OANDA REST-V20 API Documentation

Release 0.4.2

Feite Brekeveld

oandapyV20 REST-V20 API wrapper

1	Intro	duction						
	1.1	Install						
	1.2	Download from Github						
2	Inter	face OANDA's REST-V20						
	2.1	The client						
	2.2	Exceptions						
	2.3	Logging 8						
3	oand	oandapyV20.endpoints						
	3.1	oandapyV20.endpoints.accounts						
	3.2	oandapyV20.endpoints.instruments						
	3.3	oandapyV20.endpoints.orders						
	3.4	oandapyV20.endpoints.positions						
	3.5	oandapyV20.endpoints.pricing						
	3.6	oandapyV20.endpoints.trades						
	3.7	oandapyV20.endpoints.transactions						
4	oand	apyV20.definitions						
	4.1	oandapyV20.definitions.accounts						
	4.2	oandapy V20. definitions. instruments						
	4.3	oandapy V20. definitions. orders						
	4.4	oandapyV20.definitions.pricing						
	4.5	oandapyV20.definitions.trades						
	4.6	oandapyV20.definitions.transactions						
5 08	oand	apyV20.types						
	5.1	AccountID						
	5.2	AccountUnits						
	5.3	ClientComment						
	5.4	ClientID						
	5.5	ClientTag						
	5.6	DateTime						
	5.7	OrderID						
	5.8	OrderIdentifier						
	5.9	OrderSpecifier						
	5.10	Price Value						
		• • • • • • • • • • • • • • • • • • • •						

	5.11 TradeID	84			
	5.12 Units	84			
6	andapyV20.contrib				
	6.1 Factories	87			
	6.2 Generic	88			
	6.3 Order Classes	88			
	6.4 support classes	98			
7	Examples	105			
	7.1 Example for trades-endpoints	105			
8	Indices and tables	107			
Python Module Index					

Contents:

CHAPTER 1

Introduction

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.

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.

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

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 it's 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)))
```

```
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)))
```

```
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"
1.
"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}

2.1. The client 7

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.

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

Logging

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

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

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

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

Resulting loglines:

```
2016-10-22 17:50:37,988 [INFO] oandapyV20.oandapyV20 : setting up API-client for → environment practice
2016-10-22 17:50:37,990 [INFO] oandapyV20.oandapyV20 : performing request https://api- → fxpractice.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'"}]
```

2.3. Logging 9

CHAPTER 3

oandapyV20.endpoints

oandapyV20.endpoints.accounts

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.

Parameters

- account ID (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
```

```
"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"
}
```

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"

}
```

AccountDetails

 ${f class}$ oandapy V20.endpoints.accounts.Account Details (${\it accountID}$)

Bases: oandapy V20. 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 account ID (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"
```

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

- account ID (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
```

```
"instruments": [
    "marginRate": "0.05",
    "minimumTrailingStopDistance": "5.0",
    "maximumPositionSize": "0",
    "minimumTradeSize": "1",
    "displayName": "Europe 50",
    "name": "EU50_EUR",
   "displayPrecision": 1,
   "maximumTrailingStopDistance": "10000.0",
   "maximumOrderUnits": "3000",
    "tradeUnitsPrecision": 0,
    "pipLocation": 0,
    "type": "CFD"
  },
    "marginRate": "0.05",
    "minimumTrailingStopDistance": "0.00050",
    "maximumPositionSize": "0",
    "minimumTradeSize": "1",
    "displayName": "EUR/USD",
    "name": "EUR_USD",
    "displayPrecision": 5,
    "maximumTrailingStopDistance": "1.00000",
    "maximumOrderUnits": "100000000",
```

```
"tradeUnitsPrecision": 0,
  "pipLocation": -4,
  "type": "CURRENCY"
},
  "marginRate": "0.05",
  "minimumTrailingStopDistance": "5.0",
  "maximumPositionSize": "0",
  "minimumTradeSize": "1",
  "displayName": "US Wall St 30",
  "name": "US30_USD",
  "displayPrecision": 1,
  "maximumTrailingStopDistance": "10000.0",
  "maximumOrderUnits": "1000",
  "tradeUnitsPrecision": 0,
  "pipLocation": 0,
  "type": "CFD"
},
  "marginRate": "0.05",
  "minimumTrailingStopDistance": "5.0",
  "maximumPositionSize": "0",
  "minimumTradeSize": "1",
  "displayName": "France 40",
  "name": "FR40_EUR",
  "displayPrecision": 1,
  "maximumTrailingStopDistance": "10000.0",
  "maximumOrderUnits": "2000",
  "tradeUnitsPrecision": 0,
  "pipLocation": 0,
  "type": "CFD"
},
  "marginRate": "0.05",
  "minimumTrailingStopDistance": "0.00050",
  "maximumPositionSize": "0",
  "minimumTradeSize": "1",
  "displayName": "EUR/CHF",
  "name": "EUR_CHF",
  "displayPrecision": 5,
  "maximumTrailingStopDistance": "1.00000",
  "maximumOrderUnits": "100000000",
  "tradeUnitsPrecision": 0,
  "pipLocation": -4,
  "type": "CURRENCY"
},
  "marginRate": "0.05",
  "minimumTrailingStopDistance": "5.0",
  "maximumPositionSize": "0",
  "minimumTradeSize": "1",
  "displayName": "Germany 30",
  "name": "DE30_EUR",
  "displayPrecision": 1,
  "maximumTrailingStopDistance": "10000.0",
  "maximumOrderUnits": "2500",
  "tradeUnitsPrecision": 0,
  "pipLocation": 0,
```

```
"type": "CFD"
}
],
"lastTransactionID": "2124"
}
```

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": []
           ]
```

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"
```

oandapyV20.endpoints.instruments

InstrumentsCandles

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
```

```
"candles": [
    "volume": 132,
    "mid": {
     "h": "10508.0",
     "c": "10506.0",
     "1": "10503.8",
      "o": "10503.8"
    },
    "complete": true,
    "time": "2016-10-17T19:35:00.00000000Z"
  },
    "volume": 162,
    "mid": {
     "h": "10507.0",
     "c": "10504.9",
     "1": "10502.0",
     "o": "10506.0"
   },
    "complete": true,
    "time": "2016-10-17T19:40:00.00000000Z"
  },
    "volume": 196,
    "mid": {
     "h": "10509.8",
```

```
"c": "10505.0",
      "1": "10502.6",
      "o": "10504.9"
    },
    "complete": true,
    "time": "2016-10-17T19:45:00.00000000Z"
  },
  {
    "volume": 153,
    "mid": {
      "h": "10510.1",
      "c": "10509.0",
     "1": "10504.2",
      "o": "10505.0"
    },
    "complete": true,
    "time": "2016-10-17T19:50:00.00000000Z"
  },
    "volume": 172,
    "mid": {
      "h": "10509.8",
      "c": "10507.8",
      "1": "10503.2",
      "o": "10509.0"
    },
    "complete": true,
    "time": "2016-10-17T19:55:00.00000000Z"
],
"granularity": "M5",
"instrument": "DE30/EUR"
```

InstrumentsOrderBook

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:

```
"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"
```

InstrumentsPositionBook

___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:

```
"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"
}
```

oandapyV20.endpoints.orders

OrderCancel

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"
    ]
}
```

OrderClientExtensions

```
\begin{tabular}{ll} \textbf{class} \texttt{ oandapyV20.endpoints.orders.OrderClientExtensions} (accountID, orderID, data) \\ \textbf{Bases:} \texttt{ oandapyV20.endpoints.orders.Orders} \end{tabular}
```

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.

