*dict (required)*) – query params to send, check developer.oanda.com for details.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.forexlabs as labs
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> params =
    {
        "instrument": "EUR_USD"
    }
```

```
>>> r = labs.CommitmentOfTraders(params=params)
>>> client.request(r)
>>> print(r.response)
```

Output:

```
{
  "EUR_USD": [
    {
      "oi": "603460",
      "price": "1.2315925",
      "ncs": "109280",
      "ncl": "258022",
      "date": 1517288400,
      "unit": "Contracts Of EUR 125,000"
    },
    {
      "oi": "596937",
      "price": "1.2364",
      "ncs": "110546",
      "ncl": "251369",
      "date": 1517893200,
      "unit": "Contracts Of EUR 125,000"
    },
    {
      "oi": "564233",
      "price": "1.2330275",
      "ncs": "103496",
      "ncl": "230785",
      "date": 1518498000,
      "unit": "Contracts Of EUR 125,000"
    },
    {
      "oi": "567534",
      "price": "1.2346025",
      "ncs": "103147",
      "ncl": "229273",
      "date": 1519102800,
      "unit": "Contracts Of EUR 125,000"
    },
    {
      "oi": "567463",
      "price": "1.23557",
      "ncs": "100310",
      "ncl": "238287",
      "date": 1519707600,
      "unit": "Contracts Of EUR 125,000"
    }
  ]
}
```

3.2.4 HistoricalPositionRatios

class oandapyV20.endpoints.forexlabs.**HistoricalPositionRatios** (*params*)

Bases: oandapyV20.endpoints.forexlabs.ForexLabs

HistoricalPositionRatios.

Get the historical positionratios for an instrument.

ENDPOINT = 'labs/v1/historical_position_ratios'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(params)

Instantiate a HistoricalPositionRatios request.

Parameters **params** (*dict (required)*) – query params to send, check developer.oanda.com for details.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.forexlab as labs
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> params =
    {
        "instrument": "EUR_USD",
        "period": 86400
    }
```

```
>>> r = labs.HistoricalPositionRatios(params=params)
>>> client.request(r)
>>> print(r.response)
```

Output:

```
{
  "data": {
    "EUR_USD": {
      "data": [
        [
          1519924801,
          44.22,
          1.2209
        ],
        [
          1519926001,
          44.33,
          1.221
        ],
        [
          1519927200,
          44.16,
          1.2212
        ],
        [
          1519928400,
          44.2,
          1.2209
        ],
        [
          1519929601,
          43.88,
          1.2201
        ],
        [
          1519930800,
          44.15,
          1.2197
        ]
      ]
    }
  }
```

```
],  
[  
  1519932000,  
  44.51,  
  1.2204  
],  
[  
  1519933200,  
  44.55,  
  1.2233  
],  
[  
  1519934401,  
  44.55,  
  1.2254  
],  
[  
  1519935601,  
  44.08,  
  1.226  
],  
[  
  1519936801,  
  43.67,  
  1.2264  
],  
[  
  1519938001,  
  43.55,  
  1.2263  
],  
[  
  1519939201,  
  43.25,  
  1.2261  
],  
[  
  1519940401,  
  43.28,  
  1.2263  
],  
[  
  1519941601,  
  43.39,  
  1.2267  
],  
[  
  1519942801,  
  43.69,  
  1.227  
],  
[  
  1519944001,  
  43.57,  
  1.2269  
],  
[  
  1519945201,
```

```
    43.68,  
    1.2272  
  ],  
  [  
    1519946400,  
    43.51,  
    1.2268  
  ],  
  [  
    1519947601,  
    43.53,  
    1.2267  
  ],  
  [  
    1519948801,  
    43.71,  
    1.2271  
  ],  
  [  
    1519950001,  
    43.66,  
    1.2265  
  ],  
  [  
    1519951201,  
    43.78,  
    1.2269  
  ],  
  [  
    1519952401,  
    43.86,  
    1.2273  
  ],  
  [  
    1519953600,  
    43.85,  
    1.2273  
  ],  
  [  
    1519954800,  
    43.81,  
    1.2271  
  ],  
  [  
    1519956001,  
    44,  
    1.2275  
  ],  
  [  
    1519957200,  
    43.89,  
    1.2274  
  ],  
  [  
    1519958401,  
    43.95,  
    1.2273  
  ],  
  ],
```

```
[
  1519959601,
  43.93,
  1.2273
],
[
  1519960800,
  43.86,
  1.2276
],
[
  1519962000,
  44.02,
  1.2278
],
[
  1519963200,
  44.18,
  1.228
],
[
  1519964401,
  44.52,
  1.2283
],
[
  1519965600,
  44.19,
  1.2281
],
[
  1519966801,
  44.14,
  1.2278
],
[
  1519968000,
  43.93,
  1.2276
],
[
  1519969201,
  43.82,
  1.2277
],
[
  1519970401,
  43.77,
  1.2279
],
[
  1519971601,
  43.02,
  1.2269
],
[
  1519972801,
  42.99,
```



```
    1.2265
  ],
  [
    1519974001,
    42.73,
    1.2263
  ],
  [
    1519975201,
    42.22,
    1.2262
  ],
  [
    1519976400,
    42.13,
    1.2255
  ],
  [
    1519977601,
    42.02,
    1.2263
  ],
  [
    1519978801,
    42.15,
    1.2261
  ],
  [
    1519980000,
    42.5,
    1.2273
  ],
  [
    1519981201,
    42.2,
    1.2274
  ],
  [
    1519982400,
    42.06,
    1.2271
  ],
  [
    1519983600,
    42.38,
    1.2279
  ],
  [
    1519984800,
    42.29,
    1.2276
  ],
  [
    1519986000,
    42.16,
    1.2281
  ],
  [
```

```
    1519987201,  
    43.46,  
    1.2291  
  ],  
  [  
    1519988401,  
    43.51,  
    1.2291  
  ],  
  [  
    1519989601,  
    43.4,  
    1.2317  
  ],  
  [  
    1519990800,  
    43.46,  
    1.2317  
  ],  
  [  
    1519992001,  
    43.07,  
    1.2304  
  ],  
  [  
    1519993201,  
    43.56,  
    1.2316  
  ],  
  [  
    1519994401,  
    43.75,  
    1.2319  
  ],  
  [  
    1519995601,  
    43.15,  
    1.2308  
  ],  
  [  
    1519996801,  
    42.94,  
    1.2309  
  ],  
  [  
    1519998001,  
    42.99,  
    1.2315  
  ],  
  [  
    1519999201,  
    42.33,  
    1.2309  
  ],  
  [  
    1520000400,  
    41.93,  
    1.2299
```

```

    ],
    [
      1520001601,
      42.31,
      1.2303
    ],
    [
      1520002801,
      42.5,
      1.2313
    ],
    [
      1520004000,
      42.8,
      1.2326
    ],
    [
      1520005201,
      42.67,
      1.2317
    ],
    [
      1520006401,
      42.29,
      1.2309
    ],
    [
      1520007600,
      42.33,
      1.2309
    ],
    [
      1520008800,
      42.63,
      1.2321
    ],
    [
      1520010001,
      42.11,
      1.2314
    ]
  ],
  "label": "EUR/USD"
}
}
}

```

3.2.5 OrderbookData

class oandapyV20.endpoints.forexlabs.**OrderbookData** (*params*)

Bases: oandapyV20.endpoints.forexlabs.ForexLabs

OrderbookData.

Get the 'orderbook data' for an instrument.

ENDPOINT = 'labs/v1/orderbook_data'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(params)

Instantiate an OrderbookData request.

Parameters params (*dict (required)*) – query params to send, check developer.oanda.com for details.

```
>>> import oandapyV20
>>> import oandapyV20.endpoints.forexlab as labs
>>> accountID = ...
>>> client = oandapyV20.API(access_token=...)
>>> params =
    {
        "instrument": "EUR_USD",
        "period": 3600
    }
```

```
>>> r = labs.CommitmentOfTraders(params=params)
>>> client.request(r)
>>> print(r.response)
```

Output:

```
{
  "1520066400": {
    "rate": 1.2318,
    "price_points": {
      "1.23": {
        "ps": 1.2155,
        "ol": 0.3871,
        "os": 0.2615,
        "pl": 0.5633
      },
      "1.223": {
        "ps": 1.1266,
        "ol": 0.5021,
        "os": 0.2197,
        "pl": 0.3854
      },
      "1.288": {
        "ps": 0,
        "ol": 0.0105,
        "os": 0.0105,
        "pl": 0
      },
      "1.22": {
        "ps": 0.9191,
        "ol": 0.6486,
        "os": 0.136,
        "pl": 0.2965
      },
      "1.2245": {
        "ps": 0.5336,
        "ol": 0.5021,
        "os": 0.3975,
        "pl": 0.4447
      }
    }
  }
}
```



```
>>> client = oandapyV20.API(access_token=...)
>>> params =
      {
        "instrument": "EUR_USD",
        "period": 57600
      }
```

```
>>> r = labs.Spreads(params=params)
>>> client.request(r)
>>> print(r.response)
```

Output:

```
{
  "max": [
    [
      1520028000,
      6
    ]
  ],
  "avg": [
    [
      1520028000,
      3.01822
    ]
  ],
  "min": [
    [
      1520028000,
      1.7
    ]
  ]
}
```

3.3 oandapyV20.endpoints.instruments

3.3.1 InstrumentsCandles

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"
}

```

3.3.2 InstrumentsOrderBook

class oandapyV20.endpoints.instruments.**InstrumentsOrderBook** (*instrument*, *params=None*)

Bases: oandapyV20.endpoints.instruments.Instruments

Get orderbook data for a specified Instrument.

ENDPOINT = 'v3/instruments/{instrument}/orderBook'

EXPECTED_STATUS = 200

METHOD = 'GET'

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

Instantiate an InstrumentsOrderBook 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:

```
{}
```

OrderBook data example:

```

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

```



```
>>> params = ...
>>> r = instruments.InstrumentsOrderBook(instrument="EUR_USD",
>>>                                     params=params)
>>> client.request(r)
>>> print r.response
```

Output:

```
{
  "orderBook": {
    "instrument": "EUR_USD",
    "buckets": [
      {
        "price": "1.12850",
        "shortCountPercent": "0.2352",
        "longCountPercent": "0.2666"
      },
      {
        "price": "1.12900",
        "shortCountPercent": "0.2195",
        "longCountPercent": "0.3293"
      },
      {
        "price": "1.12950",
        "shortCountPercent": "0.3136",
        "longCountPercent": "0.2901"
      },
      {
        "price": "1.13000",
        "shortCountPercent": "0.3842",
        "longCountPercent": "0.4156"
      },
      {
        "price": "1.13050",
        "shortCountPercent": "0.1960",
        "longCountPercent": "0.3685"
      },
      {
        "price": "1.13100",
        "shortCountPercent": "0.2431",
        "longCountPercent": "0.2901"
      },
      {
        "price": "1.13150",
        "shortCountPercent": "0.2509",
        "longCountPercent": "0.3136"
      },
      {
        "price": "1.13200",
        "shortCountPercent": "0.2587",
        "longCountPercent": "0.3450"
      },
      {
        "price": "1.13250",
        "shortCountPercent": "0.3842",
        "longCountPercent": "0.2666"
      },
      {

```

```
"price": "1.13300",
"shortCountPercent": "0.3371",
"longCountPercent": "0.3371"
},
{
  "price": "1.13350",
  "shortCountPercent": "0.3528",
  "longCountPercent": "0.2744"
},
{
  "price": "1.13400",
  "shortCountPercent": "0.3842",
  "longCountPercent": "0.3136"
},
{
  "price": "1.13450",
  "shortCountPercent": "0.2039",
  "longCountPercent": "0.2744"
},
{
  "price": "1.13500",
  "shortCountPercent": "0.1882",
  "longCountPercent": "0.3371"
},
{
  "price": "1.13550",
  "shortCountPercent": "0.0235",
  "longCountPercent": "0.0392"
},
{
  "price": "1.13600",
  "shortCountPercent": "0.0549",
  "longCountPercent": "0.0314"
},
{
  "price": "1.13650",
  "shortCountPercent": "0.1333",
  "longCountPercent": "0.0314"
},
{
  "price": "1.13700",
  "shortCountPercent": "0.1176",
  "longCountPercent": "0.1019"
},
{
  "price": "1.13750",
  "shortCountPercent": "0.1568",
  "longCountPercent": "0.0784"
},
{
  "price": "1.13800",
  "shortCountPercent": "0.1176",
  "longCountPercent": "0.0862"
},
{
  "price": "1.13850",
  "shortCountPercent": "0.2117",
  "longCountPercent": "0.1960"
```

```

    },
    {
      "price": "1.13900",
      "shortCountPercent": "0.4548",
      "longCountPercent": "0.2587"
    },
    {
      "price": "1.13950",
      "shortCountPercent": "0.2979",
      "longCountPercent": "0.3215"
    },
    {
      "price": "1.14000",
      "shortCountPercent": "0.7449",
      "longCountPercent": "0.2901"
    },
    {
      "price": "1.14050",
      "shortCountPercent": "0.2117",
      "longCountPercent": "0.1176"
    },
    {
      "price": "1.14100",
      "shortCountPercent": "0.1960",
      "longCountPercent": "0.1333"
    },
    {
      "price": "1.14150",
      "shortCountPercent": "0.1882",
      "longCountPercent": "0.1176"
    }
  ],
  "time": "2017-06-28T10:00:00Z",
  "price": "1.13609",
  "bucketWidth": "0.00050"
}

```

3.3.3 InstrumentsPositionBook

class oandapyV20.endpoints.instruments.**InstrumentsPositionBook** (*instrument*, *params=None*)

Bases: oandapyV20.endpoints.instruments.Instruments

Get positionbook data for a specified Instrument.

ENDPOINT = 'v3/instruments/{instrument}/positionBook'

EXPECTED_STATUS = 200

METHOD = 'GET'

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

Instantiate an InstrumentsPositionBook 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:

```
{}
```

PositionBook data example:

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

Output:

