SAXO OpenAPI Documentation

Release 0.6.0

Feite Brekeveld

Sep 16, 2019

Contents:

This is the documention of the *saxo_openapi* python wrapper for the *OpenAPI* REST-API of SAXO Bank. Full documentation of the *OpenAPI* REST-API itself can be found here.

CHAPTER 1

Introduction

The *saxo_openapi* package offers an API to the SAXO OpenAPI 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 www.developer.saxo.

1.1 Install

Install the latest development version from github:

```
$ pip install git+https://github.com/hootnot/saxo_openapi.git
```

Before installing, consider using *virtual environments* to create isolated Python environments and install here.

CHAPTER 2

saxo_openapi

Top-level package for SAXO OpenAPI.

class saxo_openapi.**API**(*access_token*, *environment='simulation'*, *headers=None*, *request_params=None*)

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

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 OpenAPIError in case of HTTP response code >= 400

request_params

request_params property.

CHAPTER 3

saxo_openapi endpoints

3.1 saxo_openapi.endpoints.accounthistory

3.1.1 saxo_openapi.endpoints.accounthistory.accountvalues

AccountSummary

class saxo_openapi.endpoints.accounthistory.accountvalues.**AccountSummary**(*ClientKey*, *params=None*)

Get 'rolled up performance' for the accounts of specified client.

ENDPOINT = 'openapi/hist/v3/accountValues/{ClientKey}/'

EXPECTED_STATUS = 200

METHOD = 'GET'

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

Instantiate an AccountSummary request.

Parameters

- **ClientKey** (*string (required)*) – the ClientKey
- **params** (*dict (optional)*) – dict with parameters representing the queringstring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.accounthistory as ah
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClientKey = 'Cf4xZWiYL6W1nMKpygBLLA=='
>>> r = ah.accountvalues.AccountSummary(ClientKey=ClientKey)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:


```
{
  "Data": [
    {
      "AccountValue": 5028322.26953125,
      "AccountValueMonth": 1510.67004394531,
      "AccountValueYear": 0,
      "Key": "WMhrJRysCuCFlMbcGFvPVA==",
      "KeyType": "Account"
    },
    {
      "AccountValue": 1000,
      "AccountValueMonth": 0,
      "AccountValueYear": 0,
      "Key": "076n7v5YfRys3|tNojPVcA==",
      "KeyType": "Account"
    },
    {
      "AccountValue": 5028469.04129028,
      "AccountValueMonth": 1510.670043945311,
      "AccountValueYear": 0,
      "Key": "xwhUCDYh8X|pIwV|og2Qag==",
      "KeyType": "Client"
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.1.2 saxo_openapi.endpoints.accounthistory.historicalpositions

HistoricalPositions

class saxo_openapi.endpoints.accounthistory.historicalpositions.**HistoricalPositions**(*ClientKey*, *params*)

Get a list of historical positions for a specific account owned by a client. The required fields are ClientKey and either StandardPeriod or FromDate/ToDate.

ENDPOINT = 'openapi/hist/v3/positions/{ClientKey}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ClientKey*, *params*)
Instantiate a HistoricalPositions request.

Parameters

- **ClientKey** (*string (required)*) – the ClientKey
- **params** (*dict (required)*) – dict with parameters representing the queringstring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.accounthistory as ah
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClientKey = 'Cf4xZWiYL6W1nMKpygBLLA=='
>>> params =
        {
          "FromDate": "2019-03-01",
          "ToDate": "2019-03-10"
        }
```

```
>>> r = ah.historicalpositions.HistoricalPositions(ClientKey=ClientKey,
...                                                params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:


```
{
  "Data": [
    {
      "AccountId": "112209INET",
      "AccountValueEndOfDay": {
        "AccountBalance": 7526.17183,
        "CashTransfers": 0,
        "Date": "2016-07-19",
        "PositionsValue": -978.29753,
        "SecurityTransfers": 0,
        "TotalValue": 6547.8743
      },
      "Amount": -1,
      "AmountAccountValueCloseRatio": "2:1",
      "AmountAccountValueOpenRatio": "2:1",
      "ClosingAssetType": "CfdOnIndex",
      "ClosingTradeDate": "2016-07-19",
      "ClosingValueDate": "2016-07-19",
      "CopiedFrom": "1",
      "CorrelationType": "None",
      "Decimals": 2,
      "ExecutionTimeClose": "2016-07-19T07:25:37.000000Z",
      "ExecutionTimeOpen": "2016-07-18T10:38:06.000000Z",
      "FigureValue": 1,
      "InstrumentCcyToAccountCcyRateClose": 1.1020982542939,
      "InstrumentCcyToAccountCcyRateOpen": 1.11308229426434,
      "InstrumentSymbol": "GER30.I",
      "LongShort": {
        "PresentationValue": "Short",
        "Value": "Short"
      },
      "OpeningAssetType": "CfdOnIndex",
      "OpeningTradeDate": "2016-07-18",
      "OpeningValueDate": "2016-07-18",
      "PriceClose": 9998,
      "PriceGain": 0.004778021102926538,
      "PriceOpen": 10046,
      "PricePct": -0.4778021102926538,
      "ProfitLoss": 52.87,
```


(continues on next page)

(continued from previous page)

```
      "ProfitLossAccountValueFraction": 0.00807437613761156,
      "Uic": "1373",
      "ValueInAccountCurrencyClose": -11018.778346430412,
      "ValueInAccountCurrencyOpen": -11182.02472817956
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

3.1.3 saxo_openapi.endpoints.accounthistory.performance

AccountPerformance

class saxo_openapi.endpoints.accounthistory.performance.**AccountPerformance**(*ClientKey*, *params=None*)

Get a collection of performance metrics for a specific account. The account performance returns confidencial information that is only allowed to be viewed by the account owner / owners. The required fields are ClientKey and either StandardPeriod or FromDate/ToDate.

ENDPOINT = 'openapi/hist/v3/perf/{ClientKey}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ClientKey*, *params=None*)
Instantiate an AccountPerformance request.

Parameters

- **ClientKey** (*string (required)*) – the ClientKey
- **params** (*dict (optional)*) – dict with parameters representing the queringstring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.accounthistory as ah
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClientKey = 'Cf4xZWiYL6W1nMKpygBLLA=='
>>> params =
        {
          "FromDate": "2019-03-01",
          "ToDate": "2019-03-10"
        }
```

```
>>> r = ah.performance.AccountPerformance(ClientKey=ClientKey,
...                                       params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "AccountSummary": {
    "AverageTradeDurationInMinutes": 34260,
    "AverageTradesPerWeek": 48.2325728770595,
    "NumberOfDaysTraded": 1589,
    "NumberOfLongTrades": 5434,
    "NumberOfShortTrades": 5263,
    "TopTradedInstruments": [
      "DAX.I",
      "DJI.I",
      "EURUSD",
      "GBPUSD",
      "NAS100.I",
      "NOKSEK",
      "EURNOK",
      "SP500.I"
    ],
    "TotalReturnFraction": -0.9999963956455,
    "TradedInstruments": [
      "FxSpot",
      "Stock",
      "FxVanillaOption",
      "ContractFutures"
    ],
    "TradesTotalCount": 10697,
    "TradesWonCount": 6499,
    "WinFraction": 0.61
  },
  "Allocation": {
    "TradesPerAssetType": {
      "ClosedTradesAllocations": [
        {
          "AssetClassType": "Equity",
          "ReturnAttribution": 1,
          "TradeCount": 168,
          "TradePercent": 0.38009049773755654
        },
        {
          "AssetClassType": "Currency",
          "ReturnAttribution": 0.249937016456527,
          "TradeCount": 112,
          "TradePercent": 0.25339366515837103
        },
        {
          "AssetClassType": "Commodity",
          "ReturnAttribution": 0.5628789450009563,
          "TradeCount": 105,
          "TradePercent": 0.23755656108597284
        },
        {
          "AssetClassType": "Fixed Income",
          "ReturnAttribution": -0.013150856136249162,
          "TradeCount": 57,
          "TradePercent": 0.12895927601809956
        }
      ]
    },
```


(continues on next page)

(continued from previous page)