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"
1
```

OrderCreate

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"
]
}
```

OrderDetails

Parameters

- account ID (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
```

```
"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"
}
```

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
```

```
"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"
}
```

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 createing a replacement Order.
    ENDPOINT = 'v3/accounts/{accountID}/orders/{orderID}'
    EXPECTED_STATUS = 201
    HEADERS = {'Content-Type': 'application/json'}
```

__init__ (accountID, orderID, data)

Instantiate an OrderReplace request.

Parameters

METHOD = 'PUT'

- account ID (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"
]
```

OrdersPending

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

Example:

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

Output:

```
"orders": [
  {
    "partialFill": "DEFAULT_FILL",
    "price": "1.20000",
    "stopLossOnFill": {
     "timeInForce": "GTC",
      "price": "1.22000"
    "timeInForce": "GTC",
    "clientExtensions": {
     "comment": "myComment",
     "id": "myID"
    "id": "2304",
    "triggerCondition": "TRIGGER_DEFAULT",
    "positionFill": "POSITION_DEFAULT",
    "createTime": "2016-10-24T21:48:18.593753865Z",
    "instrument": "EUR_USD",
    "state": "PENDING",
    "units": "-100",
    "type": "LIMIT"
],
"lastTransactionID": "2305"
```

oandapyV20.endpoints.positions

OpenPositions

```
\begin{tabular}{l} \textbf{class} \verb| oandapyV20.endpoints.positions.OpenPositions ($accountID$) \\ \textbf{Bases:} \verb| oandapyV20.endpoints.positions.Positions \\ \end{tabular}
```

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
```

```
"positions": [
 {
   "short": {
     "units": "0",
     "resettablePL": "-14164.3000",
     "unrealizedPL": "0.0000",
     "pl": "-14164.3000"
   },
    "unrealizedPL": "-284.0000",
    "long": {
     "unrealizedPL": "-284.0000",
     "tradeIDs": [
       "2315"
     1,
     "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"
    }
    l,
    "lastTransactionID": "2327"
}
```

PositionClose

```
{\bf class} \ {\tt oandapyV20.endpoints.positions.PositionClose} \ ({\it accountID}, {\it instrument}, {\it data}) \\ {\tt Bases:} \ {\tt oandapyV20.endpoints.positions.Positions}
```

Closeout the open Position regarding instrument in an Account.

Parameters

- account ID (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"
}
```

```
"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>"
```

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)
```

Parameters

Instantiate a PositionDetails request.

- 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
```

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

PositionList

 $\begin{tabular}{l} \textbf{class} \verb| oandapyV20.endpoints.positions.PositionList(| \textit{accountID}) \\ \textbf{Bases:} \verb| oandapyV20.endpoints.positions.Positions \\ \end{tabular}$

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
```

```
"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"
```

oandapyV20.endpoints.pricing

Parameters

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.
```

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

Example

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

```
"prices": [
 {
   "status": "tradeable",
   "instrument": "EUR_USD",
   "quoteHomeConversionFactors": {
     "negativeUnits": "0.89160730",
     "positiveUnits": "0.89150397"
   },
    "asks": [
       "price": "1.12170",
        "liquidity": 10000000
      },
        "price": "1.12172",
       "liquidity": 10000000
     }
   1,
    "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"
]
```

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

```
>>> 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")
```

```
"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.

oandapyV20.endpoints.trades

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 account ID (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:

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
```

```
"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"
}
```

TradeClientExtensions

 $Bases: \verb|oandapyV20.endpoints.trades.Trades|\\$

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

- account ID (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"
    }
}
```

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"
]
```

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
}
```

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

```
"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",
```

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
```

```
{
  "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"
```

```
}
```

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"
}
```

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

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

oandapyV20.endpoints.transactions

TransactionDetails

Parameters

- account ID (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
```

```
"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"
}
```

TransactionIDRange

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
}
```

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

```
"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"
]
```

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
```

```
"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"
 1
```

TransactionsSinceID

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
}
```

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

```
"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"
]
```

TransactionsStream

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.

CHAPTER 4

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.

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.

```
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.

```
ABSOLUTE_SUM = 'ABSOLUTE_SUM'

MAXIMAL_SIDE = 'MAXIMAL_SIDE'

NET_SUM = 'NET_SUM'

__getitem__ (definitionID)
    return description for definitionID.

definitions
    readonly property holding definition dict.
```

oandapyV20.definitions.instruments

Instruments Definitions.

```
class oandapy V20. definitions. instruments. Candlestick Granularity 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'
     M15 = 'M15'
     M2 = 'M2'
     M30 = 'M30'
     M4 = 'M4'
     M5 = 'M5'
     $10 = 'S10'
     S15 = 'S15'
     $30 = 'S30'
     S5 = 'S5'
     W = W'
     __getitem__(definitionID)
          return description for definitionID.
     definitions
          readonly property holding definition dict.
class oandapy V20. 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.
{\bf class} \ {\tt oandapy V20.definitions.instruments.Price Components}
     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
     >>> 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.

oandapyV20.definitions.orders

Order related definitions.

```
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"
     >>> 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 oandapy V20. 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.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.OrderPositionFill
```

Definition representation of OrderPositionFill

Bases: object

```
>>> 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.

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'
```

oandapyV20.definitions.trades

Trades definitions.

```
\begin{array}{c} \textbf{class} \; \texttt{oandapyV20.definitions.trades.TradeState} \\ & \textbf{Bases:} \; \texttt{object} \end{array}
```

Definition representation of TradeState

```
>>> 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.
```

oandapyV20.definitions.transactions

Transactions definitions.

```
{\bf class} \ {\tt oandapyV20.definitions.transactions.StopLossOrderReason} \\ Bases: \ {\tt object}
```

Definition representation of StopLossOrderReason

```
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 kevs
>>> 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'
     <u>__getitem__</u>(definitionID)
         return description for definitionID.
     definitions
         readonly property holding definition dict.
class oandapy V20. 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 kevs
>>> 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'
     <u>__getitem__</u>(definitionID)
         return description for definitionID.
     definitions
         readonly property holding definition dict.
class oandapy V20. 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]
```

CLIENT_ORDER = 'CLIENT_ORDER'

REPLACEMENT = 'REPLACEMENT'

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

definitions

>>> # all keys

readonly property holding definition dict.

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

>>> print deftransactions.StopOrderReason().definitions.keys()

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'
LIMIT ORDER = 'LIMIT ORDER'
LIMIT ORDER REJECT = 'LIMIT ORDER REJECT'
MARGIN_CALL_ENTER = 'MARGIN_CALL_ENTER'
MARGIN_CALL_EXIT = 'MARGIN_CALL_EXIT'
MARGIN CALL EXTEND = 'MARGIN CALL EXTEND'
MARKET_IF_TOUCHED_ORDER = 'MARKET_IF_TOUCHED_ORDER'
MARKET_IF_TOUCHED_ORDER_REJECT = 'MARKET_IF_TOUCHED_ORDER_REJECT'
MARKET_ORDER = 'MARKET_ORDER'
MARKET ORDER REJECT = 'MARKET ORDER REJECT'
ORDER CANCEL = 'ORDER CANCEL'
ORDER CLIENT EXTENSIONS MODIFY = 'ORDER CLIENT EXTENSIONS MODIFY'
ORDER CLIENT EXTENSIONS MODIFY REJECT = 'ORDER CLIENT EXTENSIONS MODIFY REJECT'
ORDER FILL = 'ORDER FILL'
REOPEN = 'REOPEN'
RESET_RESETTABLE_PL = 'RESET_RESETTABLE_PL'
STOP LOSS ORDER = 'STOP LOSS ORDER'
STOP_LOSS_ORDER_REJECT = 'STOP_LOSS_ORDER_REJECT'
STOP_ORDER = 'STOP_ORDER'
STOP_ORDER_REJECT = 'STOP_ORDER_REJECT'
TAKE PROFIT ORDER = 'TAKE PROFIT ORDER'
TAKE PROFIT ORDER REJECT = 'TAKE PROFIT ORDER REJECT'
TRADE CLIENT EXTENSIONS MODIFY = 'TRADE CLIENT EXTENSIONS MODIFY'
TRADE_CLIENT_EXTENSIONS_MODIFY_REJECT = 'TRADE_CLIENT_EXTENSIONS_MODIFY_REJECT'
TRAILING STOP LOSS ORDER = 'TRAILING STOP LOSS ORDER'
TRAILING STOP LOSS ORDER REJECT = 'TRAILING STOP LOSS ORDER REJECT'
TRANSFER_FUNDS = 'TRANSFER_FUNDS'
TRANSFER_FUNDS_REJECT = 'TRANSFER_FUNDS_REJECT'
__getitem_ (definitionID)
   return description for definitionID.
definitions
   readonly property holding definition dict.
```

class oandapy V20. definitions. transactions. Take Profit Order Reason Bases: object

Definition representation of TakeProfitOrderReason

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

```
CLIENT_ORDER = 'CLIENT_ORDER'

ON_FILL = 'ON_FILL'

REPLACEMENT = 'REPLACEMENT'

__getitem__ (definitionID)

return description for definitionID.
```

definitions

readonly property holding definition dict.

Definition representation of OrderCancelReason

```
>>> 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 A
    STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST'
    STOP LOSS ON FILL LOSS = 'STOP LOSS ON FILL LOSS'
    TAKE PROFIT ON FILL CLIENT ORDER ID ALREADY EXISTS = 'TAKE PROFIT ON FILL CLIENT ORDER
    TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAST
    TAKE_PROFIT_ON_FILL_LOSS = 'TAKE_PROFIT_ON_FILL_LOSS'
    TIME IN FORCE EXPIRED = 'TIME IN FORCE EXPIRED'
    TRAILING_STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS = 'TRAILING_STOP_LOSS_ON_FILL
    TRAILING_STOP_LOSS_ON_FILL_GTD_TIMESTAMP_IN_PAST = 'TRAILING_STOP_LOSS_ON_FILL_GTD_TIME
    __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
```

```
>>> 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 oandapy V20. 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.

CHAPTER 5

oandapyV20.types

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 allways be a string value.

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

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.
```

AccountUnits

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

ClientComment

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

ClientID

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

ClientTag

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

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

A ValueError exception is raised in case of an invalid value

```
__init__(dateTime)

value

value property.
```

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

```
A ValueError exception is raised in case of a negative integer value
__init__(orderID)
```

value

value property.