```
{
  "positionBook": {
    "instrument": "EUR_USD",
    "buckets": [
      {
        "price": "1.12800",
        "shortCountPercent": "0.2670",
        "longCountPercent": "0.2627"
      },
      {
        "price": "1.12850",
        "shortCountPercent": "0.2034",
        "longCountPercent": "0.2712"
      },
      {
        "price": "1.12900",
        "shortCountPercent": "0.2034",
        "longCountPercent": "0.2161"
      },
      {
        "price": "1.12950",
        "shortCountPercent": "0.2670",
        "longCountPercent": "0.2839"
      },
      {
        "price": "1.13000",
        "shortCountPercent": "0.2755",
        "longCountPercent": "0.3221"
      },
      {
        "price": "1.13050",
        "shortCountPercent": "0.1949",
        "longCountPercent": "0.2839"
      },
      {
        "price": "1.13100",
        "shortCountPercent": "0.2288",
        "longCountPercent": "0.2712"
      },
    ]
  }
}
```

```
{
  "price": "1.13150",
  "shortCountPercent": "0.2416",
  "longCountPercent": "0.2712"
},
{
  "price": "1.13200",
  "shortCountPercent": "0.2204",
  "longCountPercent": "0.3178"
},
{
  "price": "1.13250",
  "shortCountPercent": "0.2543",
  "longCountPercent": "0.2458"
},
{
  "price": "1.13300",
  "shortCountPercent": "0.2839",
  "longCountPercent": "0.2585"
},
{
  "price": "1.13350",
  "shortCountPercent": "0.3602",
  "longCountPercent": "0.3094"
},
{
  "price": "1.13400",
  "shortCountPercent": "0.2882",
  "longCountPercent": "0.3560"
},
{
  "price": "1.13450",
  "shortCountPercent": "0.2500",
  "longCountPercent": "0.3009"
},
{
  "price": "1.13500",
  "shortCountPercent": "0.1738",
  "longCountPercent": "0.3475"
},
{
  "price": "1.13550",
  "shortCountPercent": "0.2119",
  "longCountPercent": "0.2797"
},
{
  "price": "1.13600",
  "shortCountPercent": "0.1483",
  "longCountPercent": "0.3094"
},
{
  "price": "1.13650",
  "shortCountPercent": "0.1483",
  "longCountPercent": "0.1314"
},
{
  "price": "1.13700",
  "shortCountPercent": "0.1568",
```

```
    "longCountPercent": "0.2034"
  },
  {
    "price": "1.13750",
    "shortCountPercent": "0.1398",
    "longCountPercent": "0.1271"
  },
  {
    "price": "1.13800",
    "shortCountPercent": "0.1314",
    "longCountPercent": "0.2034"
  },
  {
    "price": "1.13850",
    "shortCountPercent": "0.1483",
    "longCountPercent": "0.1695"
  },
  {
    "price": "1.13900",
    "shortCountPercent": "0.2924",
    "longCountPercent": "0.1653"
  },
  {
    "price": "1.13950",
    "shortCountPercent": "0.1526",
    "longCountPercent": "0.1865"
  },
  {
    "price": "1.14000",
    "shortCountPercent": "0.4365",
    "longCountPercent": "0.2034"
  },
  {
    "price": "1.14050",
    "shortCountPercent": "0.1398",
    "longCountPercent": "0.1144"
  }
],
"time": "2017-06-28T10:00:00Z",
"price": "1.13609",
"bucketWidth": "0.00050"
}
```

3.4 oandapyV20.endpoints.orders

3.4.1 OrderCancel

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"
  ]
}
```

3.4.2 OrderClientExtensions

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"
  ]
}
```

3.4.3 OrderCreate

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"
  ]
}
```

3.4.4 OrderDetails

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"
}
```

3.4.5 OrderList

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"
}
```

3.4.6 OrderReplace

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 creating 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"
  ]
}

```

3.4.7 OrdersPending

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"
}

```

3.5 oandapyV20.endpoints.positions

3.5.1 OpenPositions

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"
}

```

3.5.2 PositionClose

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>"
    }
  }
}

```

3.5.3 PositionDetails

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"
}
```

3.5.4 PositionList

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"
}

```

3.6 oandapyV20.endpoints.pricing

3.6.1 PricingInfo

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"
    }
}
]
}

```

3.6.2 PricingStream

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",
  "instrument": "EUR_JPY",
  "asks": [
    {
      "price": "114.312",
      "liquidity": 1000000
    },
    {
      "price": "114.313",
      "liquidity": 2000000
    },
    {
      "price": "114.314",
      "liquidity": 5000000
    },
    {
      "price": "114.316",
      "liquidity": 10000000
    }
  ],
  "time": "2016-10-27T08:38:43.094548890Z",
  "closeoutAsk": "114.316",
  "type": "PRICE",
  "closeoutBid": "114.291",
  "bids": [
    {
      "price": "114.295",
      "liquidity": 1000000
    },
    {
      "price": "114.294",
      "liquidity": 2000000
    },
    {
      "price": "114.293",
      "liquidity": 5000000
    },
    {
      "price": "114.291",
      "liquidity": 10000000
    }
  ]
},
{
  "type": "HEARTBEAT",
  "time": "2016-10-27T08:38:44.327443673Z"
},
{
  "status": "tradeable",
  "instrument": "EUR_USD",
  "asks": [
    {
      "price": "1.09188",
      "liquidity": 10000000
    }
  ]
}
```



```

    },
    {
      "price": "1.09190",
      "liquidity": 10000000
    }
  ],
  "time": "2016-10-27T08:38:45.664613867Z",
  "closeoutAsk": "1.09192",
  "type": "PRICE",
  "closeoutBid": "1.09173",
  "bids": [
    {
      "price": "1.09177",
      "liquidity": 10000000
    },
    {
      "price": "1.09175",
      "liquidity": 10000000
    }
  ]
},
{
  "status": "tradeable",
  "instrument": "EUR_JPY",
  "asks": [
    {
      "price": "114.315",
      "liquidity": 1000000
    },
    {
      "price": "114.316",
      "liquidity": 2000000
    },
    {
      "price": "114.317",
      "liquidity": 5000000
    },
    {
      "price": "114.319",
      "liquidity": 10000000
    }
  ],
  "time": "2016-10-27T08:38:45.681572782Z",
  "closeoutAsk": "114.319",
  "type": "PRICE",
  "closeoutBid": "114.294",
  "bids": [
    {
      "price": "114.298",
      "liquidity": 1000000
    },
    {
      "price": "114.297",
      "liquidity": 2000000
    },
    {
      "price": "114.296",
      "liquidity": 5000000
    }
  ]
}

```

```
    },  
    {  
      "price": "114.294",  
      "liquidity": 10000000  
    }  
  ]  
}
```

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

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

3.7 oandapyV20.endpoints.trades

3.7.1 OpenTrades

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"
  }

```

3.7.2 TradeCRCDO

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"  
  }  
}
```

3.7.3 TradeClientExtensions

```
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"
  ]
}

```

3.7.4 TradeClose

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"
  ]
}

```

3.7.5 TradeDetails

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"
  }
}
```

3.7.6 TradesList

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"
}
```

3.8 oandapyV20.endpoints.transactions

3.8.1 TransactionDetails

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"
}
```

3.8.2 TransactionIDRange

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"
  }
]
}

```

3.8.3 TransactionList

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?from=1&to=100",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=101&to=200",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=201&to=300",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=301&to=400",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=401&to=500",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=501&to=600",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=601&to=700",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=701&to=800",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=801&to=900",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=901&to=1000",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1001&to=1100",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1101&to=1200",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1201&to=1300",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1301&to=1400",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1401&to=1500",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1501&to=1600",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
    ↪transactions/idrange?from=1601&to=1700",
```

```

    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
↪transactions/idrange?from=1701&to=1800",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
↪transactions/idrange?from=1801&to=1900",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
↪transactions/idrange?from=1901&to=2000",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
↪transactions/idrange?from=2001&to=2100",
    "https://api-fxpractice.oanda.com/v3/accounts/101-004-1435156-001/
↪transactions/idrange?from=2101&to=2124"
  ]
}

```

3.8.4 TransactionsSinceID

```
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"
    }
]
}

```

3.8.5 TransactionsStream

```
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.

oandapyV20.definitions

The `oandapyV20.definitions` module holds all the definitions as in the definitions section of the REST-V20 specs of OANDA, see developer.oanda.com.

4.1 oandapyV20.definitions.accounts

Account Definitions.

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 the the Trade is closed
>>> # or
>>> print defaccounts.AccountFinancingMode().definitions[c.SECOND_BY_SECOND]
>>> # all keys
>>> print defaccounts.AccountFinancingMode().definitions.keys()
>>> ...
```

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 value (long or short) is used to compute the Position value or
↳margin.
>>> # or
>>> print defaccounts.PositionAggregationMode().definitions[c.NET_SUM]
>>> # all keys
>>> print defaccounts.PositionAggregationMode().definitions.keys()
>>> ...
```

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

Instruments Definitions.

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
>>> # or
>>> print definstruments.CandlestickGranularity().definitions[c.H4]
>>> # all keys
>>> print definstruments.CandlestickGranularity().definitions.keys()
>>> ...
```

D = 'D'

```

H1 = 'H1'
H12 = 'H12'
H2 = 'H2'
H3 = 'H3'
H4 = 'H4'
H6 = 'H6'
H8 = 'H8'
M = 'M'
M1 = 'M1'
M10 = 'M10'
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
>>> # or
>>> print definstruments.WeeklyAlignment().definitions[c.Monday]
>>> # all keys
>>> print definstruments.WeeklyAlignment().definitions.keys()
>>> ...

```

```

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.
```

```
class oandapyV20.definitions.instruments.PriceComponents
    Bases: object
```

Definition representation of PriceComponents

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.PriceComponents.A
A
>>> c = definstruments.PriceComponents()
>>> print c[c.A]
Ask
>>> # or
>>> print definstruments.PriceComponents().definitions[c.A]
>>> # all keys
>>> print definstruments.PriceComponents().definitions.keys()
>>> ...
```

```
A = 'A'
B = 'B'
M = 'M'
__getitem__(definitionID)
    return description for definitionID.

definitions
    readonly property holding definition dict.
```

4.3 oandapyV20.definitions.orders

Order related definitions.

```
class oandapyV20.definitions.orders.OrderStateFilter
    Bases: object
```

Definition representation of OrderStateFilter

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.OrderStateFilter.CANCELLED
CANCELLED
>>> c = deforders.OrderStateFilter()
>>> print c[c.CANCELLED]
The orders that have been cancelled
>>> # or
>>> print deforders.OrderStateFilter().definitions[c.CANCELLED]
>>> # all keys
>>> print deforders.OrderStateFilter().definitions.keys()
>>> ...

```

ALL = 'ALL'

CANCELLED = 'CANCELLED'

FILLED = 'FILLED'

PENDING = 'PENDING'

TRIGGERED = 'TRIGGERED'

__getitem__ (*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

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
>>> # or
>>> print deforders.OrderType().definitions[c.MARKET_IF_TOUCHED]
>>> # all keys
>>> print deforders.OrderType().definitions.keys()
>>> ...

```

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.CancellableOrderType

Bases: object

Definition representation of CancellableOrderType

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.CancellableOrderType.MARKET_IF_TOUCHED
MARKET_IF_TOUCHED
>>> c = deforders.CancellableOrderType()
>>> print c[c.MARKET_IF_TOUCHED]
A Market-if-touched Order
>>> # or
>>> print deforders.CancellableOrderType().definitions[c.MARKET_IF_TOUCHED]
>>> # all keys
>>> print deforders.CancellableOrderType().definitions.keys()
>>> ...
```

LIMIT = 'LIMIT'

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.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.
>>> # or
>>> print deforders.OrderPositionFill().definitions[c.REDUCE_ONLY]
>>> # all keys
>>> print deforders.OrderPositionFill().definitions.keys()
>>> ...
```

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.
```

```
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"
>>> # or
>>> print deforders.TimeInForce().definitions[c.IOC]
>>> # all keys
>>> print deforders.TimeInForce().definitions.keys()
>>> ...
```

```
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.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
>>> # or
>>> print deforders.OrderState().definitions[c.CANCELLED]
>>> # all keys
>>> print deforders.OrderState().definitions.keys()
>>> ...
```

```
CANCELLED = 'CANCELLED'
```

```

FILLED = 'FILLED'
PENDING = 'PENDING'
TRIGGERED = 'TRIGGERED'
__getitem__(definitionID)
    return description for definitionID.

```

```

definitions
    readonly property holding definition dict.

```

```
class oandapyV20.definitions.orders.OrderTriggerCondition
```

```
Bases: object
```

Definition representation of OrderTriggerCondition

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.OrderTriggerCondition.DEFAULT
DEFAULT
>>> c = deforders.OrderTriggerCondition()
>>> print c[c.DEFAULT]
Trigger an Order the "natural" way: compare its price to the ask for long Orders,
↳and bid for short Orders
>>> # or
>>> print deforders.OrderTriggerCondition().definitions[c.DEFAULT]
>>> # all keys
>>> print deforders.OrderTriggerCondition().definitions.keys()
>>> ...

```

```

ASK = 'ASK'
BID = 'BID'
DEFAULT = 'DEFAULT'
INVERSE = 'INVERSE'
MID = 'MID'
__getitem__(definitionID)
    return description for definitionID.
definitions
    readonly property holding definition dict.

```

4.4 oandapyV20.definitions.pricing

Pricing related Definitions.

```
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.
>>> # or
>>> print defpricing.PriceStatus().definitions[c.non_tradeable]
>>> # all keys
>>> print defpricing.PriceStatus().definitions.keys()
>>> ...

```

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

Trades definitions.

class oandapyV20.definitions.trades.TradePL

Bases: object

Definition representation of TradePL

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.TradePL.POSITIVE
POSITIVE
>>> c = deftrades.TradePL()
>>> print c[c.POSITIVE]
An open Trade currently has a positive (profitable) unrealized P/L, or a closed_
↪Trade realized a positive amount of P/L.
>>> # or
>>> print deftrades.TradePL().definitions[c.POSITIVE]
>>> # all keys
>>> print deftrades.TradePL().definitions.keys()
>>> ...

```

```
NEGATIVE = 'NEGATIVE'
```

```
POSITIVE = 'POSITIVE'
```

```
ZERO = 'ZERO'
```

```

__getitem__(definitionID)
    return description for definitionID.