```
    "TradesPerInstrument": {
      "ClosedTradesAllocations": [
        {
          "AssetType": "ContractFutures",
          "InstrumentDescription": "30-Year U.S. Treasury Bond - Sep 2016",
          "InstrumentSymbol": "ZBU6",
          "InstrumentUic": 3626018,
          "ReturnAttribution": -0.15987101005684304,
          "TradeCount": 40,
          "TradePercent": 0.09049773755656108
        },
        {
          "AssetType": "FxSpot",
          "InstrumentDescription": "British Pound/US Dollar",
          "InstrumentSymbol": "GBPUSD",
          "InstrumentUic": 31,
          "ReturnAttribution": 0.14685155225185834,
          "TradeCount": 37,
          "TradePercent": 0.083710407239819
        }
      ]
    }
  },
  "BalancePerformance": {
    "AccountBalanceTimeSeries": [
      {
        "Date": "2016-03-28",
        "Value": 0
      },
      {
        "Date": "2016-03-29",
        "Value": 0
      }
    ],
    "AccountValueTimeSeries": [
      {
        "Date": "2016-03-28",
        "Value": 0
      },
      {
        "Date": "2016-03-29",
        "Value": 0
      }
    ],
    "MonthlyProfitLossTimeSeries": [
      {
        "Date": "2015-11-30",
        "Value": 0
      },
      {
        "Date": "2015-12-31",
        "Value": 0
      }
    ],
    "SecurityTransferTimeSeries": [
      {
        "Date": "2016-03-28",
```


(continues on next page)

(continued from previous page)


```
        "Value": 0
      },
      {
        "Date": "2016-03-29",
        "Value": 0
      }
    ],
    "YearlyProfitLossTimeSeries": [
      {
        "Date": "2015-12-31",
        "Value": 0
      },
      {
        "Date": "2016-12-31",
        "Value": 0
      },
      {
        "Date": "2017-12-31",
        "Value": 0
      }
    ]
  },
  "From": "2016-03-28",
  "InceptionDay": "2015-11-24",
  "LastTradeDay": "2017-03-27",
  "Thru": "2017-03-27",
  "TimeWeightedPerformance": {
    "AccumulatedTimeWeightedTimeSeries": [
      {
        "Date": "2016-03-25",
        "Value": 0
      }
    ],
    "MonthlyReturnTimeSeries": [
      {
        "Date": "2016-03-25",
        "Value": 0
      }
    ],
    "PerformanceFraction": -1,
    "PerformanceKeyFigures": {
      "ClosedTradesCount": 0,
      "DrawdownReport": {
        "Drawdowns": [
          {
            "DaysCount": 3,
            "DepthInPercent": 1,
            "FromDate": "2016-08-05",
            "ThruDate": "2016-08-08"
          }
        ],
        "MaxDaysInDrawdownFromTop10Drawdowns": 3
      },
      "LosingDaysFraction": 0.03,
      "MaxDrawDownFraction": 1,
      "ReturnFraction": -1,
      "SampledStandardDeviation": 0.0618018874919214,
```


(continues on next page)

(continued from previous page)


```
      "SharpeRatio": -0.952069751000777,
      "SortinoRatio": -0.0591710418985739
    },
    "YearlyReturnTimeSeries": [
      {
        "Date": "2016-03-25",
        "Value": 0
      }
    ]
  },
  "TotalCashBalance": 20226.02,
  "TotalCashBalancePerCurrency": [
    {
      "StringValue": "CAD",
      "Value": -491.707122366824
    }
  ],
  "TotalOpenPositionsValue": 29571.057,
  "TotalPositionsValuePerCurrency": [
    {
      "StringValue": "CAD",
      "Value": -491.707122366824
    }
  ],
  "TotalPositionsValuePerProductPerSecurity": [
    {
      "Description": "Abengoa SA - Warrants",
      "ProductName": "Shares",
      "Symbol": "LOCK - 1496:xxxx",
      "Value": 0
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.2 saxo_openapi.endpoints.chart

3.2.1 saxo_openapi.endpoints.chart.charts

ChartDataRemoveSubscription

class saxo_openapi.endpoints.chart.charts.**ChartDataRemoveSubscription**(*ContextId*, *ReferenceId*)

Removes subscriptions for the given reference id on this resource, and frees resources on the server.

ENDPOINT = 'openapi/chart/v1/charts/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a ChartDataRemoveSubscription request.

Parameters

- **ContextId** (*string (required)*) – the context id
- **ReferenceId** (*string (required)*) – the reference id

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.chart as ch
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ...
>>> ReferenceId = ...
>>> r = ch.charts.ChartDataRemoveSubscription(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

ChartDataRemoveSubscriptions

class saxo_openapi.endpoints.chart.charts.**ChartDataRemoveSubscriptions**(*ContextId*, *params=None*)
Removes all subscriptions for the current session on this resource, and frees all resources on the server.

ENDPOINT = 'openapi/chart/v1/charts/subscriptions/{ContextId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params=None*)
Instantiate a ChartDataRemoveSubscriptions request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.chart as ch
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> ContextId = ...
>>> r = ch.charts.ChartDataRemoveSubscriptions(ContextId,
...                                            params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

CreateChartDataSubscription

class saxo_openapi.endpoints.chart.charts.**CreateChartDataSubscription**(*data*)
Sets up a subscription and returns an initial snapshot of most recently completed samples specified by the parameters in the request.

Subsequent samples are delivered over the streaming channel. Most often a single new sample or sample update is delivered at a time, but when a sample closes, you will typcially get two samples: The now closed bar, and the bar just opening.

ENDPOINT = 'openapi/chart/v1/charts/subscriptions'

EXPECTED_STATUS = 201

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

METHOD = 'POST'

__init__(*data*)
Instantiate a CreateChartDataSubscription request.

Parameters **data** (*dict (required)*) – dict representing the data body, in this case an order spec.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.chart as ch
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "AssetType": "FxSpot",
            "Count": 2,
            "Horizon": 1,
            "Uic": 21
          },
          "ContextId": "20190830035501020",
          "Format": "application/json",
          "ReferenceId": "UIC_21",
          "RefreshRate": 1000
        }
```