OrderIdentifier

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

5.7. OrderID 83

```
__init__ (orderID, clientID)

value

value property.
```

OrderSpecifier

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

PriceValue

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

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.
```

Units

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

value

value property.

5.12. Units 85

oandapyV20.contrib

Factories

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

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.

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.

Generic

The oandapy V20.contrib.generic module contains several classes / methods that serve a generic purpose.

granularity to time

```
oandapy V20. contrib. generic. \texttt{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'
```

Order Classes

The oandapy V20.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.

LimitOrderRequest

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest create a LimitOrderRequest.

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

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

Parameters

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

Example

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

6.3. Order Classes

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

data

data property.

return the JSON order body

MarketOrderRequest

```
class oandapyV20.contrib.requests.MarketOrderRequest (instrument, units, price-
Bound=None, position-
Fill='DEFAULT', clientEx-
tensions=None, takeProfitOn-
Fill=None, timeInForce='FOK',
stopLossOnFill=None, trailingSto-
pLossOnFill=None, tradeClien-
tExtensions=None)
```

 $Bases: \verb| oandapyV20.contrib.requests.| baserequest. BaseRequest|$

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, trailingStopLossOn-Fill=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

```
>>> 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. Order Classes 91

MITOrderRequest

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

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest create a MarketIfTouched OrderRequest.

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

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

Parameters

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

Example

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

PositionCloseRequest

```
 \begin{array}{c} \textbf{class} \; \texttt{oandapyV20.contrib.requests.PositionCloseRequest} \; (longUnits=None, \\ tExtensions=None, \\ nits=None, \\ shortUnits=None, \\ sions=None) \\ \textbf{Bases:} \; \texttt{oandapyV20.contrib.requests.baserequest.BaseRequest} \end{array}
```

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. Order Classes 93

StopLossOrderRequest

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, clientExtensions=None)
Instantiate a StopLossOrderRequest.
```

Parameters

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

Example

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import StopLossOrderRequest
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> ordr = StopLossOrderRequest(tradeID="1234", price=1.07)
>>> print(json.dumps(ordr.data, indent=4))
    "order": {
        "type": "STOP_LOSS",
        "tradeID": "1234",
        "price": "1.07000",
        "timeInForce": "GTC",
>>> # now we have the order specification, create the order request
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=4))
```

data

data property.

return the JSON body.

StopOrderRequest

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)
```

6.3. Order Classes 95

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

data

data property.

return the JSON body.

TakeProfitOrderRequest

Bases: oandapyV20.contrib.requests.baserequest.BaseRequest

create a TakeProfit OrderRequest.

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

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

Parameters

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

```
>>> 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

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)
>>> ...
```

TrailingStopLossOrderRequest

```
 \begin{array}{c} \textbf{class} \ \texttt{oandapyV20.contrib.requests.TrailingStopLossOrderRequest} \ (\textit{tradeID}, \\ & \textit{tance}, \\ & \textit{client-TradeID=None}, \\ & \textit{timeIn-Force='GTC'}, \\ & \textit{gtdTime=None}, \\ & \textit{clientExten-sions=None}) \\ \\ \textbf{Bases: oandapyV20.contrib.requests.baserequest.BaseRequest} \end{array}
```

6.3. Order Classes 97

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, clientExten-
sions=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.

support classes

The oandapy V20.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.

Client Extensions

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

__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

```
>>> 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)
>>> ...
```

StopLossDetails

```
 \begin{array}{ll} \textbf{class} \; \texttt{oandapyV20.contrib.requests.StopLossDetails} \; (\textit{price}, & \textit{timeInForce='GTC'}, & \textit{gtd-Time=None}, \textit{clientExtensions=None}) \\ & \textbf{Bases:} \; \texttt{oandapyV20.contrib.requests.onfill.OnFill} \end{array}
```

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)) -

```
>>> import json
>>> from oandapyV20 import API
>>> import oandapyV20.endpoints.orders as orders
>>> from oandapyV20.contrib.requests import (
        MarketOrderRequest, StopLossDetails)
>>>
>>>
>>> accountID = "..."
>>> client = API(access_token=...)
>>> # at time of writing EUR_USD = 1.0740
>>> # let us take profit at 1.10, GoodTillCancel (default)
>>> stopLossOnFill = StopLossDetails(price=1.06)
>>> print(stopLossOnFill)
    "timeInForce": "GTC",
    "price": "1.10000"
>>> ordr = MarketOrderRequest (
      instrument="EUR_USD",
>>>
       units=10000,
>>>
        stopLossOnFill=stopLossOnFill.data
>>> )
>>> # or as shortcut ...
>>> # stopLossOnFill=StopLossDetails(price=1.06).data
>>> print(json.dumps(ordr.data, indent=4))
>>> r = orders.OrderCreate(accountID, data=ordr.data)
>>> rv = client.request(r)
>>> ...
```

TakeProfitDetails

```
 \begin{array}{c} \textbf{class} \ \texttt{oandapyV20.contrib.requests.TakeProfitDetails} \ (price, & timeInForce='GTC', \\ & gtdTime=None, & clientExtensions=None) \\ \textbf{Bases:} \ \texttt{oandapyV20.contrib.requests.onfill.OnFill} \end{array}
```

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

```
>>> 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)
>>> ...
```

TrailingStopLossDetails

```
class oandapyV20.contrib.requests. TrailingStopLossDetails (distance, force='GTC', gtd-Time=None, clientExtensions=None)
```

 $Bases: \verb|oandapyV20.contrib.requests.onfill.OnFill|\\$

Representation of the specification for a TrailingStopLossOrder.

It is typically used to specify 'trailing stop loss details' for the 'trailingStopLossOnFill' parameter of an Order-Request. 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)) -

```
>>> 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)
```

CHAPTER 7

Examples

Examples can be found in the examples repositiony on github: examples reposition.

Example for trades-endpoints

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

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

```
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 0 in sys.argv[2:]:
      r = trades.TradeDetails(accountID, tradeID=0)
      rv = api.request(r)
      print("RESP:\n{} ".format(json.dumps(rv, indent=2)))
if chc == 'close':
  X = iter(sys.argv[2:])
  for 0 in X:
      cfg = { "units": X.next() }
      r = trades.TradeClose(accountID, tradeID=0, data=cfg)
      rv = api.request(r)
      print("RESP:\n{} ".format(json.dumps(rv, indent=2)))
if chc == 'cltext':
  for 0 in sys.argv[2:]: # tradeIDs
       cfg = { "clientExtensions": {
              "id": "myID{}".format(0),
               "comment": "myComment",
      r = trades.TradeClientExtensions(accountID, tradeID=0, 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 0 in X:
       cfg = {
               "takeProfit": {
                 "timeInForce": "GTC",
                 "price": X.next(),
               },
               "stopLoss": {
                 "timeInForce": "GTC",
                 "price": X.next()
               }
       r = trades.TradeCRCDO(accountID, tradeID=0, data=cfg)
       rv = api.request(r)
      print("RESP:\n{} ".format(json.dumps(rv, indent=2)))
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

0

```
oandapyV20.contrib.factories, 87
oandapyV20.contrib.generic, 88
oandapyV20.definitions.accounts, 65
oandapyV20.definitions.instruments, 66
oandapyV20.definitions.orders, 68
oandapyV20.definitions.pricing, 71
oandapyV20.definitions.trades, 71
oandapyV20.definitions.transactions, 72
```