```

definitions

readonly property holding definition dict.

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
>>> # or
>>> print deftrades.TradeState().definitions[c.CLOSE_WHEN_TRADABLE]
>>> # all keys
>>> print deftrades.TradeState().definitions.keys()
>>> ...
```

CLOSED = 'CLOSED'

CLOSE_WHEN_TRADABLE = 'CLOSE_WHEN_TRADABLE'

OPEN = 'OPEN'

__getitem__ (*definitionID*)

return description for definitionID.

definitions

readonly property holding definition dict.

class oandapyV20.definitions.trades.TradeStateFilter

Bases: object

Definition representation of TradeStateFilter

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.TradeStateFilter.CLOSE_WHEN_TRADEABLE
CLOSE_WHEN_TRADEABLE
>>> c = deftrades.TradeStateFilter()
>>> print c[c.CLOSE_WHEN_TRADEABLE]
The Trades that will be closed as soon as the trades' instrument becomes tradeable
>>> # or
>>> print deftrades.TradeStateFilter().definitions[c.CLOSE_WHEN_TRADEABLE]
>>> # all keys
>>> print deftrades.TradeStateFilter().definitions.keys()
>>> ...
```

ALL = 'ALL'

CLOSED = 'CLOSED'

CLOSE_WHEN_TRADEABLE = 'CLOSE_WHEN_TRADEABLE'

OPEN = 'OPEN'

__getitem__ (*definitionID*)

return description for definitionID.

definitions

readonly property holding definition dict.

4.6 oandapyV20.definitions.transactions

Transactions definitions.

class oandapyV20.definitions.transactions.**MarketOrderMarginCloseoutReason**

Bases: object

Definition representation of MarketOrderMarginCloseoutReason

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.MarketOrderMarginCloseoutReason.MARGIN_CHECK_VIOLATION
MARGIN_CHECK_VIOLATION
>>> c = deftransactions.MarketOrderMarginCloseoutReason()
>>> print c[c.MARGIN_CHECK_VIOLATION]
Trade closures resulted from violating OANDA's margin policy
>>> # or
>>> print deftransactions.MarketOrderMarginCloseoutReason().definitions[c.MARGIN_
↪CHECK_VIOLATION]
>>> # all keys
>>> print deftransactions.MarketOrderMarginCloseoutReason().definitions.keys()
>>> ...
```

MARGIN_CHECK_VIOLATION = 'MARGIN_CHECK_VIOLATION'

REGULATORY_MARGIN_CALL_VIOLATION = 'REGULATORY_MARGIN_CALL_VIOLATION'

__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 requiring a Stop Loss Order.
>>> # or
>>> print deftransactions.StopLossOrderReason().definitions[c.ON_FILL]
>>> # all keys
>>> print deftransactions.StopLossOrderReason().definitions.keys()
>>> ...
```

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
>>> # or
>>> print deftransactions.OrderFillReason().definitions[c.STOP_ORDER]
>>> # all keys
>>> print deftransactions.OrderFillReason().definitions.keys()
>>> ...
```

```
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.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.
>>> # or
>>> print deftransactions.FundingReason().definitions[c.ACCOUNT_TRANSFER]
>>> # all keys
>>> print deftransactions.FundingReason().definitions.keys()
>>> ...

```

```
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.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
>>> # or
>>> print deftransactions.MarketIfTouchedOrderReason().definitions[c.CLIENT_ORDER]
>>> # all keys
>>> print deftransactions.MarketIfTouchedOrderReason().definitions.keys()
>>> ...

```

```
CLIENT_ORDER = 'CLIENT_ORDER'
```

```
REPLACEMENT = 'REPLACEMENT'
```

```
__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
>>> # or
>>> print deftransactions.MarketOrderReason().definitions[c.TRADE_CLOSE]
>>> # all keys
>>> print deftransactions.MarketOrderReason().definitions.keys()
>>> ...

```

```
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.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
>>> # or
>>> print deftransactions.StopOrderReason().definitions[c.CLIENT_ORDER]
>>> # all keys
>>> print deftransactions.StopOrderReason().definitions.keys()
>>> ...

```

```
CLIENT_ORDER = 'CLIENT_ORDER'
```

```
REPLACEMENT = 'REPLACEMENT'
```

```
__getitem__(definitionID)
    return description for definitionID.
```

```
definitions
    readonly property holding definition dict.
```

```
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
>>> # or
>>> print deftransactions.TransactionType().definitions[c.STOP_LOSS_ORDER]
>>> # all keys
>>> print deftransactions.TransactionType().definitions.keys()
>>> ...

```

```

CLIENT_CONFIGURE = 'CLIENT_CONFIGURE'
CLIENT_CONFIGURE_REJECT = 'CLIENT_CONFIGURE_REJECT'
CLOSE = 'CLOSE'
CREATE = 'CREATE'
DAILY_FINANCING = 'DAILY_FINANCING'
DELAYED_TRADE_CLOSURE = 'DELAYED_TRADE_CLOSURE'
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_CANCEL_REJECT = 'ORDER_CANCEL_REJECT'
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.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 Trade requiring a Take Profit Order.
>>> # or
>>> print deftransactions.TakeProfitOrderReason().definitions[c.ON_FILL]
>>> # all keys
>>> print deftransactions.TakeProfitOrderReason().definitions.keys()
>>> ...

```

```

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.TransactionRejectReason
Bases: object

Definition representation of TransactionRejectReason

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.TransactionRejectReason.STOP_LOSS_ON_FILL_GTD_TIMESTAMP_
↳MISSING
STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING
>>> c = deftransactions.TransactionRejectReason()
>>> print c[c.STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING]
The Stop Loss on fill specifies a GTD TimeInForce but does not provide a GTD_
↳timestamp

```

```

>>> # or
>>> print deftransactions.TransactionRejectReason().definitions[c.STOP_LOSS_ON_
↳FILL_GTD_TIMESTAMP_MISSING]
>>> # all keys
>>> print deftransactions.TransactionRejectReason().definitions.keys()
>>> ...

```

```

ACCOUNT_CONFIGURATION_LOCKED = 'ACCOUNT_CONFIGURATION_LOCKED'
ACCOUNT_DEPOSIT_LOCKED = 'ACCOUNT_DEPOSIT_LOCKED'
ACCOUNT_LOCKED = 'ACCOUNT_LOCKED'
ACCOUNT_NOT_ACTIVE = 'ACCOUNT_NOT_ACTIVE'
ACCOUNT_ORDER_CANCEL_LOCKED = 'ACCOUNT_ORDER_CANCEL_LOCKED'
ACCOUNT_ORDER_CREATION_LOCKED = 'ACCOUNT_ORDER_CREATION_LOCKED'
ACCOUNT_WITHDRAWAL_LOCKED = 'ACCOUNT_WITHDRAWAL_LOCKED'
ADMIN_CONFIGURE_DATA_MISSING = 'ADMIN_CONFIGURE_DATA_MISSING'
ALIAS_INVALID = 'ALIAS_INVALID'
AMOUNT_INVALID = 'AMOUNT_INVALID'
AMOUNT_MISSING = 'AMOUNT_MISSING'
CLIENT_CONFIGURE_DATA_MISSING = 'CLIENT_CONFIGURE_DATA_MISSING'
CLIENT_EXTENSIONS_DATA_MISSING = 'CLIENT_EXTENSIONS_DATA_MISSING'
CLIENT_ORDER_COMMENT_INVALID = 'CLIENT_ORDER_COMMENT_INVALID'
CLIENT_ORDER_ID_ALREADY_EXISTS = 'CLIENT_ORDER_ID_ALREADY_EXISTS'
CLIENT_ORDER_ID_INVALID = 'CLIENT_ORDER_ID_INVALID'
CLIENT_ORDER_TAG_INVALID = 'CLIENT_ORDER_TAG_INVALID'
CLIENT_TRADE_COMMENT_INVALID = 'CLIENT_TRADE_COMMENT_INVALID'
CLIENT_TRADE_ID_ALREADY_EXISTS = 'CLIENT_TRADE_ID_ALREADY_EXISTS'
CLIENT_TRADE_ID_INVALID = 'CLIENT_TRADE_ID_INVALID'
CLIENT_TRADE_TAG_INVALID = 'CLIENT_TRADE_TAG_INVALID'
CLOSEOUT_POSITION_DOESNT_EXIST = 'CLOSEOUT_POSITION_DOESNT_EXIST'
CLOSEOUT_POSITION_INCOMPLETE_SPECIFICATION = 'CLOSEOUT_POSITION_INCOMPLETE_SPECIFICATI
CLOSEOUT_POSITION_PARTIAL_UNITS_MISSING = 'CLOSEOUT_POSITION_PARTIAL_UNITS_MISSING'
CLOSEOUT_POSITION_REJECT = 'CLOSEOUT_POSITION_REJECT'
CLOSEOUT_POSITION_UNITS_EXCEED_POSITION_SIZE = 'CLOSEOUT_POSITION_UNITS_EXCEED_POSITION
CLOSE_TRADE_PARTIAL_UNITS_MISSING = 'CLOSE_TRADE_PARTIAL_UNITS_MISSING'
CLOSE_TRADE_TYPE_MISSING = 'CLOSE_TRADE_TYPE_MISSING'
CLOSE_TRADE_UNITS_EXCEED_TRADE_SIZE = 'CLOSE_TRADE_UNITS_EXCEED_TRADE_SIZE'
FUNDING_REASON_MISSING = 'FUNDING_REASON_MISSING'
INSTRUMENT_MISSING = 'INSTRUMENT_MISSING'

```

INSTRUMENT_NOT_TRADEABLE = 'INSTRUMENT_NOT_TRADEABLE'
INSTRUMENT_PRICE_UNKNOWN = 'INSTRUMENT_PRICE_UNKNOWN'
INSTRUMENT_UNKNOWN = 'INSTRUMENT_UNKNOWN'
INSUFFICIENT_FUNDS = 'INSUFFICIENT_FUNDS'
INTERNAL_SERVER_ERROR = 'INTERNAL_SERVER_ERROR'
INVALID_REISSUE_IMMEDIATE_PARTIAL_FILL = 'INVALID_REISSUE_IMMEDIATE_PARTIAL_FILL'
MARGIN_RATE_INVALID = 'MARGIN_RATE_INVALID'
MARGIN_RATE_WOULD_TRIGGER_CLOSEOUT = 'MARGIN_RATE_WOULD_TRIGGER_CLOSEOUT'
MARGIN_RATE_WOULD_TRIGGER_MARGIN_CALL = 'MARGIN_RATE_WOULD_TRIGGER_MARGIN_CALL'
MARKUP_GROUP_ID_INVALID = 'MARKUP_GROUP_ID_INVALID'
ORDERS_ON_FILL_DUPLICATE_CLIENT_ORDER_IDS = 'ORDERS_ON_FILL_DUPLICATE_CLIENT_ORDER_IDS'
ORDER_DOESNT_EXIST = 'ORDER_DOESNT_EXIST'
ORDER_FILL_POSITION_ACTION_INVALID = 'ORDER_FILL_POSITION_ACTION_INVALID'
ORDER_FILL_POSITION_ACTION_MISSING = 'ORDER_FILL_POSITION_ACTION_MISSING'
ORDER_IDENTIFIER_INCONSISTENCY = 'ORDER_IDENTIFIER_INCONSISTENCY'
ORDER_ID_UNSPECIFIED = 'ORDER_ID_UNSPECIFIED'
ORDER_PARTIAL_FILL_OPTION_INVALID = 'ORDER_PARTIAL_FILL_OPTION_INVALID'
ORDER_PARTIAL_FILL_OPTION_MISSING = 'ORDER_PARTIAL_FILL_OPTION_MISSING'
PENDING_ORDERS_ALLOWED_EXCEEDED = 'PENDING_ORDERS_ALLOWED_EXCEEDED'
POSITION_AGGREGATION_MODE_INVALID = 'POSITION_AGGREGATION_MODE_INVALID'
PRICE_BOUND_INVALID = 'PRICE_BOUND_INVALID'
PRICE_BOUND_PRECISION_EXCEEDED = 'PRICE_BOUND_PRECISION_EXCEEDED'
PRICE_DISTANCE_INVALID = 'PRICE_DISTANCE_INVALID'
PRICE_DISTANCE_MAXIMUM_EXCEEDED = 'PRICE_DISTANCE_MAXIMUM_EXCEEDED'
PRICE_DISTANCE_MINIMUM_NOT_MET = 'PRICE_DISTANCE_MINIMUM_NOT_MET'
PRICE_DISTANCE_MISSING = 'PRICE_DISTANCE_MISSING'
PRICE_DISTANCE_PRECISION_EXCEEDED = 'PRICE_DISTANCE_PRECISION_EXCEEDED'
PRICE_INVALID = 'PRICE_INVALID'
PRICE_MISSING = 'PRICE_MISSING'
PRICE_PRECISION_EXCEEDED = 'PRICE_PRECISION_EXCEEDED'
REPLACING_ORDER_INVALID = 'REPLACING_ORDER_INVALID'
REPLACING_TRADE_ID_INVALID = 'REPLACING_TRADE_ID_INVALID'
STOP_LOSS_ON_FILL_CLIENT_ORDER_COMMENT_INVALID = 'STOP_LOSS_ON_FILL_CLIENT_ORDER_COMMENT_INVALID'
STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_INVALID = 'STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_INVALID'
STOP_LOSS_ON_FILL_CLIENT_ORDER_TAG_INVALID = 'STOP_LOSS_ON_FILL_CLIENT_ORDER_TAG_INVALID'
STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST'

STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING = 'STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING'
STOP_LOSS_ON_FILL_PRICE_INVALID = 'STOP_LOSS_ON_FILL_PRICE_INVALID'
STOP_LOSS_ON_FILL_PRICE_MISSING = 'STOP_LOSS_ON_FILL_PRICE_MISSING'
STOP_LOSS_ON_FILL_PRICE_PRECISION_EXCEEDED = 'STOP_LOSS_ON_FILL_PRICE_PRECISION_EXCEEDED'
STOP_LOSS_ON_FILL_TIME_IN_FORCE_INVALID = 'STOP_LOSS_ON_FILL_TIME_IN_FORCE_INVALID'
STOP_LOSS_ON_FILL_TIME_IN_FORCE_MISSING = 'STOP_LOSS_ON_FILL_TIME_IN_FORCE_MISSING'
STOP_LOSS_ON_FILL_TRIGGER_CONDITION_INVALID = 'STOP_LOSS_ON_FILL_TRIGGER_CONDITION_INVALID'
STOP_LOSS_ON_FILL_TRIGGER_CONDITION_MISSING = 'STOP_LOSS_ON_FILL_TRIGGER_CONDITION_MISSING'
STOP_LOSS_ORDER_ALREADY_EXISTS = 'STOP_LOSS_ORDER_ALREADY_EXISTS'
TAKE_PROFIT_ON_FILL_CLIENT_ORDER_COMMENT_INVALID = 'TAKE_PROFIT_ON_FILL_CLIENT_ORDER_COMMENT_INVALID'
TAKE_PROFIT_ON_FILL_CLIENT_ORDER_ID_INVALID = 'TAKE_PROFIT_ON_FILL_CLIENT_ORDER_ID_INVALID'
TAKE_PROFIT_ON_FILL_CLIENT_ORDER_TAG_INVALID = 'TAKE_PROFIT_ON_FILL_CLIENT_ORDER_TAG_INVALID'
TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST'
TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_MISSING = 'TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_MISSING'
TAKE_PROFIT_ON_FILL_PRICE_INVALID = 'TAKE_PROFIT_ON_FILL_PRICE_INVALID'
TAKE_PROFIT_ON_FILL_PRICE_MISSING = 'TAKE_PROFIT_ON_FILL_PRICE_MISSING'
TAKE_PROFIT_ON_FILL_PRICE_PRECISION_EXCEEDED = 'TAKE_PROFIT_ON_FILL_PRICE_PRECISION_EXCEEDED'
TAKE_PROFIT_ON_FILL_TIME_IN_FORCE_INVALID = 'TAKE_PROFIT_ON_FILL_TIME_IN_FORCE_INVALID'
TAKE_PROFIT_ON_FILL_TIME_IN_FORCE_MISSING = 'TAKE_PROFIT_ON_FILL_TIME_IN_FORCE_MISSING'
TAKE_PROFIT_ON_FILL_TRIGGER_CONDITION_INVALID = 'TAKE_PROFIT_ON_FILL_TRIGGER_CONDITION_INVALID'
TAKE_PROFIT_ON_FILL_TRIGGER_CONDITION_MISSING = 'TAKE_PROFIT_ON_FILL_TRIGGER_CONDITION_MISSING'
TAKE_PROFIT_ORDER_ALREADY_EXISTS = 'TAKE_PROFIT_ORDER_ALREADY_EXISTS'
TIME_IN_FORCE_GTD_TIMESTAMP_IN_PAST = 'TIME_IN_FORCE_GTD_TIMESTAMP_IN_PAST'
TIME_IN_FORCE_GTD_TIMESTAMP_MISSING = 'TIME_IN_FORCE_GTD_TIMESTAMP_MISSING'
TIME_IN_FORCE_INVALID = 'TIME_IN_FORCE_INVALID'
TIME_IN_FORCE_MISSING = 'TIME_IN_FORCE_MISSING'
TRADE_DOESNT_EXIST = 'TRADE_DOESNT_EXIST'
TRADE_IDENTIFIER_INCONSISTENCY = 'TRADE_IDENTIFIER_INCONSISTENCY'
TRADE_ID_UNSPECIFIED = 'TRADE_ID_UNSPECIFIED'
TRADE_ON_FILL_CLIENT_EXTENSIONS_NOT_SUPPORTED = 'TRADE_ON_FILL_CLIENT_EXTENSIONS_NOT_SUPPORTED'
TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_COMMENT_INVALID = 'TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_COMMENT_INVALID'
TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_INVALID = 'TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_INVALID'
TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_TAG_INVALID = 'TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_TAG_INVALID'
TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST'
TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING = 'TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING'
TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_INVALID = 'TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_INVALID'