```
>>> r = ch.charts.CreateChartDataSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```


```
{
  "ContextId": "20190830035501020",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "UIC_21",
  "RefreshRate": 1000,
  "Snapshot": {
    "Data": [
      {
        "CloseAsk": 1.10631,
        "CloseBid": 1.10611,
        "HighAsk": 1.10636,
        "HighBid": 1.10616,
        "LowAsk": 1.10631,
        "LowBid": 1.10611,
        "OpenAsk": 1.10632,
        "OpenBid": 1.10612,
        "Time": "2019-09-09T16:54:00.000000Z"
      },
      {
        "CloseAsk": 1.10631,
        "CloseBid": 1.10611,
        "HighAsk": 1.10632,
        "HighBid": 1.10612,
        "LowAsk": 1.10631,
        "LowBid": 1.10611,
        "OpenAsk": 1.10632,
        "OpenBid": 1.10612,
        "Time": "2019-09-09T16:55:00.000000Z"
      }
    ],
    "DataVersion": 1592272523
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

GetChartData

class saxo_openapi.endpoints.chart.charts.**GetChartData**(*params*)
Return chart data as specified by request parameters.

ENDPOINT = 'openapi/chart/v1/charts'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a GetChartData request.

Parameters **params** (*dict (required)*) – dict representing the request parameters. Required in params: AssetType, Horizon and Uic

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.chart as chart
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "AssetType": "FxSpot",
          "Horizon": 60,
          "Count": 24,
          "Uic": 23
        }
```

```
>>> r = chart.charts.GetChartData(params=params)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "Data": [
    {
      "CloseAsk": 1.6321,
      "CloseBid": 1.6302,
      "HighAsk": 1.633,
      "HighBid": 1.63091,
      "LowAsk": 1.6316,
      "LowBid": 1.62974,
      "OpenAsk": 1.63237,
      "OpenBid": 1.6306,
      "Time": "2019-09-05T21:00:00.000000Z"
    },
    {
      "CloseAsk": 1.63115,
      "CloseBid": 1.63035,
      "HighAsk": 1.63353,
      "HighBid": 1.63219,
      "LowAsk": 1.63092,
      "LowBid": 1.62988,
      "OpenAsk": 1.6321,
      "OpenBid": 1.6303,
      "Time": "2019-09-05T22:00:00.000000Z"
    },
    {
      "CloseAsk": 1.63103,
      "CloseBid": 1.63023,
      "HighAsk": 1.63146,
      "HighBid": 1.63066,
      "LowAsk": 1.63049,
      "LowBid": 1.62969,
      "OpenAsk": 1.63114,
      "OpenBid": 1.63034,
      "Time": "2019-09-05T23:00:00.000000Z"
    },
```

(continues on next page)

(continued from previous page)


```
    {
      "CloseAsk": 1.63072,
      "CloseBid": 1.62992,
      "HighAsk": 1.63148,
      "HighBid": 1.63068,
      "LowAsk": 1.63056,
      "LowBid": 1.62976,
      "OpenAsk": 1.63103,
      "OpenBid": 1.63023,
      "Time": "2019-09-06T00:00:00.000000Z"
    },
    {
      "CloseAsk": 1.63125,
      "CloseBid": 1.63045,
      "HighAsk": 1.63157,
      "HighBid": 1.63077,
      "LowAsk": 1.63063,
      "LowBid": 1.62983,
      "OpenAsk": 1.6307,
      "OpenBid": 1.6299,
      "Time": "2019-09-06T01:00:00.000000Z"
    },
    {
      "CloseAsk": 1.63176,
      "CloseBid": 1.63096,
      "HighAsk": 1.63211,
      "HighBid": 1.63131,
      "LowAsk": 1.63072,
      "LowBid": 1.62992,
      "OpenAsk": 1.63125,
      "OpenBid": 1.63045,
      "Time": "2019-09-06T02:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62897,
      "CloseBid": 1.62817,
      "HighAsk": 1.63182,
      "HighBid": 1.63102,
      "LowAsk": 1.62897,
      "LowBid": 1.62817,
      "OpenAsk": 1.63178,
      "OpenBid": 1.63098,
      "Time": "2019-09-06T03:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62973,
      "CloseBid": 1.62893,
      "HighAsk": 1.62981,
      "HighBid": 1.62901,
      "LowAsk": 1.62854,
      "LowBid": 1.62774,
      "OpenAsk": 1.62896,
      "OpenBid": 1.62816,
      "Time": "2019-09-06T04:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62998,
```


(continues on next page)

(continued from previous page)


```
      "CloseBid": 1.62918,
      "HighAsk": 1.63058,
      "HighBid": 1.62978,
      "LowAsk": 1.62831,
      "LowBid": 1.62751,
      "OpenAsk": 1.62974,
      "OpenBid": 1.62894,
      "Time": "2019-09-06T05:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62696,
      "CloseBid": 1.62616,
      "HighAsk": 1.63084,
      "HighBid": 1.63004,
      "LowAsk": 1.62695,
      "LowBid": 1.62614,
      "OpenAsk": 1.62999,
      "OpenBid": 1.62919,
      "Time": "2019-09-06T06:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62529,
      "CloseBid": 1.62449,
      "HighAsk": 1.62772,
      "HighBid": 1.62692,
      "LowAsk": 1.62403,
      "LowBid": 1.62323,
      "OpenAsk": 1.62695,
      "OpenBid": 1.62615,
      "Time": "2019-09-06T07:00:00.000000Z"
    },
    {
      "CloseAsk": 1.6244,
      "CloseBid": 1.6236,
      "HighAsk": 1.62576,
      "HighBid": 1.62496,
      "LowAsk": 1.62252,
      "LowBid": 1.62172,
      "OpenAsk": 1.62528,
      "OpenBid": 1.62448,
      "Time": "2019-09-06T08:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62616,
      "CloseBid": 1.62536,
      "HighAsk": 1.62839,
      "HighBid": 1.62759,
      "LowAsk": 1.62362,
      "LowBid": 1.62282,
      "OpenAsk": 1.62436,
      "OpenBid": 1.62356,
      "Time": "2019-09-06T09:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62682,
      "CloseBid": 1.62602,
      "HighAsk": 1.62782,
```


(continues on next page)

(continued from previous page)


```
      "HighBid": 1.62702,
      "LowAsk": 1.62557,
      "LowBid": 1.62477,
      "OpenAsk": 1.62616,
      "OpenBid": 1.62536,
      "Time": "2019-09-06T10:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62839,
      "CloseBid": 1.62759,
      "HighAsk": 1.62898,
      "HighBid": 1.62818,
      "LowAsk": 1.62582,
      "LowBid": 1.62502,
      "OpenAsk": 1.6268,
      "OpenBid": 1.626,
      "Time": "2019-09-06T11:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62347,
      "CloseBid": 1.62267,
      "HighAsk": 1.63078,
      "HighBid": 1.62894,
      "LowAsk": 1.6222,
      "LowBid": 1.62079,
      "OpenAsk": 1.62839,
      "OpenBid": 1.62759,
      "Time": "2019-09-06T12:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62287,
      "CloseBid": 1.62207,
      "HighAsk": 1.62688,
      "HighBid": 1.62607,
      "LowAsk": 1.62234,
      "LowBid": 1.62155,
      "OpenAsk": 1.62346,
      "OpenBid": 1.62266,
      "Time": "2019-09-06T13:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62272,
      "CloseBid": 1.62192,
      "HighAsk": 1.62351,
      "HighBid": 1.6227,
      "LowAsk": 1.62093,
      "LowBid": 1.62012,
      "OpenAsk": 1.62287,
      "OpenBid": 1.62207,
      "Time": "2019-09-06T14:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62005,
      "CloseBid": 1.61925,
      "HighAsk": 1.62339,
      "HighBid": 1.62259,
      "LowAsk": 1.61979,
```


(continues on next page)