110 Python Module Index

```
Symbols
                                                           _getitem__() (oandapy V20.definitions.transactions.TakeProfitOrderReason
__getitem__() (oandapyV20.definitions.accounts.AccountFinancingMmcthod), 77
                                                          _getitem__() (oandapy V20.definitions.transactions.TrailingStopLossOrder
         method), 65
__getitem__() (oandapyV20.definitions.accounts.PositionAggregationWoded), 79
                                                           _getitem__() (oandapy V20.definitions.transactions.TransactionType
         method), 66
__getitem__() (oandapyV20.definitions.instruments.CandlestickGranufacthod), 76
                                                          _init__() (oandapyV20.API method), 7
         method), 67
_getitem__() (oandapyV20.definitions.instruments.PriceComponents (oandapyV20.V20Error method), 8
                                                           init__() (oandapyV20.contrib.requests.ClientExtensions
         method), 68
__getitem__() (oandapyV20.definitions.instruments.WeeklyAlignmentmethod), 99
                                                          _init__() (oandapyV20.contrib.requests.LimitOrderRequest
         method), 68
                                                                   method), 89
__getitem__() (oandapyV20.definitions.orders.OrderPositionFill
                                                           _init__() (oandapy V20.contrib.requests.MITOrderRequest
         method), 70
                                                                   method), 92
 getitem () (oandapy V20.definitions.orders.OrderState
                                                           _init__() (oandapy V20.contrib.requests.MarketOrderRequest
         method), 70
                                                                   method), 90
__getitem__() (oandapy V20.definitions.orders.OrderType
                                                           _init__() (oandapy V20.contrib.requests.PositionCloseRequest
         method), 69
                                                                   method), 93
__getitem__() (oandapy V20.definitions.orders.TimeInForce
                                                           _init__() (oandapy V20.contrib.requests.StopLossDetails
         method), 69
                                                                   method), 100
__getitem__() (oandapyV20.definitions.pricing.PriceStatus
                                                           _init__() (oandapy V20.contrib.requests.StopLossOrderRequest
         method), 71
                                                                   method), 94
 _getitem__() (oandapyV20.definitions.trades.TradeState
                                                           _init__() (oandapy V20.contrib.requests.StopOrderRequest
         method), 72
__getitem__() (oandapyV20.definitions.transactions.FundingReason method), 95
                                                           _init__() (oandapyV20.contrib.requests.TakeProfitDetails
         method), 74
 _getitem__() (oandapyV20.definitions.transactions.LimitOrderReasomethod), 101
                                                          _init__() (oandapy V20.contrib.requests.TakeProfitOrderRequest
         method), 79
_getitem_() (oandapy V20.definitions.transactions.MarketIfTouched Of defices on
                                                         __init__() (oandapyV20.contrib.requests.TradeCloseRequest
         method), 74
_getitem__() (oandapyV20.definitions.transactions.MarketOrderReasthand), 97
                                                           _init__() (oandapyV20.contrib.requests.TrailingStopLossDetails
         method), 75
__getitem__() (oandapyV20.definitions.transactions.OrderCancelReason, 102
                                                          _init__() (oandapy V20.contrib.requests.TrailingStopLossOrderRequest
         method), 78
__getitem__() (oandapyV20.definitions.transactions.OrderFillReason method), 98
                                                          _init__() (oandapy V20.endpoints.accounts.AccountChanges
         method), 73
__getitem__() (oandapyV20.definitions.transactions.StopLossOrderReasonod), 11
                                                         __init__() (oandapy V20.endpoints.accounts.AccountConfiguration
         method), 72
__getitem__() (oandapyV20.definitions.transactions.StopOrderReasonmethod), 13
                                                         __init__() (oandapy V20.endpoints.accounts.AccountDetails
         method), 75
                                                                   method), 14
```

$\underline{\hspace{0.3cm}} init\underline{\hspace{0.3cm}} () (o and apy V20. endpoints. accounts. Account Instruments and the contraction of the contract$	meritsit() (oandapy V20.endpoints.transactions.TransactionList
method), 17	method), 59
· · · · · · · · · · · · · · · · · · ·	init() (oandapy V20.endpoints.transactions.TransactionsSinceID
method), 19	method), 60
	<pre>aryinit() (oandapyV20.endpoints.transactions.TransactionsStream</pre>
method), 19	method), 62
init() (oandapyV20.endpoints.instruments.Instruments	
method), 21	init() (oandapyV20.types.AccountUnits method), 82
	OrderiBook (oandapy V20.types. Client Comment method),
method), 22	82
init() (oandapyV20.endpoints.instruments.Instruments	
method), 25	init() (oandapyV20.types.ClientTag method), 82
init() (oandapyV20.endpoints.orders.OrderCancel	
method), 29	init() (oandapyV20.types.OrderID method), 83
* * * * * * * * * * * * * * * * * * * *	nsioinsit() (oandapyV20.types.OrderIdentifier method),
method), 30	83
	init() (oandapyV20.types.OrderSpecifier method),
method), 31	init() (oandapyV20.types.PriceValue method), 84
init() (oandapyV20.endpoints.orders.OrderDetails method), 32	init() (oandapy V20.types.Frice value inethod), 84 init() (oandapy V20.types.TradeID method), 84
	init() (oandapy V20.types. Trade1D inethod), 84 init() (oandapy V20.types. Units method), 84
method), 33	init() (oandapy v 20.types.Onits method), 84
init() (oandapyV20.endpoints.orders.OrderReplace	A
method), 34	
init() (oandapyV20.endpoints.orders.OrdersPending	A (oandapy V20.definitions.instruments.PriceComponents
method), 35	attribute), 68 ABSOLUTE_SUM (oan-
init() (oandapy V20.endpoints.positions.OpenPositions	dapyV20.definitions.accounts.PositionAggregationMode
method), 37	attribute), 66
init() (oandapyV20.endpoints.positions.PositionClose	ACCOUNT_LOCKED (oan-
method), 38	dapyV20.definitions.transactions.OrderCancelReason
init() (oandapy V20.endpoints.positions.PositionDetails	s attribute), 77
method), 40	ACCOUNT_NEW_POSITIONS_LOCKED (oan-
init() (oandapyV20.endpoints.positions.PositionList	dapy V20. definitions. transactions. Order Cancel Reason
method), 41	attribute), 77
init() (oandapyV20.endpoints.pricing.PricingInfo	ACCOUNT_ORDER_CREATION_LOCKED (oan-
method), 42	dapyV20.definitions.transactions.OrderCancelReason
init() (oandapyV20.endpoints.pricing.PricingStream	attribute), 77
method), 45	ACCOUNT_ORDER_FILL_LOCKED (oan-
init() (oandapyV20.endpoints.trades.OpenTrades	dapyV20.definitions.transactions.OrderCancelReason
method), 48	attribute), 77
init() (oandapyV20.endpoints.trades.TradeCRCDO	ACCOUNT_TRANSFER (oan-
method), 49	dapyV20.definitions.transactions.FundingReason
init() (oandapyV20.endpoints.trades.TradeClientExten	
method), 51	AccountChanges (class in oan-
init() (oandapyV20.endpoints.trades.TradeClose	dapyV20.endpoints.accounts), 11
method), 52	AccountConfiguration (class in oan-
init() (oandapyV20.endpoints.trades.TradeDetails	dapyV20.endpoints.accounts), 13
method), 54	AccountDetails (class in oan-
init() (oandapyV20.endpoints.trades.TradesList	dapyV20.endpoints.accounts), 14
method), 55init() (oandapy V20.endpoints.transactions.Transaction	AccountFinancingMode (class in oan-
method), 56	dapy v zorodnimironsiace cumis), ee
init() (oandapy V20.endpoints.transactions.Transaction	AccountID (class in oandapyV20.types), 81
method), 57	· ·
mourou, or	dapyV20.endpoints.accounts), 16