```

TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_MAXIMUM_EXCEEDED = 'TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_MAXIMUM_EXCEEDED'
TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_MINIMUM_NOT_MET = 'TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_MINIMUM_NOT_MET'
TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_MISSING = 'TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_MISSING'
TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_PRECISION_EXCEEDED = 'TRAILING_STOP_LOSS_ON_FILL PRICE_DISTANCE_PRECISION_EXCEEDED'
TRAILING_STOP_LOSS_ON_FILL TIME_IN_FORCE_INVALID = 'TRAILING_STOP_LOSS_ON_FILL TIME_IN_FORCE_INVALID'
TRAILING_STOP_LOSS_ON_FILL TIME_IN_FORCE_MISSING = 'TRAILING_STOP_LOSS_ON_FILL TIME_IN_FORCE_MISSING'
TRAILING_STOP_LOSS_ON_FILL TRIGGER_CONDITION_INVALID = 'TRAILING_STOP_LOSS_ON_FILL TRIGGER_CONDITION_INVALID'
TRAILING_STOP_LOSS_ON_FILL TRIGGER_CONDITION_MISSING = 'TRAILING_STOP_LOSS_ON_FILL TRIGGER_CONDITION_MISSING'
TRAILING_STOP_LOSS_ORDERS_NOT_SUPPORTED = 'TRAILING_STOP_LOSS_ORDERS_NOT_SUPPORTED'
TRAILING_STOP_LOSS_ORDER_ALREADY_EXISTS = 'TRAILING_STOP_LOSS_ORDER_ALREADY_EXISTS'
TRIGGER_CONDITION_INVALID = 'TRIGGER_CONDITION_INVALID'
TRIGGER_CONDITION_MISSING = 'TRIGGER_CONDITION_MISSING'
UNITS_INVALID = 'UNITS_INVALID'
UNITS_LIMIT_EXCEEDED = 'UNITS_LIMIT_EXCEEDED'
UNITS_MIMIMUM_NOT_MET = 'UNITS_MIMIMUM_NOT_MET'
UNITS_MISSING = 'UNITS_MISSING'
UNITS_PRECISION_EXCEEDED = 'UNITS_PRECISION_EXCEEDED'

```

```

__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.
>>> # or
>>> print deftransactions.OrderCancelReason().definitions[c.LINKED_TRADE_CLOSED]
>>> # all keys
>>> print deftransactions.OrderCancelReason().definitions.keys()
>>> ...

```

```

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_ALREADY_EXISTS'
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_ID_ALREADY_EXISTS'
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_CLIENT_ORDER_ID_ALREADY_EXISTS'
TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST'

```

```

__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 new Trade requiring a Trailing Stop Loss Order.

```

```
>>> # or
>>> print deftransactions.TrailingStopLossOrderReason().definitions[c.ON_FILL]
>>> # all keys
>>> print deftransactions.TrailingStopLossOrderReason().definitions.keys()
>>> ...
```

```
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.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
>>> # or
>>> print deftransactions.LimitOrderReason().definitions[c.CLIENT_ORDER]
>>> # all keys
>>> print deftransactions.LimitOrderReason().definitions.keys()
>>> ...
```

```
CLIENT_ORDER = 'CLIENT_ORDER'
```

```
REPLACEMENT = 'REPLACEMENT'
```

```
__getitem__(definitionID)
    return description for definitionID.
```

```
definitions
    readonly property holding definition dict.
```


The `oandapyV20.types` module contains the types representing the types that are used in the API-specs of OANDA, check developer.oanda.com. These types offer a convenient interface between Python types and the types used in the REST-API.

Take for instance the *PriceValue* type. It is the string representation of a float.

```
from oandapyV20.types import PriceValue

pv1 = PriceValue(122.345)
pv2 = PriceValue("122.345")
pv1.value
"122.345"
pv1.value == pv2.value
True
```

Regardless the value we instantiate it with, a float or a string, the *PriceValue* instance will always be a string value.

The types also validate the values passed. Invalid values will raise an exception.

5.1 AccountID

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.

```
__init__(accountID)
```

```
value  
value property.
```

5.2 AccountUnits

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

```
__init__(units)
```

```
value  
value property.
```

5.3 ClientComment

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

```
__init__(clientComment)
```

```
value  
value property.
```

5.4 ClientID

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

```
__init__(clientID)
```

```
value  
value property.
```

5.5 ClientTag

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

```
__init__(clientTag)
```

```
value  
value property.
```

5.6 DateTime

```
class oandapyV20.types.DateTime(dateTime)  
representation of a DateTime as a RFC 3339 string.
```

Parameters

- **dateTime** (*string, datetime instance, dict (required)*) –
the **dateTime** parameter must be:
 - a valid RFC3339 string representing a date-time, or
 - a dict holding the relevant datetime parts, or
 - a `datetime.datetime` instance
- **value property is always RFC3339 datetime string** (*The*) –
- **seconds are in microseconds. This compatible with** (*Fractional*) –
- **datetime.datetime.** –

Example

```
>>> print DateTime("2014-07-02T04:00:00.000000Z").value
>>> print DateTime({"year": 2014, "month": 12, "day": 2,
...                 "hour": 13, "minute": 48, "second": 12}).value
>>> from datetime import datetime
>>> print DateTime(datetime.now()).value
```

A `ValueError` exception is raised in case of an invalid value

`__init__` (*dateTime*)

value
value property.

5.7 OrderID

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

`__init__` (*orderID*)

value
value property.

5.8 OrderIdentifier

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

```
__init__(orderID, clientID)
```

```
value  
    value property.
```

5.9 OrderSpecifier

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

```
__init__(specifier)
```

```
value  
    value property.
```

5.10 PriceValue

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

```
__init__(priceValue)
```

```
value  
    value property.
```

5.11 TradeID

```
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

```
__init__(tradeID)
```

```
value  
    value property.
```

5.12 Units

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

```
__init__(units)
```

value
value property.

6.1 Factories

The `oandapyV20.contrib.factories` module contains several classes / methods that can be used optionally to generate requests.

6.1.1 InstrumentsCandlesFactory

`oandapyV20.contrib.factories.InstrumentsCandlesFactory` (*instrument*, *params=None*)
InstrumentsCandlesFactory - generate InstrumentCandles requests.

InstrumentsCandlesFactory is used to retrieve historical data by automatically generating consecutive requests when the OANDA limit of *count* records is exceeded.

This is known by calculating the number of candles between *from* and *to*. If *to* is not specified *to* will be equal to *now*.

The *count* parameter is only used to control the number of records to retrieve in a single request.

The *includeFirst* parameter is forced to make sure that results do not have a 1-record gap between consecutive requests.

Parameters

- **instrument** (*string (required)*) – the instrument to create the order for
- **params** (*params (optional)*) – the parameters to specify the historical range, see the REST-V20 docs regarding ‘instrument’ at developer.oanda.com If no params are specified, just a single InstrumentsCandles request will be generated acting the same as if you had just created it directly.

Example

The `oandapyV20.API` client processes requests as objects. So, downloading large historical batches simply comes down to:

```
>>> import json
>>> from oandapyV20 import API
>>> from oandapyV20.contrib.factories import InstrumentsCandlesFactory
>>>
>>> client = API(access_token=...)
>>> instrument, granularity = "EUR_USD", "M15"
>>> _from = "2017-01-01T00:00:00Z"
>>> params = {
...     "from": _from,
...     "granularity": granularity
...     "count": 2500,
... }
>>> with open("/tmp/{}.{}".format(instrument, granularity), "w") as OUT:
>>>     # The factory returns a generator generating consecutive
>>>     # requests to retrieve full history from date 'from' till 'to'
>>>     for r in InstrumentsCandlesFactory(instrument=instrument,
...                                     params=params)
>>>         client.request(r)
>>>         OUT.write(json.dumps(r.response.get('candles'), indent=2))
```

Note: Normally you can't combine *from*, *to* and *count*. When *count* specified, it is used to calculate the gap between *to* and *from*. The *params* passed to the generated request itself does contain the *count* parameter.

6.2 Generic

The `oandapyV20.contrib.generic` module contains several classes / methods that serve a generic purpose.

6.2.1 granularity_to_time

`oandapyV20.contrib.generic.granularity_to_time(s)`
convert a named granularity into seconds.

get value in seconds for named granularities: M1, M5 ... H1 etc.

```
>>> print(granularity_to_time("M5"))
300
```

`oandapyV20.contrib.generic.secs2time(e)`
secs2time - convert epoch to datetime.

```
>>> d = secs2time(1497499200)
>>> d
datetime.datetime(2017, 6, 15, 4, 0)
>>> d.strftime("%Y%m%d-%H:%M:%S")
'20170615-04:00:00'
```


6.3 Order Classes

The `oandapyV20.contrib.requests` module contains several classes that can be used optionally 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*.

These order specifications and additional data that goes with these order specifications can be created by the `contrib.requests.*Order*` classes and the `contrib.requests.*Details` classes.