(continued from previous page)


```
      "LowBid": 1.619,
      "OpenAsk": 1.62271,
      "OpenBid": 1.62191,
      "Time": "2019-09-06T15:00:00.000000Z"
    },
    {
      "CloseAsk": 1.61863,
      "CloseBid": 1.61783,
      "HighAsk": 1.62038,
      "HighBid": 1.61958,
      "LowAsk": 1.61848,
      "LowBid": 1.61767,
      "OpenAsk": 1.62005,
      "OpenBid": 1.61925,
      "Time": "2019-09-06T16:00:00.000000Z"
    },
    {
      "CloseAsk": 1.61971,
      "CloseBid": 1.61891,
      "HighAsk": 1.62004,
      "HighBid": 1.61924,
      "LowAsk": 1.6174,
      "LowBid": 1.6166,
      "OpenAsk": 1.6186,
      "OpenBid": 1.6178,
      "Time": "2019-09-06T17:00:00.000000Z"
    },
    {
      "CloseAsk": 1.61992,
      "CloseBid": 1.61912,
      "HighAsk": 1.62017,
      "HighBid": 1.61937,
      "LowAsk": 1.61818,
      "LowBid": 1.61738,
      "OpenAsk": 1.61971,
      "OpenBid": 1.61891,
      "Time": "2019-09-06T18:00:00.000000Z"
    },
    {
      "CloseAsk": 1.62072,
      "CloseBid": 1.61992,
      "HighAsk": 1.62083,
      "HighBid": 1.62005,
      "LowAsk": 1.61969,
      "LowBid": 1.61889,
      "OpenAsk": 1.61993,
      "OpenBid": 1.61913,
      "Time": "2019-09-06T19:00:00.000000Z"
    },
    {
      "CloseAsk": 1.61933,
      "CloseBid": 1.61732,
      "HighAsk": 1.6212,
      "HighBid": 1.62041,
      "LowAsk": 1.61861,
      "LowBid": 1.61683,
      "OpenAsk": 1.62073,
```


(continues on next page)

(continued from previous page)

```
      "OpenBid": 1.61993,
      "Time": "2019-09-06T20:00:00.000000Z"
    }
  ],
  "DataVersion": 1715815481
}
```

expected_status

response
response - get the response of the request.

status_code

3.3 saxo_openapi.endpoints.eventnotificationservices

3.3.1 saxo_openapi.endpoints.eventnotificationservices

CreateSubscriptionForClientEvents

class saxo_openapi.endpoints.eventnotificationservices.clientactivities.**CreateSubscriptionF**
Set up an active subscription to listen client events.

ENDPOINT = 'openapi/ens/v1/activities/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a CreateSubscriptionForClientEvents request.

Parameters **data** (*dict (required)*) – dict representing the data body, the subscription specification

body example:

```
data =
    {
      "Arguments": {
        "AccountKey": "mroYddvgiGqqudzBPn8daA==",
        "Activities": [
          "AccountFundings",
          "MarginCalls",
          "Orders",
          "Positions"
        ],
        "ClientKey": "URpoxLBgX2I33Af3IhCvHg==",
        "FieldGroups": [
          "DisplayAndFormat",
          "ExchangeInfo"
        ],
        "FromDateTime": "2015-02-02T01:02:03Z",
        "IncludeSubAccounts": false,
        "SequenceId": "37456"
      },
```

(continues on next page)

(continued from previous page)

```
    "ContextId": "Context_20190503051756665",
    "ReferenceId": "M_344",
    "RefreshRate": 0
}
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.eventnotificationservices as ens
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data = ...
>>> r = ens.clientactivities.CreateSubscriptionForClientEvents(data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "ContextId": "Context_20190503051756665",
  "InactivityTimeout": 30,
  "ReferenceId": "M_344",
  "RefreshRate": 0,
  "Snapshot": {
    "Data": []
  },
  "State": "Active"
}
```

expected_status

response
response - get the response of the request.

status_code

GetActivities

class saxo_openapi.endpoints.eventnotificationservices.clientactivities.**GetActivities**(*param*
Return a list of activities specified by the parameters in the request.

ENDPOINT = 'openapi/ens/v1/activities'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a GetActivities request.

Parameters **params** (*dict (required)*) – dict with querystring parameters

params example:

```
params =
    {
      "Activities": "Positions",
      "AccountKey": "fOA0tvOyQqW2aHpWi9P5bw==",
      "ClientKey": "fOA0tvOyQqW2aHpWi9P5bw=="
    }
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.eventnotificationservices as ens
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = ens.clientactivities.GetActivities(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "__count": 7,
  "Data": [
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:04.346666Z",
      "ActivityType": "Positions",
      "Amount": 89000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "ClientId": "9300675",
      "Commission": 5.08,
      "ConversionRate": 0.876035,
      "CorrelationKey": "290088ee-c30b-43e0-bb31-f54379950c25",
      "CurrencyCode": "CHF",
      "ExecutionTime": "2019-05-07T11:52:04.344981Z",
      "OpenPrice": 1.14166,
      "OpenSpot": 1.14166,
      "PositionEvent": "New",
      "PositionId": "219378806",
      "PriceType": "Amount",
      "SequenceId": "840526",
      "SourceOrderId": "77283801",
      "SpotDate": "2019-05-09",
      "Symbol": "EURCHF",
      "TotalCost": 5.08,
      "Uic": 14,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    },
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:04.836666Z",
      "ActivityType": "Positions",
      "Amount": 142000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "ClientId": "9300675",
      "Commission": 5.09,
      "ConversionRate": 0.876035,
      "CorrelationKey": "2f3b72b6-4bc8-4d35-a4bc-8ca092586f81",
      "CurrencyCode": "CHF",
      "ExecutionTime": "2019-05-07T11:52:04.833449Z",
      "OpenPrice": 0.71691,
      "OpenSpot": 0.71691,
      "PositionEvent": "New",
      "PositionId": "219378808",
```


(continues on next page)

(continued from previous page)


```
      "PriceType": "Amount",
      "SequenceId": "840529",
      "SourceOrderId": "77283802",
      "SpotDate": "2019-05-09",
      "Symbol": "AUDCHF",
      "TotalCost": 5.09,
      "Uic": 5027,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    },
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:06.470000Z",
      "ActivityType": "Positions",
      "Amount": 76000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "ClientId": "9300675",
      "Commission": 5.07,
      "ConversionRate": 0.876035,
      "CorrelationKey": "f77bf761-380c-49b9-8027-93281f3fa321",
      "CurrencyCode": "CHF",
      "ExecutionTime": "2019-05-07T11:52:06.469006Z",
      "OpenPrice": 1.33438,
      "OpenSpot": 1.33438,
      "PositionEvent": "New",
      "PositionId": "219378812",
      "PriceType": "Amount",
      "SequenceId": "840532",
      "SourceOrderId": "77283803",
      "SpotDate": "2019-05-09",
      "Symbol": "GBPCHF",
      "TotalCost": 5.07,
      "Uic": 24,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    },
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:06.930000Z",
      "ActivityType": "Positions",
      "Amount": 152000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "ClientId": "9300675",
      "Commission": 5.12,
      "ConversionRate": 0.876035,
      "CorrelationKey": "f9cc8f67-b48e-490d-bc0e-811d2cd2dc8a",
      "CurrencyCode": "CHF",
      "ExecutionTime": "2019-05-07T11:52:06.929444Z",
      "OpenPrice": 0.67363,
      "OpenSpot": 0.67363,
      "PositionEvent": "New",
      "PositionId": "219378814",
      "PriceType": "Amount",
      "SequenceId": "840535",
      "SourceOrderId": "77283804",
```


(continues on next page)