AccountList (class in oandapyV20.endpoints.account	ınts),	CLIENT_		(oan-
19			dapyV20.definitions.transactions.OrderCa	incelReason
` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` `	oan-	CI IENTE	attribute), 77	,
dapyV20.endpoints.accounts), 19		CLIENT_	_TRADE_ID_ALREADY_EXISTS	(oan-
AccountUnits (class in oandapyV20.types), 82			dapyV20.definitions.transactions.OrderCa	incelReason
ADJUSTMENT (oandapy V20. definitions. transaction		_		
attribute), 73			mment (class in oandapyV20.types), 82	
API (class in oandapyV20), 5		ClientExt	tensions (class in oandapyV20.contrib.requipped)	iests),
В		ClientID	(class in oandapyV20.types), 82	
B (oandapy V20.definitions.instruments.PriceCompon			g (class in oandapyV20.types), 82	
attribute), 68			oandapyV20.definitions.transactions.Trans	actionType
	oan-	`	attribute), 76	71
dapyV20.definitions.transactions.OrderCan	celRes	CLOSE		(oan-
attribute), 77	iccirce	45011 —	dapyV20.definitions.trades.TradeState	at-
attribute), 77			tribute), 72	
C		CLOSED	oandapyV20.definitions.trades.TradeSta	te at-
			tribute), 72	
CANCELLED (oandapy V20. definitions. orders. Order	rstate	CREATE	(oandapyV20.definitions.transactions.Tran	nsactionType
attribute), 70			attribute), 76	71
•	oan-		**	
dapyV20.definitions.instruments), 66	·	D		
CLIENT_CONFIGURE (example dapy V20. definitions. transactions. Transactions.	oan-	D (oanda	nvV20 definitions instruments Candlestick	Granularity
	onType	(Odrida)	attribute), 66	Grandiarity
attribute), 75	·	DAII V (d	pandapyV20.definitions.accounts.Account	FinancingMode
,	,		attribute), 65	manemgiviode
dapyV20.definitions.transactions.Transactio	onType	DAILY I	FINANCING	(oan-
attribute), 75		D. HET_I	dapyV20.definitions.transactions.Transact	`
_ `	oan-		attribute), 76	nonType
dapyV20.definitions.transactions.FundingR	Ceason	data (o	andapyV20.contrib.requests.LimitOrderRe	eauest
attribute), 74		uuu (o	attribute), 90	quest
CLIENT_ORDER (edapyV20.definitions.transactions.LimitOrde	oan-	data (oai		eauest
otteibute) 70	erkeas	OH COL	attribute), 91	quest
attribute), 79	oan-	data (oan	dapyV20.contrib.requests.MITOrderReque	est at-
CLIENT_ORDER (dapyV20.definitions.transactions.MarketIfT				
	Touche	data (oan	dapyV20.contrib.requests.StopLossOrderR	equest
attribute), 74 CLIENT_ORDER (6		aata (oan	attribute), 94	equest
dapyV20.definitions.transactions.MarketOr	oan-	data (oan	danyV20 contrib requests StonOrderReque	est at-
ottribute) 74	ideikea	ason (can	tribute), 96	
attribute), 74	.	data (oan	dapyV20.contrib.requests.TakeProfitOrderl	Request
_ `			attribute), 96	request
dapyV20.definitions.transactions.StopLoss	Orderk	data (oan	dapyV20.contrib.requests.TrailingStopLos	sOrderRequest
attribute), 72 CLIENT_ORDER (6		aata (oan	attribute), 98	Soracificquest
dapyV20.definitions.transactions.StopOrde	oan-	.DateTime	c (class in oandapyV20.types), 82	
attributa) 75	Reasc	DEFAUL	T (oandapy V20.definitions.orders.OrderPo	sitionFill
attribute), 75 CLIENT_ORDER (6		DEFFICE	attribute), 70	
dapyV20.definitions.transactions.TakeProfi	oan-	definition	s (oandapyV20 definitions accounts Accou	ıntFinancingMode
	norder	Reason	attribute), 66	maneingivious
attribute), 77	.	definition	is (oandapy V20.definitions.accounts.Positions)	on Aggregation Mod
				om iggiegatiomvioa
dapyV20.definitions.transactions.TrailingSt	wpros	definition	ason oute), 66 s (oandapy V20. definitions. instruments. Ca	ndlestickGranularity
attribute), 79			attribute), 67	
CLIENT_REQUEST (equipment of the dapy V20. definitions. transactions. Order Can	oan-	definition	s (oandapyV20.definitions instruments Price	ceComponents
attribute), 77	iceine?	25011-11-01	attribute), 68	
atu 10ute 1. / /				

definitions (oandapy V20.definitions.instruments.Weekly Aligand POINT (oandapy V20.endpoints.accounts.Account Summary
attribute), 68 attribute), 19
definitions (oandapy V20.definitions.orders.OrderPositionFilENDPOINT (oandapy V20.endpoints.instruments.InstrumentsCandles
attribute), 70 attribute), 20
definitions (oandapy V20.definitions.orders.OrderState at-
tribute), 70 attribute), 22
definitions (oandapy V20.definitions.orders.OrderType attribute), 69 ENDPOINT (oandapy V20.endpoints.instruments.InstrumentsPositionBook attribute), 25
definitions (oandapy V20.definitions.orders.TimeInForce ENDPOINT (oandapy V20.endpoints.orders.OrderCancel
attribute), 69 attribute), 29
definitions (oandapyV20.definitions.pricing.PriceStatus ENDPOINT (oandapyV20.endpoints.orders.OrderClientExtensions
attribute), 71 attribute), 30
definitions (oandapy V20.definitions.trades.TradeState at- ENDPOINT (oandapy V20.endpoints.orders.OrderCreate
tribute), 72 attribute), 31
definitions (oandapy V20.definitions.transactions.Funding ReENDPOINT (oandapy V20.endpoints.orders.OrderDetails
attribute), 74 attribute), 32
definitions (oandapy V20.definitions.transactions.LimitOrder IRN 10300 INT (oandapy V20.endpoints.orders.OrderList at-
attribute), 79 tribute), 33
definitions (oandapy V20.definitions.transactions.MarketIfTon N104PO104FReasonapy V20.endpoints.orders.OrderReplace
attribute), 74 attribute), 34
definitions (oandapy V20.definitions.transactions.MarketOrdEnReasonNT (oandapy V20.endpoints.orders.OrdersPending
attribute), 75 attribute), 35
definitions (oandapy V20.definitions.transactions.OrderCanc EIREROI NT (oandapy V20.endpoints.positions.OpenPositions
attribute), 78 attribute), 36
definitions (oandapy V20.definitions.transactions.OrderFillR END POINT (oandapy V20.endpoints.positions.PositionClose
attribute), 73 attribute), 38
definitions (oandapy V20. definitions. transactions. Stop Loss OEN FREQUOIT (oandapy V20. endpoints. positions. Position Details
attribute), 72 attribute), 40 definitions (oandapy V20.definitions.transactions.StopOrder RENIDPOINT (oandapy V20.endpoints.positions.PositionList
attribute), 75 attribute), 41
definitions (oandapy V20.definitions.transactions.TakeProfit@in@Red3%) (oandapy V20.endpoints.pricing.PricingInfo
attribute), 77 attribute), 42
definitions (oandapy V20.definitions.transactions.TrailingStopNovPOIdeFReasonapy V20.endpoints.pricing.PricingStream
attribute), 79 attribute), 45
definitions (oandapy V20.definitions.transactions.Transaction ENDPOINT (oandapy V20.endpoints.trades.OpenTrades
attribute), 76 attribute), 48
DELAYED_TRADE_CLOSE (oan- ENDPOINT (oandapy V20.endpoints.trades.TradeClientExtensions
dapyV20.definitions.transactions.MarketOrderReason attribute), 51
attribute), 74 ENDPOINT (oandapyV20.endpoints.trades.TradeClose
DIVISION_MIGRATION (oan-attribute), 52
dapyV20.definitions.transactions.FundingReason ENDPOINT (oandapyV20.endpoints.trades.TradeCRCDO
attribute), 74 attribute), 49
ENDPOINT (oandapyV20.endpoints.trades.TradeDetails attribute), 54
,,
ENDPOINT (oandapy V20.endpoints.accounts.AccountChartenDPOINT (oandapy V20.endpoints.trades.TradesList
attribute), 11 attribute), 55
ENDPOINT (oandapy V20.endpoints.accounts.AccountConfeyNDPOINT (oandapy V20.endpoints.transactions.TransactionDetails attribute), 13
ENDPOINT (oandapy V20.endpoints.accounts.AccountDetaENDPOINT (oandapy V20.endpoints.transactions.TransactionIDRange attribute), 14 attribute), 57
attribute), 14 attribute), 57 ENDPOINT (oandapyV20.endpoints.accounts.AccountInstrENDPOINT (oandapyV20.endpoints.transactions.TransactionList
attribute), 17 attribute), 59 ENDPOINT (oandapyV20.endpoints.accounts.AccountList ENDPOINT (oandapyV20.endpoints.transactions.TransactionsSinceID
attribute), 19 attribute), 60
uuiiouoj, 17