6.3.1 LimitOrderRequest

```
class oandapyV20.contrib.requests.LimitOrderRequest(instrument, units, price, positionFill='DEFAULT', clientExtensions=None, takeProfitOnFill=None, timeInForce='GTC', gtdTime=None, 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', gtdTime=None, 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=...)
>>> ordr = LimitOrderRequest(instrument="EUR_USD",
...                          units=10000, price=1.08)
>>> print(json.dumps(ordr.data, indent=4))
{
  "order": {
    "timeInForce": "GTC",
```

```

        "instrument": "EUR_USD",
        "units": "10000",
        "price": "1.08000",
        "type": "LIMIT",
        "positionFill": "DEFAULT"
    }
}
>>> r = orders.orderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>>

```

data

data property.

return the JSON order body

6.3.2 MarketOrderRequest

```

class oandapyV20.contrib.requests.MarketOrderRequest (instrument, units, price-
    Bound=None, position-
    Fill='DEFAULT', client-
    Extensions=None, take-
    ProfitOnFill=None, timeIn-
    Force='FOK', stopLossOn-
    Fill=None, trailingStopLos-
    sOnFill=None, tradeClientEx-
    tensions=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 Order-Create 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.

6.3.3 MITOrderRequest

```
class oandapyV20.contrib.requests.MITOrderRequest (instrument, units, price,
                                                    priceBound=None, position-
                                                    Fill='DEFAULT', timeIn-
                                                    Force='GTC', gtdTime=None,
                                                    clientExtensions=None, take-
                                                    ProfitOnFill=None, stopLossOn-
                                                    Fill=None, trailingStopLossOn-
                                                    Fill=None, tradeClientExten-
                                                    sions=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=...)
>>> ordr = MITOrderRequest(instrument="EUR_USD",
...                        units=10000, price=1.08)
>>> print(json.dumps(ordr.data, indent=4))
{
  "order": {
    "timeInForce": "GTC",
    "instrument": "EUR_USD",
    "units": "10000",
    "price": "1.08000",
    "type": "MARKET_IF_TOUCHED",
    "positionFill": "DEFAULT"
  }
}
```

```
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

data

data property.

return the JSON order body

6.3.4 PositionCloseRequest

```
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

One of the parameters or both must be supplied.

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)
>>> ...
```

6.3.5 StopLossOrderRequest

```
class oandapyV20.contrib.requests.StopLossOrderRequest (tradeID, price, client-
    TradeID=None, time-
    InForce='GTC', gtd-
    Time=None, clientExten-
    sions=None)
```

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a StopLossOrderRequest.

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

```
__init__ (tradeID, price, clientTradeID=None, timeInForce='GTC', gtdTime=None, clientExten-
    sions=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(json.dumps(rv, indent=4))
>>> ...
```

data
 data property.
 return the JSON body.

6.3.6 StopOrderRequest

```
class oandapyV20.contrib.requests.StopOrderRequest (instrument, units, price,
                                                    priceBound=None, position-
                                                    Fill='DEFAULT', timeIn-
                                                    Force='GTC', gtdTime=None,
                                                    clientExtensions=None, take-
                                                    ProfitOnFill=None, stopLossOn-
                                                    Fill=None, trailingStopLossOn-
                                                    Fill=None, tradeClientExten-
                                                    sions=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(json.dumps(rv, indent=4))
>>> ...

```

data

data property.

return the JSON body.

6.3.7 TakeProfitOrderRequest

```

class oandapyV20.contrib.requests.TakeProfitOrderRequest (tradeID, price, client-
    TradeID=None, time-
    InForce='GTC', gtd-
    Time=None, clientExten-
    sions=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, clientExten-
    sions=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))
{
  "order": {
    "timeInForce": "GTC",
    "price": "1.22000",
    "type": "TAKE_PROFIT",
    "tradeID": "1234"
  }
}

```



```

    }
  }
  >>> r = orders.OrderCreate(accountID, data=ordr.data)
  >>> rv = client.request(r)
  >>> ...

```

data

data property.

return the JSON order body

6.3.8 TradeCloseRequest

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)
>>> ...

```

6.3.9 TrailingStopLossOrderRequest

```
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(json.dumps(rv, indent=4))
>>> ...
```

data

data property.

return the JSON body.

6.4 support classes

The `oandapyV20.contrib.requests` module contains several classes that can be used optionally 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*.

These order specifications and additional data that goes with these order specifications can be created by the `contrib.requests.*Order*` classes and the `contrib.requests.*Details` classes.

6.4.1 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)
>>> ...

```

6.4.2 StopLossDetails

```

class oandapyV20.contrib.requests.StopLossDetails(price,           timeInForce='GTC',
                                                  gtdTime=None,      clientExten-
                                                  sions=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 StopLossDetails.

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"
}

```

```

>>> order = MarketOrderRequest (
>>>     instrument="EUR_USD",
>>>     units=10000,
>>>     stopLossOnFill=stopLossOnFill.data
>>> )
>>> # or as shortcut ...
>>> # stopLossOnFill=StopLossDetails(price=1.06).data
>>> print(json.dumps(order.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=order.data)
>>> rv = client.request(r)
>>> ...

```

6.4.3 TakeProfitDetails

```

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.data)
{
  "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))
{
  "order": {
    "timeInForce": "FOK",
    "instrument": "EUR_USD",
    "units": "10000",
    "positionFill": "DEFAULT",
    "type": "MARKET",
    "takeProfitOnFill": {
      "timeInForce": "GTC",
      "price": "1.10000"
    }
  }
}
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...

```

6.4.4 TrailingStopLossDetails

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

Bases: oandapyV20.contrib.requests.onfill.OnFill

Representation of the specification for a TrailingStopLossOrder.

It is typically used to specify 'trailing 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 TrailingStopLossOrder 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 TrailingStopLossDetails.