(continued from previous page)


```
      "SpotDate": "2019-05-09",
      "Symbol": "NZDCHF",
      "TotalCost": 5.12,
      "Uic": 34,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    },
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:07.273333Z",
      "ActivityType": "Positions",
      "Amount": 102000,
      "AssetType": "FxSpot",
      "BuySell": "Sell",
      "ClientId": "9300675",
      "Commission": 553,
      "ConversionRate": 0.0080785,
      "CorrelationKey": "0f0ee8ff-da82-4945-911c-5e232749d8a0",
      "CurrencyCode": "JPY",
      "ExecutionTime": "2019-05-07T11:52:07.271770Z",
      "OpenPrice": 108.419,
      "OpenSpot": 108.419,
      "PositionEvent": "New",
      "PositionId": "219378816",
      "PriceType": "Amount",
      "SequenceId": "840538",
      "SourceOrderId": "77283805",
      "SpotDate": "2019-05-09",
      "Symbol": "CHFJPY",
      "TotalCost": 553,
      "Uic": 8,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    },
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:09.190000Z",
      "ActivityType": "Positions",
      "Amount": 135000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "ClientId": "9300675",
      "Commission": 5.12,
      "ConversionRate": 0.876085,
      "CorrelationKey": "9b29a7b7-8e5f-4b53-b9d0-91dbeb881a95",
      "CurrencyCode": "CHF",
      "ExecutionTime": "2019-05-07T11:52:09.187595Z",
      "OpenPrice": 0.75825,
      "OpenSpot": 0.75825,
      "PositionEvent": "New",
      "PositionId": "219378832",
      "PriceType": "Amount",
      "SequenceId": "840541",
      "SourceOrderId": "77283806",
      "SpotDate": "2019-05-09",
      "Symbol": "CADCHF",
      "TotalCost": 5.12,
```


(continues on next page)

(continued from previous page)


```
      "Uic": 5,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    },
    {
      "AccountId": "9300675",
      "ActivityTime": "2019-05-07T11:52:12.400000Z",
      "ActivityType": "Positions",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "ClientId": "9300675",
      "Commission": 5.1,
      "ConversionRate": 0.876065,
      "CorrelationKey": "4eeb4738-b1b9-4b36-bd14-31113fa77a2d",
      "CurrencyCode": "CHF",
      "ExecutionTime": "2019-05-07T11:52:12.398657Z",
      "OpenPrice": 1.0205,
      "OpenSpot": 1.0205,
      "PositionEvent": "New",
      "PositionId": "219379050",
      "PriceType": "Amount",
      "SequenceId": "840544",
      "SourceOrderId": "77283807",
      "SpotDate": "2019-05-09",
      "Symbol": "USDCHF",
      "TotalCost": 5.1,
      "Uic": 39,
      "UserId": 9300675,
      "ValueDate": "2019-05-09"
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

RemoveSubscription

class saxo_openapi.endpoints.eventnotificationservices.clientactivities.**RemoveSubscription**

Remove subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/ens/v1/activities/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a RemoveSubscription request.

Parameters

- **ContextId** (*string (required)*) – the ContextId of the subscription
- **ReferenceId** (*string (required)*) – the ReferenceId of the subscription

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.eventnotificationservices as ens
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ...
>>> ReferenceId = ...
>>> r = ens.clientactivities.RemoveSubscription(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

RemoveSubscriptions

class saxo_openapi.endpoints.eventnotificationservices.clientactivities.**RemoveSubscriptions**

Remove multiple/all subscriptions for the current session on this resource and free all resources on the server.

ENDPOINT = 'openapi/ens/v1/activities/subscriptions/{ContextId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params=None*)
Instantiate a RemoveSubscriptions request.

Parameters

- **ContextId** (*string (required)*) – the ContextId of the subscription
- **params** (*dict (optional)*) – dict with querystring parameters

params example:

```
params =
    {
      "Tag": "PAGE2"
    }
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.eventnotificationservices as ens
>>> import json
>>> client = saxo_openapi.API(access_token=...)
```

(continues on next page)

(continued from previous page)

```
>>> ContextId = ...
>>> r = ens.clientactivities.RemoveSubscriptions(ContextId,
...                                              params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.4 saxo_openapi.endpoints.portfolio

3.4.1 saxo_openapi.endpoints.portfolio.accountgroups

AccountGroupDetails

class saxo_openapi.endpoints.portfolio.accountgroups.**AccountGroupDetails**(*AccountGroupKey*, *params*)

Get details about a single account group.

ENDPOINT = 'openapi/port/v1/accountgroups/{AccountGroupKey}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*AccountGroupKey*, *params*)
Instantiate an AccountGroupDetails request.

Parameters

- **AccountGroupKey** (*string (required)*) – the accountGroupKey
- **params** (*dict (required)*) – dict with querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> AccountGroupKey = '...'
>>> r = pf.accountgroups.AccountGroupDetails(
...             AccountGroupKey=AccountGroupKey,
...             params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "Data": []
}
```

expected_status

response
response - get the response of the request.

status_code

AccountGroupUpdate

class saxo_openapi.endpoints.portfolio.accountgroups.**AccountGroupUpdate**(*AccountGroupKey*, *params*, *data*)

Update account group settings. Particularly the account group AccountValueProtectionLimit.

ENDPOINT = 'openapi/port/v1/accountgroups/{AccountGroupKey}'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*AccountGroupKey*, *params*, *data*)
Instantiate an AccountGroupUpdate request.

Parameters

- **AccountGroupKey** (*string (required)*) – the accountGroupKey
- **params** (*dict (required)*) – dict with attributes
- **data** (*dict (required)*) – dict with attributes

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> data =
        {
          "AccountValueProtectionLimit": 100000.0
        }
```

```
>>> r = pf.accountgroups.AccountGroupUpdate(
...             AccountGroupKey=AccountGroupKey,
...             params=params,
...             data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

expected_status

response
response - get the response of the request.

status_code

AccountGroupsList

class saxo_openapi.endpoints.portfolio.accountgroups.**AccountGroupsList**(*params*)
Get a list of all account groups used by the specified client.

ENDPOINT = 'openapi/port/v1/accountgroups/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate an AccountGroupsList request.

Parameters **params** (*dict (required)*) – dict with querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.accountgroups.AccountGroupsList(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "Data": []
}
```

expected_status

response
response - get the response of the request.

status_code

AccountGroupsMe

class saxo_openapi.endpoints.portfolio.accountgroups.**AccountGroupsMe**(*params=None*)
Get all accounts groups under a particular client to which the logged in user belongs.

ENDPOINT = 'openapi/port/v1/accountgroups/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate an AccountGroupsMe request.

Parameters **params** (*dict (optional)*) – dict with querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.accountgroups.AccountGroupsMe(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "Data": []
}
```

expected_status

response
response - get the response of the request.

status_code

3.4.2 saxo_openapi.endpoints.portfolio.accounts

AccountDetails

class saxo_openapi.endpoints.portfolio.accounts.**AccountDetails**(*AccountKey*)
Get details about a single account.

ENDPOINT = 'openapi/port/v1/accounts/{AccountKey}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*AccountKey*)
Instantiate an AccountDetails request.

Parameters **AccountKey** (*string (required)*) – Account key to perform the request on.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> AccountKey = 'f4xZWiYL6W1nMKpygBLLA=='
>>> r = pf.accounts.AccountDetails(AccountKey=AccountKey)
>>> rv = client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "CreationDate": "2019-02-02T10:47:42.313000Z",
  "CanUseCashPositionsAsMarginCollateral": true,
  "AccountType": "Normal",
  "CurrencyDecimals": 2,
  "IsCurrencyConversionAtSettlementTime": true,
  "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "IsShareable": false,
  "IndividualMargining": false,
  "DirectMarketAccess": false,
  "LegalAssetTypes": [
    "FxSpot",
    "FxForwards",
    "FxVanillaOption",
    "FxKnockInOption",
    "FxKnockOutOption",
    "FxOneTouchOption",
    "FxNoTouchOption"
  ],
  "AccountGroupKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "Active": true,
  "Sharing": [
    "NoSharing"
  ],
  "IsMarginTradingAllowed": true,
  "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "SupportsAccountValueProtectionLimit": false,
  "UseCashPositionsAsMarginCollateral": true,
  "ClientId": "9226397",
  "IsTrialAccount": true,
  "AccountId": "9226397",
  "Currency": "EUR"
}
```


expected_status

response
response - get the response of the request.

status_code

AccountListByClient

class saxo_openapi.endpoints.portfolio.accounts.**AccountListByClient**(*params*)
Get all accounts owned by the specified client.