ENDPOINT (oandapyV20.endpoints.transactions.7	Transacti	ionsStrean		ose
attribute), 62			attribute), 38	
EXPECTED_STATUS	(oan-	EXPECT	ED_STATUS	(oan-
dapyV20.endpoints.accounts.AccountCh	anges		dapyV20.endpoints.positions.PositionDe	tails
attribute), 11			attribute), 40	
EXPECTED_STATUS			ED_STATUS	(oan-
dapyV20.endpoints.accounts.AccountCo	nfigurati	ion	dapyV20.endpoints.positions.PositionLis	t
attribute), 13			attribute), 41	
EXPECTED_STATUS	(oan-	EXPECT	ED_STATUS	(oan-
dapyV20.endpoints.accounts.AccountDe	etails		dapyV20.endpoints.pricing.PricingInfo	at-
attribute), 14			tribute), 42	
EXPECTED_STATUS	(oan-	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.accounts.AccountIns	strument	S	dapyV20.endpoints.pricing.PricingStream	n
attribute), 17			attribute), 45	
EXPECTED_STATUS	(oan-	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.accounts.AccountLis			dapyV20.endpoints.trades.OpenTrades	at-
attribute), 19			tribute), 48	
EXPECTED_STATUS	(oan-	EXPECT	TED STATUS	(oan-
dapyV20.endpoints.accounts.AccountSu	`		dapy V20.endpoints.trades.TradeClientEx	`
attribute), 19			attribute), 51	
EXPECTED_STATUS	(oan-	EXPECT	ED_STATUS	(oan-
dapyV20.endpoints.instruments.Instrume			dapyV20.endpoints.trades.TradeClose	at-
attribute), 20	omocune	1103	tribute), 52	ut
EXPECTED_STATUS	(oan-	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.instruments.Instrume			dapyV20.endpoints.trades.TradeCRCDO	(Oali-
attribute), 22	insoruc	IDOOK	attribute), 49	
EXPECTED_STATUS	(oan	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.instruments.Instrume				at-
	EIRSPOSI	JOHDOOK	= :	at-
attribute), 25	(EVDECT	tribute), 54	(
EXPECTED_STATUS	(oan-	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.orders.OrderCancel	at-		dapyV20.endpoints.trades.TradesList	at-
tribute), 29	,	EXPECT	tribute), 55	,
EXPECTED_STATUS			TED_STATUS	(oan-
dapyV20.endpoints.orders.OrderClientE	xtension	S	dapyV20.endpoints.transactions.Transact	ionDetails
attribute), 30	,		attribute), 56	,
EXPECTED_STATUS	*	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.orders.OrderCreate	at-		dapyV20.endpoints.transactions.Transact	ionIDRange
tribute), 31			attribute), 57	
EXPECTED_STATUS	(oan-	EXPECT	ED_STATUS	(oan-
dapyV20.endpoints.orders.OrderDetails	at-		dapyV20.endpoints.transactions.Transact	ionList
tribute), 32			attribute), 59	
EXPECTED_STATUS	(oan-	EXPECT	ED_STATUS	(oan-
dapyV20.endpoints.orders.OrderList attr	ribute),		dapyV20.endpoints.transactions.Transact	ionsSinceID
33			attribute), 60	
EXPECTED_STATUS	(oan-	EXPECT	TED_STATUS	(oan-
dapyV20.endpoints.orders.OrderReplace	;		dapyV20.endpoints.transactions.Transact	ionsStream
attribute), 34			attribute), 62	
EXPECTED_STATUS	(oan-	_		
dapyV20.endpoints.orders.OrdersPendin	g	F		
attribute), 35		FIFO VI	OLATION	(oan-
EXPECTED_STATUS	(oan-		dapy V20. definitions. transactions. Order C	`
dapyV20.endpoints.positions.OpenPositi			attribute), 78	
attribute), 36		FILLED		ite at-
EXPECTED_STATUS	(oan-		tribute). 70	··· ···

FOK	(oandapy V20. definitions. orders. Time In Force attribute), 69	Instrume	ntsOrderBook dapyV20.endpoir	(class nts.instrumer	in nts), 22	oan-
Friday	(o and apy V20. definitions. instruments. Weekly Alignmattribute), 67	m ēns trume	ntsPositionBook dapyV20.endpoir	(class nts.instrumer	in nts), 25	oan-
Fundi	ngReason (class in oan-dapyV20.definitions.transactions), 73	INSUFFI	CIENT_LIQUIDI dapyV20.definition		ons.Order((oan- CancelReason
G		INSUFFI	attribute), 78 [CIENT_MARGIN	J		(oan-
GFD	(oandapy V20.definitions.orders.TimeInForce attribute), 69	11 (15 01 1 1	dapyV20.definitionattribute), 78		ons.Order(`
granu	arity_to_time() (in module oan-	INTERN	AL_SERVER_ER			(oan-
СТС	dapyV20.contrib.generic), 88		dapyV20.definition attribute), 78	ons.transacti	ons.Order(CancelReason
GTC	(oandapy V20.definitions.orders.TimeInForce attribute), 69	invalid ((oandapyV20.defir	nitions.pricin	g.PriceSta	tus at-
GTD	(oandapyV20.definitions.orders.TimeInForce at-		tribute), 71			
	tribute), 69	IOC (o	andapyV20.definit	ions.orders.'	TimeInFor	ce at-
Н			tribute), 69			
	undapyV20.definitions.instruments.CandlestickGranu	l ılarity				
(attribute), 66	LIMIT	(oandapyV20	definitions.	orders.Ord	lerType
H12 (oandapyV20.definitions.instruments.CandlestickGran		attribute), 69	/20 4-E-:4:-	4	i OndonEillDoor
H2 (o:	attribute), 67 andapyV20.definitions.instruments.CandlestickGranu		attribute), 73	/ 20.deimino	ns.transact	tions.OrderFillReason
112 (0	attribute), 67			/20.definitio	ns.transact	tions.TransactionType
H3 (oa	andapyV20.definitions.instruments.CandlestickGranu		attribute), 76			
TT4 /	attribute), 67		ORDER_REJECT	ona transacti	ona Transa	(oan-
H4 (0	andapy V20. definitions. instruments. Candlestick Grant attribute), 67	ilarity	dapyV20.definition attribute), 76	ms.u ansacu	ons. mansa	ctionType
H6 (oa	andapyV20.definitions.instruments.CandlestickGranu	յլ երիր յitOrd	lerReason	(class	in	oan-
	attribute), 67		dapyV20.definition			
H8 (oa	andapyV20.definitions.instruments.CandlestickGranu	<u>ıllanın</u> yıtOrd	lerRequest dapyV20.contrib.	(class	in	oan-
HEAI	attribute), 67 DERS (oandapy V20.endpoints.accounts.AccountCon	fi LINKE D			,	(oan-
	attribute), 13		dapyV20.definition		ons.Order(`
HEAI	DERS (oandapy $V20$.endpoints.orders.OrderClientExt		attribute), 78			,
III: A I	attribute), 30 DERS (oandapyV20.endpoints.orders.OrderCreate	LOSING	_TAKE_PROFIT dapyV20.definition	one trancacti	ons Order((oan- CancelReason
пеаг	attribute), 31		attribute), 78	Jiis.ti aiisacti	ons.order	Janeenceason
HEAI	DERS (oandapy V20.endpoints.orders.OrderReplace attribute), 34	М	,,			
HEAI	DERS (oandapy V20.endpoints.positions.PositionClos	seM (oanda		.instruments	.Candlesti	ckGranularity
TIEAT	attribute), 38	. 3 & 1	attribute), 67		D : C	
HEAL	DERS (oandapy V20.endpoints.trades.TradeClientExt attribute), 51	ensa qosnda	apy V20.definitions attribute), 68	.instruments	.PriceCom	ponents
HEAI		M1 (oand	lapy V20.definition attribute), 67	s.instrumen	ts.Candles	ickGranularity
HEAI	DERS (oandapy V20.endpoints.trades.TradeCRCDO	M15 (oar	ndapyV20.definition	ons.instrume	nts.Candle	stickGranularity
	attribute), 49		attribute), 67			•
I			lapyV20.definition attribute), 67			
Instru	mentsCandles (class in oan-	M30 (oar	ndapyV20.definitio	ons.instrume	nts.Candle	stickGranularity
Inctm	dapyV20.endpoints.instruments), 20 mentsCandlesFactory() (in module oan-	M4 (oand	attribute), 67 lapyV20.definition	instrumen	ts Candles	tickGranularity
1115U U	danyV20 contrib factories) 87	MTT (Oalic	attribute), 67	is.iiisu uiiiCll	anuics	TOROTALIUIAIILY

```
M5 (oandapy V20.definitions.instruments.CandlestickGranullatanketOrderReason
                                                                                (class
                                                                                             in
                                                                                                      oan-
         attribute), 67
                                                               dapyV20.definitions.transactions), 74
MARGIN CALL ENTER
                                               (oan- MarketOrderRequest
                                                                                (class
                                                                                             in
                                                                                                      oan-
         dapyV20.definitions.transactions.TransactionType
                                                               dapyV20.contrib.requests), 90
                                                      MAXIMAL SIDE
         attribute), 76
                                                                                                     (oan-
MARGIN CALL EXIT
                                               (oan-
                                                               dapyV20.definitions.accounts.PositionAggregationMode
         dapyV20.definitions.transactions.TransactionType
                                                               attribute), 66
                                                      METHOD (oandapy V20.endpoints.accounts.AccountChanges
         attribute), 76
MARGIN CALL EXTEND
                                               (oan-
                                                               attribute), 11
         dapyV20.definitions.transactions.TransactionTypeMETHOD (oandapyV20.endpoints.accounts.AccountConfiguration
         attribute), 76
                                                               attribute), 13
MARGIN_CLOSEOUT
                                               (oan-METHOD (oandapy V20.endpoints.accounts.AccountDetails
         dapyV20.definitions.transactions.MarketOrderReason
                                                               attribute), 14
         attribute), 75
                                                      METHOD (oandapy V20.endpoints.accounts.AccountInstruments
MARKET (oandapy V20.definitions.orders.OrderType at-
                                                               attribute), 17
         tribute), 69
                                                      METHOD (oandapy V20.endpoints.accounts.AccountList
MARKET_HALTED
                                               (oan-
                                                               attribute), 19
         dapyV20.definitions.transactions.OrderCancelReal6ETHOD (oandapyV20.endpoints.accounts.AccountSummary
         attribute), 78
                                                               attribute), 19
MARKET IF TOUCHED
                                                      METHOD (oandapy V20.endpoints.instruments.InstrumentsCandles
                                               (oan-
                                                               attribute), 20
         dapyV20.definitions.orders.OrderType
                                                 at-
         tribute), 69
                                                      METHOD (oandapy V20.endpoints.instruments.InstrumentsOrderBook
MARKET_IF_TOUCHED_ORDER
                                                               attribute), 22
                                               (oan-
         dapyV20.definitions.transactions.OrderFillReasonMETHOD (oandapyV20.endpoints.instruments.InstrumentsPositionBook
         attribute), 73
                                                               attribute), 25
MARKET IF TOUCHED ORDER
                                               (oan- METHOD
                                                                 (oandapy V20.endpoints.orders.OrderCancel
         dapyV20.definitions.transactions.TransactionType
                                                               attribute), 29
         attribute), 76
                                                      METHOD (oandapy V20.endpoints.orders.OrderClientExtensions
MARKET_IF_TOUCHED_ORDER_REJECT
                                                               attribute), 30
                                               (oan-
         dapyV20.definitions.transactions.TransactionTypeMETHOD
                                                                  (oandapyV20.endpoints.orders.OrderCreate
         attribute), 76
                                                               attribute), 31
MARKET_ORDER
                                               (oan-
                                                     METHOD (oandapy V20.endpoints.orders.OrderDetails
         dapyV20.definitions.transactions.OrderFillReason
                                                               attribute), 32
         attribute), 73
                                                      METHOD (oandapy V20.endpoints.orders.OrderList at-
MARKET ORDER
                                                               tribute), 33
                                               (oan-
         dapyV20.definitions.transactions.TransactionTypeMETHOD (oandapyV20.endpoints.orders.OrderReplace
         attribute), 76
                                                               attribute), 34
MARKET_ORDER_DELAYED_TRADE_CLOSE
                                                      METHOD (oandapy V20. endpoints. orders. Orders Pending
         (oandapyV20.definitions.transactions.OrderFillReason
                                                               attribute), 35
         attribute), 73
                                                      METHOD (oandapy V20.endpoints.positions.OpenPositions
MARKET_ORDER_MARGIN CLOSEOUT
                                               (oan-
                                                               attribute), 36
         dapyV20.definitions.transactions.OrderFillReasonMETHOD (oandapyV20.endpoints.positions.PositionClose
         attribute), 73
                                                               attribute), 38
MARKET_ORDER_POSITION_CLOSEOUT
                                               (oan-
                                                      METHOD (oandapy V20.endpoints.positions.PositionDetails
        dapyV20.definitions.transactions.OrderFillReason
                                                               attribute), 40
         attribute), 73
                                                      METHOD (oandapy V20.endpoints.positions.PositionList
MARKET_ORDER_REJECT
                                                               attribute), 41
                                               (oan-
         dapyV20.definitions.transactions.TransactionTypeMETHOD
                                                                  (oandapy V20.endpoints.pricing.PricingInfo
                                                               attribute), 42
         attribute), 76
MARKET_ORDER_TRADE_CLOSE
                                               (oan- METHOD (oandapy V20.endpoints.pricing.PricingStream
         dapyV20.definitions.transactions.OrderFillReason
                                                               attribute), 45
                                                      METHOD
                                                                   (oandapyV20.endpoints.trades.OpenTrades
         attribute), 73
MarketIfTouchedOrderReason
                                (class
                                         in
                                                oan-
                                                               attribute), 48
         dapyV20.definitions.transactions), 74
                                                      METHOD (oandapy V20.endpoints.trades.TradeClientExtensions
```