Parameters

- **distance** (*float or string (required)*) – the price to trigger trailing stop loss 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)
>>> ...
```


Examples can be found in the examples repository on github: [examplesrepo](#).

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 cleft <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-yyyYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY"
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

`oandapyV20.contrib.factories`, 107
`oandapyV20.contrib.generic`, 108
`oandapyV20.definitions.accounts`, 79
`oandapyV20.definitions.instruments`, 80
`oandapyV20.definitions.orders`, 82
`oandapyV20.definitions.pricing`, 86
`oandapyV20.definitions.trades`, 87
`oandapyV20.definitions.transactions`, 89

Symbols

`__getitem__()` (oandapyV20.definitions.accounts.AccountFinancingMode method), 79
`__getitem__()` (oandapyV20.definitions.accounts.PositionAggregationMode method), 80
`__getitem__()` (oandapyV20.definitions.instruments.CandlestickGranularity method), 81
`__getitem__()` (oandapyV20.definitions.instruments.PriceComponents method), 82
`__getitem__()` (oandapyV20.definitions.instruments.WeeklyAlignment method), 82
`__getitem__()` (oandapyV20.definitions.orders.CancellableOrderType method), 84
`__getitem__()` (oandapyV20.definitions.orders.OrderPositionFill method), 85
`__getitem__()` (oandapyV20.definitions.orders.OrderState method), 86
`__getitem__()` (oandapyV20.definitions.orders.OrderStateFilter method), 83
`__getitem__()` (oandapyV20.definitions.orders.OrderTriggerCondition method), 86
`__getitem__()` (oandapyV20.definitions.orders.OrderType method), 83
`__getitem__()` (oandapyV20.definitions.orders.TimeInForce method), 85
`__getitem__()` (oandapyV20.definitions.pricing.PriceStatus method), 87
`__getitem__()` (oandapyV20.definitions.trades.TradePL method), 87
`__getitem__()` (oandapyV20.definitions.trades.TradeState method), 88
`__getitem__()` (oandapyV20.definitions.trades.TradeStateFilter method), 88
`__getitem__()` (oandapyV20.definitions.transactions.FundingReason method), 91
`__getitem__()` (oandapyV20.definitions.transactions.LimitOrderReason method), 100
`__getitem__()` (oandapyV20.definitions.transactions.MarketIfTouchedOrderReason method), 91
`__getitem__()` (oandapyV20.definitions.transactions.MarketOrderMarginClose method), 89
`__getitem__()` (oandapyV20.definitions.transactions.MarketOrderReason method), 92
`__getitem__()` (oandapyV20.definitions.transactions.OrderCancelReason method), 99
`__getitem__()` (oandapyV20.definitions.transactions.OrderFillReason method), 90
`__getitem__()` (oandapyV20.definitions.transactions.StopLossOrderReason method), 90
`__getitem__()` (oandapyV20.definitions.transactions.StopOrderReason method), 92
`__getitem__()` (oandapyV20.definitions.transactions.TakeProfitOrderReason method), 94
`__getitem__()` (oandapyV20.definitions.transactions.TrailingStopLossOrderReason method), 100
`__getitem__()` (oandapyV20.definitions.transactions.TransactionRejectReason method), 98
`__getitem__()` (oandapyV20.definitions.transactions.TransactionType method), 94
`__init__()` (oandapyV20.API method), 7
`__init__()` (oandapyV20.V20Error method), 8
`__init__()` (oandapyV20.contrib.requests.ClientExtensions method), 119
`__init__()` (oandapyV20.contrib.requests.LimitOrderRequest method), 109
`__init__()` (oandapyV20.contrib.requests.MITOrderRequest method), 112
`__init__()` (oandapyV20.contrib.requests.MarketOrderRequest method), 110
`__init__()` (oandapyV20.contrib.requests.PositionCloseRequest method), 113
`__init__()` (oandapyV20.contrib.requests.StopLossDetails method), 120
`__init__()` (oandapyV20.contrib.requests.StopLossOrderRequest method), 114
`__init__()` (oandapyV20.contrib.requests.StopOrderRequest method), 115
`__init__()` (oandapyV20.contrib.requests.TakeProfitDetails method), 121

<code>__init__()</code> (oandapyV20.contrib.requests.TakeProfitOrderRequest method), 116	<code>__init__()</code> (oandapyV20.endpoints.positions.PositionClose method), 52
<code>__init__()</code> (oandapyV20.contrib.requests.TradeCloseRequest method), 117	<code>__init__()</code> (oandapyV20.endpoints.positions.PositionDetails method), 53
<code>__init__()</code> (oandapyV20.contrib.requests.TrailingStopLossDetails method), 122	<code>__init__()</code> (oandapyV20.endpoints.positions.PositionList method), 54
<code>__init__()</code> (oandapyV20.contrib.requests.TrailingStopLossOrderRequest method), 118	<code>__init__()</code> (oandapyV20.endpoints.pricing.PricingInfo method), 56
<code>__init__()</code> (oandapyV20.endpoints.accounts.AccountChanges method), 11	<code>__init__()</code> (oandapyV20.endpoints.pricing.PricingStream method), 59
<code>__init__()</code> (oandapyV20.endpoints.accounts.AccountConfiguration method), 13	<code>__init__()</code> (oandapyV20.endpoints.trades.OpenTrades method), 62
<code>__init__()</code> (oandapyV20.endpoints.accounts.AccountDetails method), 14	<code>__init__()</code> (oandapyV20.endpoints.trades.TradeCRCDO method), 63
<code>__init__()</code> (oandapyV20.endpoints.accounts.AccountInstrument method), 17	<code>__init__()</code> (oandapyV20.endpoints.trades.TradeClientExtensions method), 65
<code>__init__()</code> (oandapyV20.endpoints.accounts.AccountList method), 19	<code>__init__()</code> (oandapyV20.endpoints.trades.TradeClose method), 66
<code>__init__()</code> (oandapyV20.endpoints.accounts.AccountSummary method), 19	<code>__init__()</code> (oandapyV20.endpoints.trades.TradeDetails method), 68
<code>__init__()</code> (oandapyV20.endpoints.forexlabs.Autochartist method), 21	<code>__init__()</code> (oandapyV20.endpoints.trades.TradesList method), 68
<code>__init__()</code> (oandapyV20.endpoints.forexlabs.Calendar method), 22	<code>__init__()</code> (oandapyV20.endpoints.transactions.TransactionDetails method), 70
<code>__init__()</code> (oandapyV20.endpoints.forexlabs.CommitmentsOfTraders method), 23	<code>__init__()</code> (oandapyV20.endpoints.transactions.TransactionIDRange method), 71
<code>__init__()</code> (oandapyV20.endpoints.forexlabs.HistoricalPositionRatio method), 25	<code>__init__()</code> (oandapyV20.endpoints.transactions.TransactionList method), 72
<code>__init__()</code> (oandapyV20.endpoints.forexlabs.OrderbookData method), 32	<code>__init__()</code> (oandapyV20.endpoints.transactions.TransactionsSinceID method), 74
<code>__init__()</code> (oandapyV20.endpoints.forexlabs.Spreads method), 33	<code>__init__()</code> (oandapyV20.endpoints.transactions.TransactionsStream method), 76
<code>__init__()</code> (oandapyV20.endpoints.instruments.InstrumentsCandles method), 34	<code>__init__()</code> (oandapyV20.types.AccountID method), 101
<code>__init__()</code> (oandapyV20.endpoints.instruments.InstrumentsOrderBook method), 36	<code>__init__()</code> (oandapyV20.types.AccountUnits method), 102
<code>__init__()</code> (oandapyV20.endpoints.instruments.InstrumentsPositionBook method), 39	<code>__init__()</code> (oandapyV20.types.ClientComment method), 103
<code>__init__()</code> (oandapyV20.endpoints.orders.OrderCancel method), 43	<code>__init__()</code> (oandapyV20.types.ClientID method), 102
<code>__init__()</code> (oandapyV20.endpoints.orders.OrderClientExtensions method), 43	<code>__init__()</code> (oandapyV20.types.ClientTag method), 102
<code>__init__()</code> (oandapyV20.endpoints.orders.OrderCreate method), 44	<code>__init__()</code> (oandapyV20.types.DateTime method), 103
<code>__init__()</code> (oandapyV20.endpoints.orders.OrderDetails method), 46	<code>__init__()</code> (oandapyV20.types.OrderID method), 103
<code>__init__()</code> (oandapyV20.endpoints.orders.OrderList method), 47	<code>__init__()</code> (oandapyV20.types.OrderIdentifier method), 103
<code>__init__()</code> (oandapyV20.endpoints.orders.OrderReplace method), 48	<code>__init__()</code> (oandapyV20.types.OrderSpecifier method), 104
<code>__init__()</code> (oandapyV20.endpoints.orders.OrdersPending method), 49	<code>__init__()</code> (oandapyV20.types.PriceValue method), 104
<code>__init__()</code> (oandapyV20.endpoints.positions.OpenPositions method), 50	<code>__init__()</code> (oandapyV20.types.TradeID method), 104
	<code>__init__()</code> (oandapyV20.types.Units method), 104
	A
	A (oandapyV20.definitions.instruments.PriceComponents attribute), 82
	ABSOLUTE_SUM (oandapyV20.definitions.accounts.PositionAggregationMode attribute), 82

attribute), 80

ACCOUNT_CONFIGURATION_LOCKED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ACCOUNT_DEPOSIT_LOCKED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ACCOUNT_LOCKED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 98

ACCOUNT_LOCKED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ACCOUNT_NEW_POSITIONS_LOCKED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 98

ACCOUNT_NOT_ACTIVE (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ACCOUNT_ORDER_CANCEL_LOCKED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ACCOUNT_ORDER_CREATION_LOCKED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 98

ACCOUNT_ORDER_CREATION_LOCKED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ACCOUNT_ORDER_FILL_LOCKED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 98

ACCOUNT_TRANSFER (oandapyV20.definitions.transactions.FundingReason attribute), 91

ACCOUNT_WITHDRAWAL_LOCKED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

AccountChanges (class in oandapyV20.endpoints.accounts), 11

AccountConfiguration (class in oandapyV20.endpoints.accounts), 13

AccountDetails (class in oandapyV20.endpoints.accounts), 14

AccountFinancingMode (class in oandapyV20.definitions.accounts), 79

AccountID (class in oandapyV20.types), 101

AccountInstruments (class in oandapyV20.endpoints.accounts), 16

AccountList (class in oandapyV20.endpoints.accounts), 19

AccountSummary (class in oandapyV20.endpoints.accounts), 19

AccountUnits (class in oandapyV20.types), 102

ADJUSTMENT (oandapyV20.definitions.transactions.FundingReason attribute), 91

ADMIN_CONFIGURE_DATA_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ALIAS_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ALL (oandapyV20.definitions.orders.OrderStateFilter attribute), 83

ALL (oandapyV20.definitions.trades.TradeStateFilter attribute), 88

AMOUNT_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

AMOUNT_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

ANSI_QUOTES (class in oandapyV20), 5

ASK (oandapyV20.definitions.orders.OrderTriggerCondition attribute), 86

Artist (class in oandapyV20.endpoints.forexlabs), 20

B

B (oandapyV20.definitions.instruments.PriceComponents attribute), 82

BROKER (oandapyV20.definitions.orders.OrderTriggerCondition attribute), 86

BOUNDS_VIOLATION

(oandapyV20.definitions.transactions.OrderCancelReason attribute), 98

C

Calendar (class in oandapyV20.endpoints.forexlabs), 22

CancellableOrderType (class in oandapyV20.definitions.orders), 84

CANCELLED (oandapyV20.definitions.orders.OrderState attribute), 85

CANCELLED (oandapyV20.definitions.orders.OrderStateFilter attribute), 83

CandlestickGranularity (class in oandapyV20.definitions.instruments), 80

CLIENT_CONFIGURE (oandapyV20.definitions.transactions.TransactionType attribute), 93

CLIENT_CONFIGURE_DATA_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

CLIENT_CONFIGURE_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 93

CLIENT_EXTENSIONS_DATA_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

CLIENT_FUNDING	(oandapyV20.definitions.transactions.FundingReason attribute), 91	CLIENT_TRADE_TAG_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_ORDER	(oandapyV20.definitions.transactions.LimitOrderReason attribute), 100	ClientComment	(class in oandapyV20.types), 102
CLIENT_ORDER	(oandapyV20.definitions.transactions.MarketIfTouchedOrderReason attribute), 91	ClientExtensions	(class in oandapyV20.contrib.requests), 119
CLIENT_ORDER	(oandapyV20.definitions.transactions.MarketOrderReason attribute), 92	ClientID	(class in oandapyV20.types), 102
CLIENT_ORDER	(oandapyV20.definitions.transactions.StopLossOrderReason attribute), 89	CloseIfTagged	(class in oandapyV20.types), 102
CLIENT_ORDER	(oandapyV20.definitions.transactions.StopOrderReason attribute), 92	CLOSE	(oandapyV20.definitions.transactions.TransactionType attribute), 93
CLIENT_ORDER	(oandapyV20.definitions.transactions.TakeProfitOrderReason attribute), 94	CLOSE_TRADE_PARTIAL_UNITS_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_ORDER	(oandapyV20.definitions.transactions.TrailingStopLossOrderReason attribute), 100	CLOSE_TRADE_TYPE_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_ORDER_COMMENT_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CLOSE_TRADE_UNITS_EXCEED_TRADE_SIZE	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_ORDER_ID_ALREADY_EXISTS	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CLOSE_WHEN_TRADABLE	(oandapyV20.definitions.trades.TradeState attribute), 88
CLIENT_ORDER_ID_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CLOSE_WHEN_TRADEABLE	(oandapyV20.definitions.trades.TradeStateFilter attribute), 88
CLIENT_ORDER_TAG_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CLOSED	(oandapyV20.definitions.trades.TradeStateFilter attribute), 88
CLIENT_REQUEST	(oandapyV20.definitions.transactions.OrderCancelReason attribute), 99	CLOSEOUT_POSITION_DOESNT_EXIST	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_REQUEST_REPLACED	(oandapyV20.definitions.transactions.OrderCancelReason attribute), 99	CLOSEOUT_POSITION_INCOMPLETE_SPECIFICATION	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_TRADE_COMMENT_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CLOSEOUT_POSITION_PARTIAL_UNITS_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_TRADE_ID_ALREADY_EXISTS	(oandapyV20.definitions.transactions.OrderCancelReason attribute), 99	CLOSEOUT_POSITION_REJECT	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_TRADE_ID_ALREADY_EXISTS	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CLOSEOUT_POSITION_UNITS_EXCEED_POSITION_SIZE	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95
CLIENT_TRADE_ID_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95	CommitmentsOfTraders	(class in oandapyV20.endpoints.forexlabs), 23
		CREATE	(oandapyV20.definitions.transactions.TransactionType attribute), 93
		D	
		D	(oandapyV20.definitions.instruments.CandlestickGranularity attribute), 80
		DAILY	(oandapyV20.definitions.accounts.AccountFinancingMode attribute), 79

DAILY_FINANCING (oandapyV20.definitions.transactions.TransactionType attribute), 93

data (oandapyV20.contrib.requests.LimitOrderRequest attribute), 110

data (oandapyV20.contrib.requests.MarketOrderRequest attribute), 111

data (oandapyV20.contrib.requests.MITOrderRequest attribute), 113

data (oandapyV20.contrib.requests.StopLossOrderRequest attribute), 114

data (oandapyV20.contrib.requests.StopOrderRequest attribute), 116

data (oandapyV20.contrib.requests.TakeProfitOrderRequest attribute), 117

data (oandapyV20.contrib.requests.TrailingStopLossOrderRequest attribute), 118

DateTime (class in oandapyV20.types), 102

DEFAULT (oandapyV20.definitions.orders.OrderPositionFill attribute), 84

DEFAULT (oandapyV20.definitions.orders.OrderTriggerCondition attribute), 86

definitions (oandapyV20.definitions.accounts.AccountFinancing attribute), 80

definitions (oandapyV20.definitions.accounts.PositionAggregation attribute), 80

definitions (oandapyV20.definitions.instruments.CandlestickOHLC attribute), 81

definitions (oandapyV20.definitions.instruments.PriceComponents attribute), 82

definitions (oandapyV20.definitions.instruments.WeeklyAlignment attribute), 82

definitions (oandapyV20.definitions.orders.CancellableOrderType attribute), 84

definitions (oandapyV20.definitions.orders.OrderPositionFill attribute), 85

definitions (oandapyV20.definitions.orders.OrderState attribute), 86

definitions (oandapyV20.definitions.orders.OrderStateFilter attribute), 83

definitions (oandapyV20.definitions.orders.OrderTriggerCondition attribute), 86

definitions (oandapyV20.definitions.orders.OrderType attribute), 83

definitions (oandapyV20.definitions.orders.TimeInForce attribute), 85

definitions (oandapyV20.definitions.pricing.PriceStatus attribute), 87

definitions (oandapyV20.definitions.trades.TradePL attribute), 87

definitions (oandapyV20.definitions.trades.TradeState attribute), 88

definitions (oandapyV20.definitions.trades.TradeStateFilter attribute), 88

definitions (oandapyV20.definitions.transactions.FundingReason attribute), 91

definitions (oandapyV20.definitions.transactions.LimitOrderReason attribute), 100

definitions (oandapyV20.definitions.transactions.MarketIfTouchedOrderReason attribute), 91

definitions (oandapyV20.definitions.transactions.MarketOrderMarginClose attribute), 89

definitions (oandapyV20.definitions.transactions.MarketOrderReason attribute), 92

definitions (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

definitions (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

definitions (oandapyV20.definitions.transactions.StopLossOrderReason attribute), 90

definitions (oandapyV20.definitions.transactions.StopOrderReason attribute), 92

definitions (oandapyV20.definitions.transactions.TakeProfitOrderReason attribute), 94