ENDPOINT = 'openapi/port/v1/accounts/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate an AccountListByClient request.

Parameters **params** (*dict (required)*) – params must contain at least the ClientKey

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.accounts.AccountListByClient(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "Data": [
    {
      "IsMarginTradingAllowed": true,
      "DirectMarketAccess": false,
      "AccountId": "9226397",
      "ClientId": "9226397",
      "CreationDate": "2019-02-02T10:47:42.313000Z",
      "SupportsAccountValueProtectionLimit": false,
      "CurrencyDecimals": 2,
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "IsTrialAccount": true,
      "Sharing": [
        "NoSharing"
      ],
      "IsCurrencyConversionAtSettlementTime": true,
      "AccountType": "Normal",
      "Active": true,
      "IndividualMargining": false,
      "CanUseCashPositionsAsMarginCollateral": true,
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "AccountGroupKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Currency": "EUR",
      "IsShareable": false,
      "UseCashPositionsAsMarginCollateral": true,
      "LegalAssetTypes": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption",
        "FxOneTouchOption",
        "FxNoTouchOption"
      ]
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

AccountReset

class saxo_openapi.endpoints.portfolio.accounts.**AccountReset**(*AccountKey*, *data*)

Reset the trial account. Cannot be used in live environment.

ENDPOINT = 'openapi/port/v1/accounts/{AccountKey}/reset'

EXPECTED_STATUS = 204

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*AccountKey*, *data*)

Instantiate an AccountReset request.

Parameters

- **AccountKey** (*string (required)*) – the AccountKey
- **data** (*dict (required)*) – dict representing the requestbody.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> data = {'NewBalance': '1000000'}
>>> AccountKey = '...'
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.accounts.AccountReset(AccountKey=AccountKey, data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response

response - get the response of the request.

status_code

AccountUpdate

class saxo_openapi.endpoints.portfolio.accounts.**AccountUpdate**(*AccountKey*, *data*)

Update account details. Particularly the user account shield value, the benchmark instrument or the account display name.

ENDPOINT = 'openapi/port/v1/accounts/{AccountKey}'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*AccountKey*, *data*)

Instantiate an AccountUpdate request.

Parameters

- **AccountKey** (*string (required)*) – the AccountKey
- **data** (*dict (required)*) – dict representing the requestbody.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "DisplayName": "MyTestName"
        }
```

```
>>> AccountKey = '...'
>>> r = pf.account.AccountUpdate(AccountKey=AccountKey, data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response
response - get the response of the request.

status_code

AccountsMe

class saxo_openapi.endpoints.portfolio.accounts.**AccountsMe**(*params=None*)
Get all accounts under a particular client to which the logged in user belongs.

ENDPOINT = 'openapi/port/v1/accounts/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate an AccountsMe request.

Parameters **params** (*dict (optional)*) – optional querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.accounts.AccountsMe(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Data": [
    {
```

(continues on next page)

(continued from previous page)


```
      "ClientId": "9226397",
      "DirectMarketAccess": false,
      "IsTrialAccount": true,
      "AccountType": "Normal",
      "UseCashPositionsAsMarginCollateral": true,
      "CanUseCashPositionsAsMarginCollateral": true,
      "LegalAssetTypes": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption",
        "FxOneTouchOption",
        "FxNoTouchOption"
      ],
      "SupportsAccountValueProtectionLimit": false,
      "IndividualMargining": false,
      "IsShareable": false,
      "Sharing": [
        "NoSharing"
      ],
      "CreationDate": "2019-02-02T10:47:42.313000Z",
      "IsCurrencyConversionAtSettlementTime": true,
      "Active": true,
      "CurrencyDecimals": 2,
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "AccountGroupKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "AccountId": "9226397",
      "Currency": "EUR",
      "IsMarginTradingAllowed": true
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

SubscriptionCreate

class saxo_openapi.endpoints.portfolio.accounts.**SubscriptionCreate**(*data*)
Set up a subscription and returns an initial snapshot containing a list of accounts as specified by the parameters in the request.

ENDPOINT = 'openapi/port/v1/accounts/subscriptions/'

EXPECTED_STATUS = 201

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

METHOD = 'POST'

__init__(*data*)
Instantiate a SubscriptionCreate request.

Parameters **data** (*dict (required)*) – dict representing the requestbody.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1551455553043",
          "ReferenceId": "I_213"
        }
```

```
>>> r = pf.accounts.SubscriptionCreate(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "ContextId": "explorer_1551455553043",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "I_213",
  "RefreshRate": 5000,
  "Snapshot": {
    "Data": [
      {
        "AccountGroupKey": "Cf4xZWiYL6W1nMKpygBLLA==",
        "AccountId": "9226397",
        "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
        "AccountType": "Normal",
        "Active": true,
        "CanUseCashPositionsAsMarginCollateral": true,
        "ClientId": "9226397",
        "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
        "CreationDate": "2019-02-02T10:47:42.313000Z",
        "Currency": "EUR",
        "CurrencyDecimals": 2,
        "DirectMarketAccess": false,
        "DisplayName": "bram",
        "IndividualMargining": false,
        "IsCurrencyConversionAtSettlementTime": true,
        "IsMarginTradingAllowed": true,
        "IsShareable": false,
        "IsTrialAccount": true,
        "LegalAssetTypes": [
          "FxSpot",
          "FxForwards",
          "FxVanillaOption",
          "FxKnockInOption",
          "FxKnockOutOption",
          "FxOneTouchOption",
          "FxNoTouchOption"
        ],
        "Sharing": [
```

(continues on next page)

(continued from previous page)