attribute), 51	OPEN_TRADES_ALLOWED_EXCEE	
METHOD (oandapyV20.endpoints.trades.TradeClose at-	dapyV20.definitions.transaction	ons.OrderCancelReason
tribute), 52	attribute), 78	
	OpenPositions (class	in oan-
attribute), 49	dapyV20.endpoints.positions).	
METHOD (oandapyV20.endpoints.trades.TradeDetails attribute), 54	OpenTrades (class in oandapyV20.endpo ORDER_CANCEL	(oan-
METHOD (oandapyV20.endpoints.trades.TradesList at-	dapyV20.definitions.transaction	`
tribute), 55	attribute), 76	ms. Transaction Type
METHOD (oandapy V20.endpoints.transactions.Transaction		DDIFY (oan-
attribute), 56	dapyV20.definitions.transaction	
$METHOD\ (o and apy V20. endpoints. transactions. Transaction$	DRange attribute), 76	
attribute), 57	ORDER_CLIENT_EXTENSIONS_MO	
METHOD (oandapy V20.endpoints.transactions.Transaction		actions.TransactionType
attribute), 59	attribute), 76	
METHOD (oandapy V20.endpoints.transactions.Transaction	= -	.transactions.TransactionType
attribute), 60	attribute), 76	acinta andana) 20
METHOD (oandapy V20.endpoints.transactions.Transaction attribute), 62	OrderCancelReason (class	in oan-
MIGRATION (oandapy V20.definitions.transactions.OrderC		
attribute), 78	OrderClientExtensions (class	in oan-
MITOrderRequest (class in oan-	dapyV20.endpoints.orders), 29	
dapyV20.contrib.requests), 92	OrderCreate (class in oandapyV20.endp	
Monday (oandapy V20. definitions. instruments. Weekly Align		
attribute), 67	OrderFillReason (class	in oan-
NI	dapyV20.definitions.transaction	
N	OrderID (class in oandapyV20.types), 83	
$NET_SUM~(o and apy V20. definitions. accounts. Position Agg$	eralge dentitier (class in oandapy V20.ty)	pes), 83
attribute), 66	OrderList (class in oandapyV20.endpoin	
NO_FINANCING (oan-	OrderPositionFill (class	in oan-
dapyV20.definitions.accounts.AccountFinancingl	Tode dapy V20. definitions. orders), 7 Order Replace (class in oandapy V20. end	
attribute), 65 non_tradeable (oandapy V20.definitions.pricing.PriceStatus	OrderSpecifier (class in oandapy V20.cm)	
attribute), 71	OrdersPending (class in oandapyV20.e	
attribute), 71	35	1
0	OrderState (class in oandapyV20.definit	ions.orders), 69
oandapyV20.contrib.factories (module), 87	OrderType (class in oandapyV20.definit	ions.orders), 69
oandapy V20.contrib.generic (module), 88	D	
oandapy V20. definitions. accounts (module), 65	P	
oandapyV20.definitions.instruments (module), 66	PENDING (oandapyV20.definitions.o	orders.OrderState
oandapy V20.definitions.orders (module), 68	attribute), 70	
oandapyV20.definitions.pricing (module), 71	PENDING_ORDERS_ALLOWED_EX	
oandapyV20.definitions.trades (module), 71	dapyV20.definitions.transactio	ons.OrderCancelReason
oandapyV20.definitions.transactions (module), 72	attribute), 78	(oon
ON_FILL (oandapy V20.definitions.transactions.StopLossC	derneason_CLOSEOU1 dapyV20.definitions.transactio	(0an- ons MarketOrderReason
attribute), 72 ON_FILL (oandapy V20.definitions.transactions.TakeProfit	1.0	nis.WarketOrderReason
attribute), 77	POSITION_CLOSEOUT_FAILED	(oan-
ON_FILL (oandapy V20.definitions.transactions.TrailingSto		
attribute), 79	attribute), 78	
	POSITION_SIZE_EXCEEDED	(oan-
tribute), 72	dapyV20.definitions.transaction	ons.OrderCancelReason
OPEN_ONLY (oandapyV20.definitions.orders.OrderPositionattribute), 70	nFill attribute), 78	