definitions (oandapyV20.definitions.transactions.TrailingStopLossOrderReason attribute), 100

definitions (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98

definitions (oandapyV20.definitions.transactions.TransactionType attribute), 94

DEFINITION_TRADE_CLOSE (oandapyV20.definitions.transactions.MarketOrderReason attribute), 92

DEFINITION_TRADE_CLOSURE (oandapyV20.definitions.transactions.TransactionType attribute), 93

DEFINITION_MIGRATION (oandapyV20.definitions.transactions.FundingReason attribute), 91

E

ENDPOINT (oandapyV20.endpoints.accounts.AccountChanges attribute), 11

ENDPOINT (oandapyV20.endpoints.accounts.AccountConfiguration attribute), 13

ENDPOINT (oandapyV20.endpoints.accounts.AccountDetails attribute), 14

ENDPOINT (oandapyV20.endpoints.accounts.AccountInstruments attribute), 17

ENDPOINT (oandapyV20.endpoints.accounts.AccountList attribute), 19

ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19

ENDPOINT (oandapyV20.endpoints.forexlabs.Autochartist attribute), 20

ENDPOINT (oandapyV20.endpoints.forexlabs.Calendar attribute), 22

ENDPOINT (oandapyV20.endpoints.forexlabs.Commitments attribute), 23	ENDPOINT (oandapyV20.endpoints.transactions.TransactionIDRange attribute), 71
ENDPOINT (oandapyV20.endpoints.forexlabs.HistoricalPositions attribute), 24	ENDPOINT (oandapyV20.endpoints.transactions.TransactionList attribute), 72
ENDPOINT (oandapyV20.endpoints.forexlabs.OrderbookData attribute), 31	ENDPOINT (oandapyV20.endpoints.transactions.TransactionsSinceID attribute), 74
ENDPOINT (oandapyV20.endpoints.forexlabs.Spreads attribute), 33	ENDPOINT (oandapyV20.endpoints.transactions.TransactionsStream attribute), 76
ENDPOINT (oandapyV20.endpoints.instruments.InstrumentsExpectedStatus attribute), 34	ENDPOINT (oandapyV20.endpoints.accounts.AccountChanges attribute), 11
ENDPOINT (oandapyV20.endpoints.instruments.InstrumentsOrderBook attribute), 36	ENDPOINT (oandapyV20.endpoints.accounts.AccountConfiguration attribute), 13
ENDPOINT (oandapyV20.endpoints.instruments.InstrumentsPositionBook attribute), 39	ENDPOINT (oandapyV20.endpoints.accounts.AccountDetails attribute), 14
ENDPOINT (oandapyV20.endpoints.orders.OrderCancel attribute), 42	ENDPOINT (oandapyV20.endpoints.accounts.AccountInstruments attribute), 17
ENDPOINT (oandapyV20.endpoints.orders.OrderClientExtensions attribute), 43	ENDPOINT (oandapyV20.endpoints.accounts.AccountList attribute), 19
ENDPOINT (oandapyV20.endpoints.orders.OrderCreate attribute), 44	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.orders.OrderDetails attribute), 46	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.orders.OrderList attribute), 46	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.orders.OrderReplace attribute), 47	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.orders.OrdersPending attribute), 49	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.positions.OpenPositions attribute), 50	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.positions.PositionClose attribute), 51	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.positions.PositionDetails attribute), 53	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.positions.PositionList attribute), 54	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.pricing.PricingInfo attribute), 56	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.pricing.PricingStream attribute), 59	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.trades.OpenTrades attribute), 62	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.trades.TradeClientExtensions attribute), 65	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.trades.TradeClose attribute), 66	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.trades.TradeCRCDO attribute), 63	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.trades.TradeDetails attribute), 68	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.trades.TradesList attribute), 68	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19
ENDPOINT (oandapyV20.endpoints.transactions.TransactionDetails attribute), 70	ENDPOINT (oandapyV20.endpoints.accounts.AccountSummary attribute), 19

dapyV20.endpoints.orders.OrderCancel tribute), 42	at-	dapyV20.endpoints.trades.TradesList tribute), 68	at-
EXPECTED_STATUS dapyV20.endpoints.orders.OrderClientExtensions tribute), 43	(oan-	EXPECTED_STATUS dapyV20.endpoints.transactions.TransactionDetails tribute), 70	(oan-
EXPECTED_STATUS dapyV20.endpoints.orders.OrderCreate tribute), 44	(oan- at-	EXPECTED_STATUS dapyV20.endpoints.transactions.TransactionIDRange tribute), 71	(oan-
EXPECTED_STATUS dapyV20.endpoints.orders.OrderDetails tribute), 46	(oan- at-	EXPECTED_STATUS dapyV20.endpoints.transactions.TransactionList tribute), 72	(oan-
EXPECTED_STATUS dapyV20.endpoints.orders.OrderList tribute), 46	(oan- attribute),	EXPECTED_STATUS dapyV20.endpoints.transactions.TransactionsSinceID tribute), 74	(oan-
EXPECTED_STATUS dapyV20.endpoints.orders.OrderReplace tribute), 47	(oan-	EXPECTED_STATUS dapyV20.endpoints.transactions.TransactionsStream tribute), 76	(oan-
EXPECTED_STATUS dapyV20.endpoints.orders.OrdersPending tribute), 49	(oan-	F	
EXPECTED_STATUS dapyV20.endpoints.positions.OpenPositions tribute), 50	(oan-	FIFO_VIOLATION dapyV20.definitions.transactions.OrderCancelReason tribute), 99	(oan-
EXPECTED_STATUS dapyV20.endpoints.positions.PositionClose tribute), 52	(oan-	FILLED (oandapyV20.definitions.orders.OrderState tribute), 85	at-
EXPECTED_STATUS dapyV20.endpoints.positions.PositionDetails tribute), 53	(oan-	FILLED (oandapyV20.definitions.orders.OrderStateFilter tribute), 83	at-
EXPECTED_STATUS dapyV20.endpoints.positions.PositionList tribute), 54	(oan-	FOK (oandapyV20.definitions.orders.TimeInForce tribute), 85	at-
EXPECTED_STATUS dapyV20.endpoints.pricing.PricingInfo tribute), 56	(oan- at-	Friday (oandapyV20.definitions.instruments.WeeklyAlignment tribute), 81	at-
EXPECTED_STATUS dapyV20.endpoints.pricing.PricingStream tribute), 59	(oan-	FUNDING_REASON_MISSING dapyV20.definitions.transactions.TransactionRejectReason tribute), 95	(oan-
EXPECTED_STATUS dapyV20.endpoints.trades.OpenTrades tribute), 62	(oan- at-	FundingReason (class in oandapyV20.definitions.transactions), 90	oan-
EXPECTED_STATUS dapyV20.endpoints.trades.TradeClientExtensions tribute), 65	(oan- at-	G	
EXPECTED_STATUS dapyV20.endpoints.trades.TradeClose tribute), 66	(oan- at-	GFD (oandapyV20.definitions.orders.TimeInForce tribute), 85	at-
EXPECTED_STATUS dapyV20.endpoints.trades.TradeCRCDO tribute), 63	(oan-	granularity_to_time() (in module oandapyV20.contrib.generic), 108	oan-
EXPECTED_STATUS dapyV20.endpoints.trades.TradeDetails tribute), 68	(oan- at-	GTC (oandapyV20.definitions.orders.TimeInForce tribute), 85	at-
EXPECTED_STATUS	(oan-	GTD (oandapyV20.definitions.orders.TimeInForce tribute), 85	at-
H			
	(oan-	H1 (oandapyV20.definitions.instruments.CandlestickGranularity tribute), 80	at-
	(oan- at-	H12 (oandapyV20.definitions.instruments.CandlestickGranularity tribute), 81	at-
	(oan-	H2 (oandapyV20.definitions.instruments.CandlestickGranularity tribute), 81	at-

H3 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

H4 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

H6 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

H8 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

HEADERS (oandapyV20.endpoints.accounts.AccountConfiguration attribute), 13

HEADERS (oandapyV20.endpoints.orders.OrderClientExtensions attribute), 43

HEADERS (oandapyV20.endpoints.orders.OrderCreate attribute), 44

HEADERS (oandapyV20.endpoints.orders.OrderReplace attribute), 47

HEADERS (oandapyV20.endpoints.positions.PositionCloseIOC attribute), 52

HEADERS (oandapyV20.endpoints.trades.TradeClientExtensions attribute), 65

HEADERS (oandapyV20.endpoints.trades.TradeClose attribute), 66

HEADERS (oandapyV20.endpoints.trades.TradeCRCD attribute), 63

HistoricalPositionRatios (class in oandapyV20.endpoints.forexlabs), 24

I

INSTRUMENT_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

INSTRUMENT_NOT_TRADEABLE (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 95

INSTRUMENT_PRICE_UNKNOWN (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

INSTRUMENT_UNKNOWN (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

InstrumentsCandles (class in oandapyV20.endpoints.instruments), 34

InstrumentsCandlesFactory() (in module oandapyV20.contrib.factories), 107

InstrumentsOrderBook (class in oandapyV20.endpoints.instruments), 36

InstrumentsPositionBook (class in oandapyV20.endpoints.instruments), 39

INSUFFICIENT_FUNDS (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

INSUFFICIENT_LIQUIDITY (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

INSUFFICIENT_MARGIN (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

INTERNAL_SERVER_ERROR (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

INTERNAL_SERVER_ERROR (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

INVALID (oandapyV20.definitions.pricing.PriceStatus attribute), 87

INVALID_REISSUE_IMMEDIATE_PARTIAL_FILL (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

INVERSE (oandapyV20.definitions.orders.OrderTriggerCondition attribute), 86

(oandapyV20.definitions.orders.TimeInForce attribute), 85

L

LIMIT (oandapyV20.definitions.orders.CancellableOrderType attribute), 84

LIMIT (oandapyV20.definitions.orders.OrderType attribute), 83

LIMIT_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

LIMIT_ORDER (oandapyV20.definitions.transactions.TransactionType attribute), 93

LIMIT_ORDER_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 93

LimitOrderReason (class in oandapyV20.definitions.transactions), 100

LimitOrderRequest (class in oandapyV20.contrib.requests), 109

LINKED_TRADE_CLOSED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

LOSING_TAKE_PROFIT (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

M

M (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

M (oandapyV20.definitions.instruments.PriceComponents attribute), 82

M1 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

M15 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

M2 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 90
 attribute), 81 MARKET_ORDER (oandapyV20.definitions.transactions.TransactionType attribute), 93

M30 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81

M4 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81 MARKET_ORDER_DELAYED_TRADE_CLOSE (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

M5 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81 MARKET_ORDER_MARGIN_CLOSEOUT (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

MARGIN_CALL_ENTER (oandapyV20.definitions.transactions.TransactionType attribute), 93 MARKET_ORDER_POSITION_CLOSEOUT (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

MARGIN_CALL_EXIT (oandapyV20.definitions.transactions.TransactionType attribute), 93 MARKET_ORDER_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 93

MARGIN_CALL_EXTEND (oandapyV20.definitions.transactions.TransactionType attribute), 93 MARKET_ORDER_TRADE_CLOSE (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

MARGIN_CHECK_VIOLATION (oandapyV20.definitions.transactions.MarketOrderMarginCloseoutReason attribute), 89 MarketIfTouchedOrderReason (class in oandapyV20.definitions.transactions), 91

MARGIN_CLOSEOUT (oandapyV20.definitions.transactions.MarketOrderMarginCloseoutReason attribute), 92 MarketOrderMarginCloseoutReason (class in oandapyV20.definitions.transactions), 89

MARGIN_RATE_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 MarketOrderReason (class in oandapyV20.definitions.transactions), 91

MARGIN_RATE_WOULD_TRIGGER_CLOSEOUT (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 MarketOrderRequest (class in oandapyV20.contrib.requests), 110

MARGIN_RATE_WOULD_TRIGGER_MARGIN_CALL (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 MARKET_GROUP_ID_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

MARKET (oandapyV20.definitions.orders.OrderType attribute), 83 MARKET_SIDE (oandapyV20.definitions.accounts.PositionAggregationMode attribute), 80

MARKET_HALTED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99 METHOD (oandapyV20.endpoints.accounts.AccountChanges attribute), 11

MARKET_IF_TOUCHED (oandapyV20.definitions.orders.CancellableOrderType attribute), 84 METHOD (oandapyV20.endpoints.accounts.AccountConfiguration attribute), 13

MARKET_IF_TOUCHED (oandapyV20.definitions.orders.OrderType attribute), 83 METHOD (oandapyV20.endpoints.accounts.AccountDetails attribute), 14

MARKET_IF_TOUCHED_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 90 METHOD (oandapyV20.endpoints.accounts.AccountInstruments attribute), 17

MARKET_IF_TOUCHED_ORDER (oandapyV20.definitions.transactions.TransactionType attribute), 93 METHOD (oandapyV20.endpoints.accounts.AccountList attribute), 19

MARKET_IF_TOUCHED_ORDER_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 90 METHOD (oandapyV20.endpoints.accounts.AccountSummary attribute), 19

MARKET_IF_TOUCHED_ORDER_ORDER (oandapyV20.definitions.transactions.TransactionType attribute), 93 METHOD (oandapyV20.endpoints.forexlabs.Autochartist attribute), 21

MARKET_IF_TOUCHED_ORDER_ORDER_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 93 METHOD (oandapyV20.endpoints.forexlabs.Calendar attribute), 22

MARKET_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 25 METHOD (oandapyV20.endpoints.forexlabs.CommitmentsOfTraders attribute), 23

MARKET_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 25 METHOD (oandapyV20.endpoints.forexlabs.HistoricalPositionRatios attribute), 25

MARKET_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 25 METHOD (oandapyV20.endpoints.forexlabs.OrderbookData attribute), 25

- attribute), 32
 - METHOD (oandapyV20.endpoints.forexlab.Spreads attribute), 33
 - METHOD (oandapyV20.endpoints.instruments.Instruments attribute), 34
 - METHOD (oandapyV20.endpoints.instruments.Instruments attribute), 36
 - METHOD (oandapyV20.endpoints.instruments.Instruments attribute), 39
 - METHOD (oandapyV20.endpoints.orders.OrderCancel attribute), 42
 - METHOD (oandapyV20.endpoints.orders.OrderClientExtensions attribute), 43
 - METHOD (oandapyV20.endpoints.orders.OrderCreate attribute), 44
 - METHOD (oandapyV20.endpoints.orders.OrderDetails attribute), 46
 - METHOD (oandapyV20.endpoints.orders.OrderList attribute), 47
 - METHOD (oandapyV20.endpoints.orders.OrderReplace attribute), 48
 - METHOD (oandapyV20.endpoints.orders.OrdersPending attribute), 49
 - METHOD (oandapyV20.endpoints.positions.OpenPositions attribute), 50
 - METHOD (oandapyV20.endpoints.positions.PositionClose attribute), 52
 - METHOD (oandapyV20.endpoints.positions.PositionDetails attribute), 53
 - METHOD (oandapyV20.endpoints.positions.PositionList attribute), 54
 - METHOD (oandapyV20.endpoints.pricing.PricingInfo attribute), 56
 - METHOD (oandapyV20.endpoints.pricing.PricingStream attribute), 59
 - METHOD (oandapyV20.endpoints.trades.OpenTrades attribute), 62
 - METHOD (oandapyV20.endpoints.trades.TradeClientExtensions attribute), 65
 - METHOD (oandapyV20.endpoints.trades.TradeClose attribute), 66
 - METHOD (oandapyV20.endpoints.trades.TradeCRDO attribute), 63
 - METHOD (oandapyV20.endpoints.trades.TradeDetails attribute), 68
 - METHOD (oandapyV20.endpoints.trades.TradesList attribute), 68
 - METHOD (oandapyV20.endpoints.transactions.TransactionDetails attribute), 70
 - METHOD (oandapyV20.endpoints.transactions.TransactionRange attribute), 71
 - METHOD (oandapyV20.endpoints.transactions.TransactionList attribute), 72
 - METHOD (oandapyV20.endpoints.transactions.TransactionsSinceID attribute), 74
 - METHOD (oandapyV20.endpoints.transactions.TransactionsStream attribute), 76
 - METHOD (oandapyV20.definitions.orders.OrderTriggerCondition attribute), 86
 - METHOD (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99
 - NotOnBreak (class in oandapyV20.contrib.requests), 112
 - Monday (oandapyV20.definitions.instruments.WeeklyAlignment attribute), 81
- ## N
- NEGATIVE (oandapyV20.definitions.trades.TradePL attribute), 87
 - NET_SUM (oandapyV20.definitions.accounts.PositionAggregationMode attribute), 80
 - NO_FINANCING (oandapyV20.definitions.accounts.AccountFinancingMode attribute), 79
 - non_tradeable (oandapyV20.definitions.pricing.PriceStatus attribute), 87
- ## O
- oandapyV20.contrib.factories (module), 107
 - oandapyV20.contrib.generic (module), 108
 - oandapyV20.definitions.accounts (module), 79
 - oandapyV20.definitions.instruments (module), 80
 - oandapyV20.definitions.orders (module), 82
 - oandapyV20.definitions.pricing (module), 86
 - oandapyV20.definitions.trades (module), 87
 - oandapyV20.definitions.transactions (module), 89
 - ON_FILL (oandapyV20.definitions.transactions.StopLossOrderReason attribute), 89
 - ON_FILL (oandapyV20.definitions.transactions.TakeProfitOrderReason attribute), 94
 - ON_FILL (oandapyV20.definitions.transactions.TrailingStopLossOrderReason attribute), 100
 - OPEN (oandapyV20.definitions.trades.TradeState attribute), 88
 - OPEN (oandapyV20.definitions.trades.TradeStateFilter attribute), 88
 - OPEN_ONLY (oandapyV20.definitions.orders.OrderPositionFill attribute), 84