```
            "NoSharing"
          ],
          "SupportsAccountValueProtectionLimit": false,
          "UseCashPositionsAsMarginCollateral": true
        }
      ]
    },
    "State": "Active"
}
```

expected_status

response
response - get the response of the request.

status_code

SubscriptionRemoveById

class saxo_openapi.endpoints.portfolio.accounts.**SubscriptionRemoveById**(*ContextId*, *ReferenceId*)

Remove subscription for the current session identified by subscription Id.

ENDPOINT = 'openapi/port/v1/accounts/subscriptions/{ContextId}/{ReferenceId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a SubscriptionRemoveById request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'ctxt_20190314'
>>> ReferenceId = 'sub_1'
>>> r = pf.accounts.SubscriptionRemoveById(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response
response - get the response of the request.

status_code

SubscriptionRemoveByTag

class saxo_openapi.endpoints.portfolio.accounts.**SubscriptionRemoveByTag**(*ContextId*, *params*)

Remove all subscriptions for the current session on this resource marked with a specific tag, and frees all resources on the server.

ENDPOINT = 'openapi/port/v1/accounts/subscriptions/{ContextId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params*)

Instantiate a SubscriptionRemoveByTag request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (required)*) – dict with the querystring parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params = {_v3_SubscriptionRemoveByTag_params}
>>> r = pf.accounts.SubscriptionRemoveByTag(ContextId, params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response

response - get the response of the request.

status_code

3.4.3 saxo_openapi.endpoints.portfolio.balances

AccountBalances

class saxo_openapi.endpoints.portfolio.balances.**AccountBalances**(*params*)

Get balance data for a client, account group or an account.

ENDPOINT = 'openapi/port/v1/balances'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)

Instantiate an AccountBalance request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.balances.AccountBalances(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "CalculationReliability": "Ok",
  "CashBalance": 999956.74,
  "ChangesScheduled": false,
  "ClosedPositionsCount": 0,
  "CollateralCreditValue": {
    "Line": 979022.6,
    "UtilzationPct": 0
  },
  "CostToClosePositions": -37.46,
  "Currency": "EUR",
  "CurrencyDecimals": 2,
  "InitialMargin": {
    "MarginAvailable": 979022.6,
    "MarginUsedByCurrentPositions": -17692.44,
    "MarginUtilizationPct": 1.78,
    "NetEquityForMargin": 996715.04
  },
  "IsPortfolioMarginModelSimple": true,
  "MarginAvailableForTrading": 979022.6,
  "MarginCollateralNotAvailable": 0,
  "MarginExposureCoveragePct": 140.84,
  "MarginNetExposure": 707697.6,
  "MarginUsedByCurrentPositions": -17692.44,
  "MarginUtilizationPct": 1.78,
  "NetEquityForMargin": 996715.04,
  "NetPositionsCount": 3,
  "NonMarginPositionsValue": 0,
  "OpenPositionsCount": 3,
  "OptionPremiumsMarketValue": 0,
  "OrdersCount": 1,
  "OtherCollateral": 0,
  "TotalValue": 996715.04,
  "TransactionsNotBooked": 0,
  "UnrealizedMarginClosedProfitLoss": 0,
  "UnrealizedMarginOpenProfitLoss": -3204.24,
  "UnrealizedMarginProfitLoss": -3204.24,
  "UnrealizedPositionsValue": -3241.7
}
```


expected_status

response

response - get the response of the request.

status_code

AccountBalancesMe

class saxo_openapi.endpoints.portfolio.balances.**AccountBalancesMe**

Get balance data for a client or an account. Defaults to logged-in client.

ENDPOINT = 'openapi/port/v1/balances/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()

Instantiate an AccountBalancesMe request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.balances.AccountBalancesMe()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "CalculationReliability": "Ok",
  "CashBalance": 999956.74,
  "ChangesScheduled": false,
  "ClosedPositionsCount": 0,
  "CollateralCreditValue": {
    "Line": 978723.62,
    "UtilzationPct": 0
  },
  "CostToClosePositions": -37.39,
  "Currency": "EUR",
  "CurrencyDecimals": 2,
  "InitialMargin": {
    "MarginAvailable": 978723.61,
    "MarginUsedByCurrentPositions": -17662.4,
    "MarginUtilizationPct": 1.77,
    "NetEquityForMargin": 996386.01
  },
  "IsPortfolioMarginModelSimple": true,
  "MarginAvailableForTrading": 978723.61,
  "MarginCollateralNotAvailable": 0,
  "MarginExposureCoveragePct": 141.03,
  "MarginNetExposure": 706507.11,
  "MarginUsedByCurrentPositions": -17662.4,
  "MarginUtilizationPct": 1.77,
  "NetEquityForMargin": 996386.01,
  "NetPositionsCount": 3,
  "NonMarginPositionsValue": 0,
  "OpenPositionsCount": 3,
  "OptionPremiumsMarketValue": 0,
```

(continues on next page)

(continued from previous page)


```
  "OrdersCount": 1,
  "OtherCollateral": 0,
  "TotalValue": 996386.01,
  "TransactionsNotBooked": 0,
  "UnrealizedMarginClosedProfitLoss": 0,
  "UnrealizedMarginOpenProfitLoss": -3533.34,
  "UnrealizedMarginProfitLoss": -3533.34,
  "UnrealizedPositionsValue": -3570.73
}
```


expected_status

response
response - get the response of the request.

status_code

BalanceSubscriptionCreate

class saxo_openapi.endpoints.portfolio.balances.**BalanceSubscriptionCreate**(*data*)
Set up a subscription and returns an initial snapshot of a balance.

ENDPOINT = 'openapi/port/v1/balances/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate an BalanceSubscriptionCreate request.

Parameters **data** (*dict (required)*) – dict representing the data body parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1551792578055",
          "ReferenceId": "U_452"
        }
```

```
>>> r = pf.balances.BalanceSubscriptionCreate(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "ContextId": "explorer_1551792578055",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "U_452",
```


(continues on next page)

(continued from previous page)


```
  "RefreshRate": 1000,
  "Snapshot": {
    "CalculationReliability": "Ok",
    "CashBalance": 999956.74,
    "ChangesScheduled": false,
    "ClosedPositionsCount": 0,
    "CollateralCreditValue": {
      "Line": 979847,
      "UtilzationPct": 0
    },
    "CostToClosePositions": -37.54,
    "Currency": "EUR",
    "CurrencyDecimals": 2,
    "InitialMargin": {
      "MarginAvailable": 979847,
      "MarginUsedByCurrentPositions": -17733.77,
      "MarginUtilizationPct": 1.78,
      "NetEquityForMargin": 997580.77
    },
    "IsPortfolioMarginModelSimple": true,
    "MarginAvailableForTrading": 979847,
    "MarginCollateralNotAvailable": 0,
    "MarginExposureCoveragePct": 140.63,
    "MarginNetExposure": 709350.7,
    "MarginUsedByCurrentPositions": -17733.77,
    "MarginUtilizationPct": 1.78,
    "NetEquityForMargin": 997580.77,
    "NetPositionsCount": 3,
    "NonMarginPositionsValue": 0,
    "OpenPositionsCount": 3,
    "OptionPremiumsMarketValue": 0,
    "OrdersCount": 1,
    "OtherCollateral": 0,
    "TotalValue": 997580.77,
    "TransactionsNotBooked": 0,
    "UnrealizedMarginClosedProfitLoss": 0,
    "UnrealizedMarginOpenProfitLoss": -2338.43,
    "UnrealizedMarginProfitLoss": -2338.43,
    "UnrealizedPositionsValue": -2375.97
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

BalanceSubscriptionRemoveById

class saxo_openapi.endpoints.portfolio.balances.**BalanceSubscriptionRemoveById**(*ContextId*, *ReferenceId*)

Removes subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/port/v1/balances/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 201

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)

Instantiate an BalanceSubscriptionRemoveById request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = "explorer_1551792578055"
>>> ReferenceId = "G_953"
>>> r = pf.BalanceSubscriptionRemoveById(ContextId=ContextId,
...                                      ReferenceId=ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

BalanceSubscriptionRemoveByTag

class saxo_openapi.endpoints.portfolio.balances.**BalanceSubscriptionRemoveByTag**(*ContextId*, *params*)

Remove multiple subscriptions for the current session on this resource and frees all resources on the server.

ENDPOINT = 'openapi/port/v1/balances/subscriptions/{ContextId}'

EXPECTED_STATUS = 201

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params*)

Instantiate an BalanceSubscriptionRemoveByTag request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "Tag": "PAGE1"
        }
```

```
>>> ContextId = "explorer_1551792578055"
>>> r = pf.BalanceSubscriptionRemoveByTag(ContextId=ContextId,
...                                       params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

MarginOverview

class saxo_openapi.endpoints.portfolio.balances.**MarginOverview**(*params*)
Get margin overview for a client, account group or an account.

ENDPOINT = 'openapi/port/v1/balances/marginoverview/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate an MarginOverview request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.balances.MarginOverview(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Groups": [
    {
      "Contributors": [
        {
          "AssetTypes": [
            "FxSpot"
          ],
          "InstrumentDescription": "British Pound/Canadian Dollar",
          "InstrumentSpecifier": "GBPCAD",
          "Margin": 2908,
          "Uic": 23
        },
        {
          "AssetTypes": [
            "FxSpot"
          ],
          "InstrumentDescription": "British Pound/Australian Dollar",
          "InstrumentSpecifier": "GBPAUD",
          "Margin": 14540,
          "Uic": 22
        },
        {
          "AssetTypes": [
            "FxSpot"
          ],
          "InstrumentDescription": "British Pound/US Dollar",
          "InstrumentSpecifier": "GBPUSD",
          "Margin": 291,
          "Uic": 31
        }
      ],
      "GroupType": "FX",
      "TotalMargin": 17739
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.4.4 saxo_openapi.endpoints.portfolio.clients

ClientDetails

class saxo_openapi.endpoints.portfolio.clients.**ClientDetails**(*ClientKey*)
Get details about a client.

ENDPOINT = 'openapi/port/v1/clients/{ClientKey}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ClientKey*)
Instantiate a ClientDetails request.

Parameters **ClientKey** (*string (required)*) – the ClientKey

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClientKey = 'Cf4xZWiYL6W1nMKpygBLLA'
>>> r = pf.clients.ClientDetails(ClientKey=ClientKey)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "AccountValueProtectionLimit": 0,
  "ClientId": "9226397",
  "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "CurrencyDecimals": 2,
  "DefaultAccountId": "9226397",
  "DefaultAccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "DefaultCurrency": "EUR",
  "IsMarginTradingAllowed": true,
  "IsVariationMarginEligible": false,
  "LegalAssetTypes": [
    "FxSpot",
    "FxForwards",
    "FxVanillaOption",
    "FxKnockInOption",
    "FxKnockOutOption",
    "FxOneTouchOption",
    "FxNoTouchOption"
  ],
  "LegalAssetTypesAreIndicative": false,
  "Name": "F. Brekeveld",
  "PositionNettingMode": "EndOfDay",
  "SupportsAccountValueProtectionLimit": true
}
```