PositionAggregationMode (class in oan-dapyV20.definitions.accounts), 66	S15 (oandapy V20.definitions.instruments.CandlestickGranularity attribute), 67
PositionClose (class in oandapyV20.endpoints.positions).	
38	attribute), 67
`	
dapyV20.contrib.requests), 93	attribute), 67
PositionDetails (class in oan-	, 1,
dapyV20.endpoints.positions), 40	attribute), 67
PositionList (class in oandapyV20.endpoints.positions)	
41	dapyV20.definitions.accounts.AccountFinancingMode
PriceComponents (class in oan-	
dapyV20.definitions.instruments), 68	secs2time() (in module oandapyV20.contrib.generic), 88
PriceStatus (class in oandapyV20.definitions.pricing), 71	SITE_MIGRATION (oan-
PriceValue (class in oandapyV20.types), 84	dapyV20.definitions.transactions.FundingReason
PricingInfo (class in oandapyV20.endpoints.pricing), 42	attribute), 74
PricingStream (class in oandapyV20.endpoints.pricing)	STOP (oandapyV20.definitions.orders.OrderType at-
45	tribute), 69
D	STOP_LOSS (oandapyV20.definitions.orders.OrderType
R	attribute), 69
REDUCE_FIRST (oan-	STOP_LOSS_ON_FILL_CLIENT_ORDER_ID_ALREADY_EXISTS
dapyV20.definitions.orders.OrderPositionFill	(oandapyV20.definitions.transactions.OrderCancelReason
attribute), 70	attribute), 78
REDUCE_ONLY (oan-	
dapyV20.definitions.orders.OrderPositionFill	(oandapy V20.definitions.transactions.OrderCancelReason
attribute), 70	attribute), 78
REOPEN (oandapy V20. definitions. transactions. Transactions.	
	dapyV20.definitions.transactions.OrderCancelReason
attribute), 76	
REPLACEMENT (oan-	
dapyV20.definitions.transactions.LimitOrderRe	
attribute), 79	dapyV20.definitions.transactions.OrderFillReason
REPLACEMENT (oan-	
dapyV20.definitions.transactions.MarketIfTouc	
attribute), 74	dapyV20.definitions.transactions.TransactionType
REPLACEMENT (oan-	
dapyV20.definitions.transactions.StopLossOrde	erRSTOR_LOSS_ORDER_REJECT (oan-
attribute), 72	dapyV20.definitions.transactions.TransactionType
REPLACEMENT (oan-	
dapyV20.definitions.transactions.StopOrderRea	aso STOP_ORDER (oandapy V20. definitions. transactions. Order Fill Reason
attribute), 75	attribute), 73
REPLACEMENT (oan-	STOP_ORDER (oandapy V20.definitions.transactions.TransactionType
dapyV20.definitions.transactions.TakeProfitOrc	derReason attribute), 76
attribute), 77	STOP_ORDER_REJECT (oan-
REPLACEMENT (oan-	
dapyV20.definitions.transactions.TrailingStopL	
attribute), 79	StopLossDetails (class in oandapyV20.contrib.requests),
request() (oandapyV20.API method), 8	100
request_params (oandapyV20.API attribute), 8	StopLossOrderReason (class in oan-
RESET_RESETTABLE_PL (oan-	
dapyV20.definitions.transactions.TransactionTy	/peStopLossOrderRequest (class in oan- dapyV20.contrib.requests), 94
attribute), 76	
S	StopOrderReason (class in oan-
	dapyV20.definitions.transactions), 75
S10(o and apy V20. definitions. instruments. Can dlestick Graduations and the state of the st	nustopOrderRequest (class in oan-
attribute), 67	dapyV20.contrib.requests), 95

STREAM (oandapyV20.endpoints.pricing.PricingStream attribute), 45		(oandapyV20.defi tribute), 71	nitions.pricing	.PriceStatu	ıs at-
STREAM (oandapy V20.endpoints.transactions.Transaction attribute), 62		ntExtensions dapyV20.endpoin	(class ts.trades), 51	in	oan-
Sunday (oandapy V20.definitions.instruments.Weekly Alignments.Weekly Alignments.Week				its.trades).	52
attribute), 68	TradeClos		(class	in	oan-
attribute), 00		dapyV20.contrib.1	`	111	Odii
T		CDO (class in o	-	dnointe tra	des)
		49	andapy v 20.cm	ироппъ.па	ides),
TAKE_PROFIT (oandapy V20.definitions.orders.OrderType) TradaData		nnyW20 andna	into tradas	5.4
attribute), 69		ails (class in oanda		mis.trades,	1, 34
TAKE_PROFIT_ON_FILL_CLIENT_ORDER_ID_ALRE	Albade HXI	SIRSS in oandapy v	20.types), 84	. 1) /	- ~
(oandapyV20.definitions.transactions.OrderCance	elikeassh1s	t (class in oandapy	y v 20.enapoini	s.trades),	71
attribute), 78		e (class in oandapy	y V20.definitio		
$TAKE_PROFIT_ON_FILL_GTD_TIMESTAMP_IN_PAS'$	ŢľRAILIN	G_STOP_LOSS			(oan-
(oandapyV20.definitions.transactions.OrderCance	elReason	dapy V20. definition	ons.orders.Ord	erType	at-
attribute), 78		tribute), 69			
TAKE_PROFIT_ON_FILL_LOSS (oan-					DER_ID_ALREADY_
dapyV20.definitions.transactions.OrderCancelRe	ason	(oandapyV20.defi	nitions.transac	ctions.Orde	erCancelReason
attribute), 78		attribute), 78			
TAKE_PROFIT_ORDER (oan-	TRAILIN	G_STOP_LOSS_	ON_FILL_G7	D_TIMES	STAMP_IN_PAST
dapyV20.definitions.transactions.OrderFillReason	n	(oandapyV20.defi	nitions.transac	ctions.Orde	erCancelReason
attribute), 73		attribute), 78			
TAKE_PROFIT_ORDER (oan-	TRAILIN	G_STOP_LOSS_	ORDER		(oan-
dapyV20.definitions.transactions.TransactionType		dapyV20.definitio		s.OrderFill	Reason
attribute), 76		attribute), 73			
TAKE_PROFIT_ORDER_REJECT (oan-		G_STOP_LOSS_	ORDER		(oan-
dapyV20.definitions.transactions.TransactionType		dapyV20.definitio			`
		attribute), 76	ins.cramsaction	o. Hunsuet	оптурс
attribute), 76		G_STOP_LOSS_	ORDER REI	FCT	(oan-
TakeProfitDetails (class in oandapyV20.contrib.requests),		dapyV20.definitio			`
101			nis.u ansacuon	s. mansacu	onType
TakeProfitOrderReason (class in oan-		attribute), 76	(alaga	i	000
dapyV20.definitions.transactions), 76	_	topLossDetails	(class	in	oan-
TakeProfitOrderRequest (class in oan-		dapyV20.contrib.i			
dapyV20.contrib.requests), 96	_	copLossOrderReas		in	oan-
terminate() (oandapy V20.endpoints.pricing.PricingStream		dapyV20.definitio			
method), 48		opLossOrderRequ		in	oan-
$terminate() \ (o and apy V20. endpoints. transactions. Transactio$	nsStream	dapyV20.contrib.i	requests), 97		
method), 63	Transactio		`	in	oan-
Thursday (oandapy V20. definitions. instruments. Weekly Alig		dapyV20.endpoin), 56	
attribute), 68		onIDRange	(class	in	oan-
TIME_IN_FORCE_EXPIRED (oan-		dapyV20.endpoin	ts.transactions), 57	
dapyV20.definitions.transactions.OrderCancelRea	a Tra nsactio	onList (cl	lass i	n	oan-
attribute), 78		dapyV20.endpoin	ts.transactions), 59	
TimeInForce (class in oandapyV20.definitions.orders), 68	Transactio	onsSinceID	(class	in	oan-
TRADE_CLIENT_EXTENSIONS_MODIFY (oan-		dapyV20.endpoin	ts.transactions), 60	
dapyV20.definitions.transactions.TransactionType			(class	in	oan-
attribute), 76		dapyV20.endpoin	•), 62	
TRADE_CLIENT_EXTENSIONS_MODIFY_REJECT	Transactio		_	n	oan-
		dapyV20.definitio			
(oandapy V20.definitions.transactions.Transaction		ER_FUNDS			(oan-
attribute), 76		dapyV20.definitio	ne transaction		`
TRADE_CLOSE (oan-		attribute), 76	nis.u ansacuon	s. mansacti	ontype
dapyV20.definitions.transactions.MarketOrderRe		ER_FUNDS_REJ	FСТ		(oan-
attribute), 75					`
		dapyV20.definitio	nis.u ansacuon	s. mansacu	onrype

```
attribute), 76
TRIGGERED (oandapy V20. definitions. orders. Order State
         attribute), 70
Tuesday (oandapy V20. definitions. instruments. Weekly Alignment
         attribute), 68
U
Units (class in oandapyV20.types), 84
V20Error (class in oandapyV20), 8
value (oandapyV20.types.AccountID attribute), 82
value (oandapyV20.types.AccountUnits attribute), 82
value (oandapy V20.types.ClientComment attribute), 82
value (oandapyV20.types.ClientID attribute), 82
value (oandapyV20.types.ClientTag attribute), 82
value (oandapyV20.types.DateTime attribute), 83
value (oandapyV20.types.OrderID attribute), 83
value (oandapyV20.types.OrderIdentifier attribute), 84
value (oandapyV20.types.OrderSpecifier attribute), 84
value (oandapyV20.types.PriceValue attribute), 84
value (oandapyV20.types.TradeID attribute), 84
value (oandapyV20.types.Units attribute), 84
W
W (oandapy V20.definitions.instruments.CandlestickGranularity
         attribute), 67
Wednesday (oandapy V20. definitions. instruments. Weekly Alignment
         attribute), 68
WeeklyAlignment
                          (class
                                        in
                                                   oan-
         dapyV20.definitions.instruments), 67
```