 - OPEN_TRADES_ALLOWED_EXCEEDED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99
 - OpenPositions (class in oandapyV20.endpoints.positions), 50
 - OpenTrades (class in oandapyV20.endpoints.trades), 62
 - ORDER_CANCEL (oandapyV20.definitions.transactions.TransactionType attribute), 93

ORDER_CANCEL_REJECT (oandapyV20.definitions.transactions.TransactionTypeOrderState (class in oandapyV20.definitions.orders), 85 attribute), 93 OrderStateFilter (class in oandapyV20.definitions.orders), 82

ORDER_CLIENT_EXTENSIONS_MODIFY (oandapyV20.definitions.transactions.TransactionTypeOrderTriggerCondition (class in oandapyV20.definitions.orders), 86 attribute), 93

ORDER_CLIENT_EXTENSIONS_MODIFY_REJECT OrderType (class in oandapyV20.definitions.orders), 83 (oandapyV20.definitions.transactions.TransactionType attribute), 93

ORDER_DOESNT_EXIST (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 86 PENDING (oandapyV20.definitions.orders.OrderState attribute), 86 PENDING (oandapyV20.definitions.orders.OrderStateFilter attribute), 83

ORDER_FILL (oandapyV20.definitions.transactions.TransactionType attribute), 93 PENDING_ORDERS_ALLOWED_EXCEEDED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

ORDER_FILL_POSITION_ACTION_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 PENDING_ORDERS_ALLOWED_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

ORDER_FILL_POSITION_ACTION_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 POSITION_AGGREGATION_MODE_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

ORDER_ID_UNSPECIFIED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 POSITION_CLOSEOUT (oandapyV20.definitions.transactions.MarketOrderReason attribute), 92

ORDER_IDENTIFIER_INCONSISTENCY (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 POSITION_CLOSEOUT_FAILED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

ORDER_PARTIAL_FILL_OPTION_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 POSITION_SIZE_EXCEEDED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

ORDER_PARTIAL_FILL_OPTION_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96 PositionAggregationMode (class in oandapyV20.definitions.accounts), 80

OrderbookData (class in oandapyV20.endpoints.forexlabs), 31 PositionClose (class in oandapyV20.endpoints.positions), 51

OrderCancel (class in oandapyV20.endpoints.orders), 42 PositionCloseRequest (class in oandapyV20.contrib.requests), 113

OrderCancelReason (class in oandapyV20.definitions.transactions), 98 PositionDetails (class in oandapyV20.endpoints.positions), 53

OrderClientExtensions (class in oandapyV20.endpoints.orders), 43 PositionList (class in oandapyV20.endpoints.positions), 54

OrderCreate (class in oandapyV20.endpoints.orders), 44 POSITIVE (oandapyV20.definitions.trades.TradePL attribute), 87

OrderDetails (class in oandapyV20.endpoints.orders), 46 PRICE_BOUND_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

OrderFillReason (class in oandapyV20.definitions.transactions), 90 PRICE_BOUND_PRECISION_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

OrderID (class in oandapyV20.types), 103 PRICE_DISTANCE_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

OrderIdentifier (class in oandapyV20.types), 103 PRICE_DISTANCE_MAXIMUM_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

OrderList (class in oandapyV20.endpoints.orders), 46

OrderPositionFill (class in oandapyV20.definitions.orders), 84

OrderReplace (class in oandapyV20.endpoints.orders), 47

ORDERS_ON_FILL_DUPLICATE_CLIENT_ORDER_ID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

OrderSpecifier (class in oandapyV20.types), 104

OrdersPending (class in oandapyV20.endpoints.orders),

attribute), 96
 PRICE_DISTANCE_MINIMUM_NOT_MET (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 PRICE_DISTANCE_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 PRICE_DISTANCE_PRECISION_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 PRICE_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 PRICE_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 PRICE_PRECISION_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 PriceComponents (class in oandapyV20.definitions.instruments), 82
 PriceStatus (class in oandapyV20.definitions.pricing), 86
 PriceValue (class in oandapyV20.types), 104
 PricingInfo (class in oandapyV20.endpoints.pricing), 56
 PricingStream (class in oandapyV20.endpoints.pricing), 59

R

REDUCE_FIRST (oandapyV20.definitions.orders.OrderPositionFill attribute), 84
 REDUCE_ONLY (oandapyV20.definitions.orders.OrderPositionFill attribute), 85
 REGULATORY_MARGIN_CALL_VIOLATION (oandapyV20.definitions.transactions.MarketOrderMarginCloseoutReason attribute), 89
 REOPEN (oandapyV20.definitions.transactions.TransactionType attribute), 93
 REPLACEMENT (oandapyV20.definitions.transactions.LimitOrderReason attribute), 100
 REPLACEMENT (oandapyV20.definitions.transactions.MarketIfTouchedOrderReason attribute), 91
 REPLACEMENT (oandapyV20.definitions.transactions.StopLossOrderReason attribute), 90
 REPLACEMENT (oandapyV20.definitions.transactions.StopOrderReason attribute), 92
 REPLACEMENT (oandapyV20.definitions.transactions.TakeProfitOrderReason attribute), 94
 REPLACEMENT (oandapyV20.definitions.transactions.TrailingStopLossOrderReason attribute), 100
 REPLACING_ORDER_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 REPLACING_TRADE_ID_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 request() (oandapyV20.API method), 8
 request_params (oandapyV20.API attribute), 8
 RESETTABLE_PL (oandapyV20.definitions.transactions.TransactionType attribute), 93

S

S10 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81
 S15 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81
 S30 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81
 S5 (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81
 Saturday (oandapyV20.definitions.instruments.WeeklyAlignment attribute), 82
 SECOND_BY_SECOND (oandapyV20.definitions.accounts.AccountFinancingMode attribute), 79
 secs2time() (in module oandapyV20.contrib.generic), 108
 SITE_MIGRATION (oandapyV20.definitions.transactions.FundingReason attribute), 91
 Spreads (class in oandapyV20.endpoints.forexlabs), 33
 STOP (oandapyV20.definitions.orders.CancellableOrderType attribute), 84
 STOP (oandapyV20.definitions.orders.OrderType attribute), 83
 STOP_LOSS (oandapyV20.definitions.orders.CancellableOrderType attribute), 84
 STOP_LOSS (oandapyV20.definitions.orders.OrderType attribute), 83
 STOP_LOSS_ON_FILL_CLIENT_ORDER_COMMENT_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99
 STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96
 STOP_LOSS_ON_FILL_CLIENT_ORDER_TAG_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 96

STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST	StopLossOrderReason	(class in oan-
(oandapyV20.definitions.transactions.OrderCancelReason	dapyV20.definitions.transactions), 89	attribute), 99
STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST	StopLossOrderRequest	(class in oan-
(oandapyV20.definitions.transactions.TransactionRejectReason	dapyV20.contrib.requests), 114	attribute), 96
STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING	StopOrderRequest	(class in oan-
(oandapyV20.definitions.transactions.TransactionRejectReason	dapyV20.definitions.transactions), 92	attribute), 96
STOP_LOSS_ON_FILL_LOSS	STREAM	(oandapyV20.endpoints.pricing.PricingStream
(oandapyV20.definitions.transactions.OrderCancelReason	attribute), 59	attribute), 99
STOP_LOSS_ON_FILL_PRICE_INVALID	STREAM	(oandapyV20.endpoints.transactions.TransactionsStream
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 76	attribute), 97
STOP_LOSS_ON_FILL_PRICE_MISSING	Sunday	(oandapyV20.definitions.instruments.WeeklyAlignment
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 82	attribute), 97
STOP_LOSS_ON_FILL_PRICE_PRECISION_EXCEEDED	TAKE_PROFIT	(oandapyV20.definitions.orders.CancellableOrderType
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 84	attribute), 97
STOP_LOSS_ON_FILL_TIME_IN_FORCE_INVALID	TAKE_PROFIT	(oandapyV20.definitions.orders.OrderType
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 83	attribute), 97
STOP_LOSS_ON_FILL_TIME_IN_FORCE_MISSING	TAKE_PROFIT_ON_FILL_CLIENT_ORDER_COMMENT_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_LOSS_ON_FILL_TRIGGER_CONDITION_INVALID	TAKE_PROFIT_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_LOSS_ON_FILL_TRIGGER_CONDITION_MISSING	TAKE_PROFIT_ON_FILL_CLIENT_ORDER_ID_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_LOSS_ORDER	TAKE_PROFIT_ON_FILL_CLIENT_ORDER_TAG_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_LOSS_ORDER_ALREADY_EXISTS	TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_LOSS_ORDER_REJECT	TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_ORDER	TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
STOP_ORDER_REJECT	TAKE_PROFIT_ON_FILL_LOSS	(oandapyV20.definitions.transactions.TransactionRejectReason
(oandapyV20.definitions.transactions.TransactionRejectReason	attribute), 97	attribute), 97
StopLossDetails	TAKE_PROFIT_ON_FILL_PRICE_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason
(class in oandapyV20.contrib.requests), 120	dapyV20.definitions.transactions.TransactionRejectReason	attribute), 97
	TAKE_PROFIT_ON_FILL_PRICE_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason
	attribute), 97	attribute), 97
	TAKE_PROFIT_ON_FILL_PRICE_PRECISION_EXCEEDED	(oandapyV20.definitions.transactions.TransactionRejectReason
	attribute), 97	attribute), 97
	TAKE_PROFIT_ON_FILL_TIME_IN_FORCE_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason
	attribute), 97	attribute), 97

(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TAKE_PROFIT_ON_FILL_TIME_IN_FORCE_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TAKE_PROFIT_ON_FILL_TRIGGER_CONDITION_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TAKE_PROFIT_ON_FILL_TRIGGER_CONDITION_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TAKE_PROFIT_ORDER (oandapyV20.definitions.transactions.OrderFillReason attribute), 90

TAKE_PROFIT_ORDER (oandapyV20.definitions.transactions.TransactionType attribute), 93

TAKE_PROFIT_ORDER_ALREADY_EXISTS (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TAKE_PROFIT_ORDER_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 93

TakeProfitDetails (class in oandapyV20.contrib.requests), 121

TakeProfitOrderReason (class in oandapyV20.definitions.transactions), 94

TakeProfitOrderRequest (class in oandapyV20.contrib.requests), 116

terminate() (oandapyV20.endpoints.pricing.PricingStream method), 62

terminate() (oandapyV20.endpoints.transactions.TransactionStream method), 77

Thursday (oandapyV20.definitions.instruments.WeeklyAlignment attribute), 82

TIME_IN_FORCE_EXPIRED (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

TIME_IN_FORCE_GTD_TIMESTAMP_IN_PAST (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TIME_IN_FORCE_GTD_TIMESTAMP_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TIME_IN_FORCE_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TIME_IN_FORCE_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TimeInForce (class in oandapyV20.definitions.orders), 85

TRADE_CLIENT_EXTENSIONS_MODIFY (oandapyV20.definitions.transactions.TransactionType attribute), 93

TRADE_CLIENT_EXTENSIONS_MODIFY_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 94

TRADE_CLOSE (oandapyV20.definitions.transactions.MarketOrderReason attribute), 92

TRADE_DOESNT_EXIST (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TRADE_ID_UNSPECIFIED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TRADE_IDENTIFIER_INCONSISTENCY (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TRADE_ON_FILL_CLIENT_EXTENSIONS_NOT_SUPPORTED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TradeClientExtensions (class in oandapyV20.endpoints.trades), 65

TradeClose (class in oandapyV20.endpoints.trades), 66

TradeCloseRequest (class in oandapyV20.contrib.requests), 117

TradeCRCDO (class in oandapyV20.endpoints.trades), 63

TradeDetails (class in oandapyV20.endpoints.trades), 68

TradeID (class in oandapyV20.types), 104

TradePL (class in oandapyV20.definitions.trades), 87

TradesList (class in oandapyV20.endpoints.trades), 68

TradeState (class in oandapyV20.definitions.trades), 88

TradeStateFilter (class in oandapyV20.definitions.trades), 88

TRAILING_STOP_LOSS (oandapyV20.definitions.orders.CancellableOrderType attribute), 84

TRAILING_STOP_LOSS (oandapyV20.definitions.orders.OrderType attribute), 83

TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_COMMENT_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS (oandapyV20.definitions.transactions.OrderCancelReason attribute), 99

TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_TAG_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97

TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST (oandapyV20.definitions.transactions.OrderCancelReason attribute), 97

attribute), 99		dapyV20.contrib.requests), 118
TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97	TrailingStopLossDetails (class in oandapyV20.endpoints.transactions), 70
TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97	TransactionIDRange (class in oandapyV20.endpoints.transactions), 71
TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97	TransactionList (class in oandapyV20.endpoints.transactions), 72
TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_MAXIMALLY_EXCEEDED	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97	TransactionRejectReason (class in oandapyV20.definitions.transactions), 94
TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_MINIMUM_NOT_MET	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 97	TransactionsSinceID (class in oandapyV20.endpoints.transactions), 76
TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TransactionStream (class in oandapyV20.endpoints.transactions), 76
TRAILING_STOP_LOSS_ON_FILL_PRICE_DISTANCE_PRECISION_EXCEEDED	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TransactionType (class in oandapyV20.definitions.transactions), 92
TRAILING_STOP_LOSS_ON_FILL_TIME_IN_FORCE_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TRANSFER_FUNDS (oandapyV20.definitions.transactions.TransactionType attribute), 94
TRAILING_STOP_LOSS_ON_FILL_TIME_IN_FORCE_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TRANSFER_FUNDS_REJECT (oandapyV20.definitions.transactions.TransactionType attribute), 94
TRAILING_STOP_LOSS_ON_FILL_TRIGGER_CONDITION_INVALID	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TRIGGER_CONDITION_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TRAILING_STOP_LOSS_ON_FILL_TRIGGER_CONDITION_MISSING	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TRIGGER_CONDITION_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TRAILING_STOP_LOSS_ON_FILL_TRIGGER_CONDITION_ALREADY_TRIGGERED	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TRIGGERED (oandapyV20.definitions.orders.OrderStateFilter attribute), 86
TRAILING_STOP_LOSS_ON_FILL_TRIGGER_CONDITION_ALREADY_TRIGGERED	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	TRIGGERED (oandapyV20.definitions.orders.OrderStateFilter attribute), 83
TRAILING_STOP_LOSS_ON_FILL_TRIGGER_CONDITION_ALREADY_TRIGGERED	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	Units (class in oandapyV20.types), 104
TRAILING_STOP_LOSS_ORDER	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 90	UNITS_INVALID (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TRAILING_STOP_LOSS_ORDER	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 94	UNITS_LIMIT_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TRAILING_STOP_LOSS_ORDER_ALREADY_EXISTS	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	UNITS_MIMIMUM_NOT_MET (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TRAILING_STOP_LOSS_ORDER_REJECT	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 94	UNITS_MISSING (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TRAILING_STOP_LOSS_ORDERS_NOT_SUPPORTED	(oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98	UNITS_PRECISION_EXCEEDED (oandapyV20.definitions.transactions.TransactionRejectReason attribute), 98
TrailingStopLossDetails	(class in oandapyV20.contrib.requests), 122	
TrailingStopLossOrderReason	(class in oandapyV20.definitions.transactions), 99	
TrailingStopLossOrderRequest	(class in oandapyV20.contrib.requests), 122	

U

V

value (oandapyV20.types.AccountID attribute), 102
value (oandapyV20.types.AccountUnits attribute), 102
value (oandapyV20.types.ClientComment attribute), 102
value (oandapyV20.types.ClientID attribute), 102
value (oandapyV20.types.ClientTag attribute), 102
value (oandapyV20.types.DateTime attribute), 103
value (oandapyV20.types.OrderID attribute), 103
value (oandapyV20.types.OrderIdentifier attribute), 104
value (oandapyV20.types.OrderSpecifier attribute), 104
value (oandapyV20.types.PriceValue attribute), 104
value (oandapyV20.types.TradeID attribute), 104
value (oandapyV20.types.Units attribute), 104

W

W (oandapyV20.definitions.instruments.CandlestickGranularity attribute), 81
Wednesday (oandapyV20.definitions.instruments.WeeklyAlignment attribute), 82
WeeklyAlignment (class in oandapyV20.definitions.instruments), 81

Z

ZERO (oandapyV20.definitions.trades.TradePL attribute), 87