expected_status

response
response - get the response of the request.

status_code

ClientDetailsByOwner

class saxo_openapi.endpoints.portfolio.clients.**ClientDetailsByOwner**(*params*)
Get details about clients under a particular owner.

ENDPOINT = 'openapi/port/v1/clients/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a ClientDetailsByOwner request.

Parameters **params** (*dict (required)*) – the dict representing the querystring parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "OwnerKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.clients.ClientDetailsByOwner(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "Data": [
    {
      "AccountValueProtectionLimit": 10000,
      "ClientId": "9226397",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CurrencyDecimals": 2,
      "DefaultAccountId": "9226397",
      "DefaultAccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "DefaultCurrency": "EUR",
      "IsMarginTradingAllowed": true,
      "IsVariationMarginEligible": false,
      "LegalAssetTypes": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption",
        "FxOneTouchOption",
        "FxNoTouchOption"
      ],
      "LegalAssetTypesAreIndicative": false,
      "Name": "F. Brekeveld",
      "PositionNettingMode": "EndOfDay",
      "SupportsAccountValueProtectionLimit": true
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

ClientDetailsMe

class saxo_openapi.endpoints.portfolio.clients.**ClientDetailsMe**
Get details about logged-in user's client.

ENDPOINT = 'openapi/port/v1/clients/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a ClientDetailsMe request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.clients.ClientDetailsMe()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "AccountValueProtectionLimit": 0,
  "ClientId": "9226397",
  "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "CurrencyDecimals": 2,
  "DefaultAccountId": "9226397",
  "DefaultAccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "DefaultCurrency": "EUR",
  "IsMarginTradingAllowed": true,
  "IsVariationMarginEligible": false,
  "LegalAssetTypes": [
    "FxSpot",
    "FxForwards",
    "FxVanillaOption",
    "FxKnockInOption",
    "FxKnockOutOption",
    "FxOneTouchOption",
    "FxNoTouchOption"
  ],
  "LegalAssetTypesAreIndicative": false,
  "Name": "F. Brekeveld",
  "PositionNettingMode": "EndOfDay",
  "SupportsAccountValueProtectionLimit": true
}
```

expected_status

response
response - get the response of the request.

status_code

ClientDetailsUpdate

class saxo_openapi.endpoints.portfolio.clients.**ClientDetailsUpdate**(*data*)
Enables user of the client to switch position netting mode of its own.

ENDPOINT = 'openapi/port/v1/clients/me'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*data*)
Instantiate a ClientDetailsUpdate request.

Parameters **data** (*dict (required)*) – dict with parameters representing the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AccountValueProtectionLimit": 10000,
          "NewPositionNettingMode": "EndOfDay"
        }
```

```
>>> r = pf.clients.ClientDetailsUpdate(data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

ClientSwitchPosNettingMode

class saxo_openapi.endpoints.portfolio.clients.**ClientSwitchPosNettingMode**(*params*, *data*)
Enables IB to switch position netting mode and change account value protection limit on behalf of its clients.

ENDPOINT = 'openapi/port/v1/clients/'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*params*, *data*)
Instantiate a ClientSwitchPosNettingMode request.

Parameters

- **params** (*dict (required)*) – the dict representing the querystring parameters.
- **data** (*dict (required)*) – the dict representing the data body parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> data =
        {
          "AccountValueProtectionLimit": 100000,
          "NewPositionNettingMode": "EndOfDay"
        }
```

```
>>> r = pf.clients.ClientSwitchPosNettingMode(params=params, data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.4.5 saxo_openapi.endpoints.portfolio.closedpositions

ClosedPositionById

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionById**(*ClosedPositionId*, *params*)

Get a single position by the ClosedPositionId.

ENDPOINT = 'openapi/port/v1/closedpositions/{ClosedPositionId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ClosedPositionId*, *params*)
Instantiate a ClosedPositionById request.

Parameters

- **ClosedPositionId** (*string (required)*) – the ClosedPositionId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClosedPositionId = '212702698-212702774'
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.closedpositions.ClosedPositionById(
...                           ClosedPositionId=ClosedPositionId,
...                           params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "ClosedPosition": {
    "AccountId": "9226397",
    "Amount": 80000,
    "AssetType": "FxSpot",
    "BuyOrSell": "Buy",
    "ClientId": "9226397",
    "ClosedProfitLoss": -260,
    "ClosedProfitLossInBaseCurrency": -171.0969,
    "ClosingMarketValue": 0,
    "ClosingMarketValueInBaseCurrency": 0,
    "ClosingMethod": "Fifo",
    "ClosingPositionId": "212702774",
    "ClosingPrice": 1.75612,
    "ConversionRateInstrumentToBaseSettledClosing": false,
    "ConversionRateInstrumentToBaseSettledOpening": false,
    "CostClosing": -7.02,
    "CostClosingInBaseCurrency": -4.62,
    "CostOpening": -7.04,
    "CostOpeningInBaseCurrency": -4.63,
    "ExecutionTimeClose": "2019-03-05T22:57:51.935866Z",
    "ExecutionTimeOpen": "2019-03-05T22:39:43.738721Z",
    "OpeningPositionId": "212702698",
    "OpenPrice": 1.75937,
    "Uic": 23
  },
  "ClosedPositionUniqueId": "212702698-212702774",
  "NetPositionId": "GBPCAD__FxSpot"
}
```


expected_status

response
response - get the response of the request.

status_code

ClosedPositionDetails

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionDetails**(*ClosedPositionId*, *params=None*)

Gets detailed information about a single position as specified by the query parameters

ENDPOINT = 'openapi/port/v1/closedpositions/{ClosedPositionId}/details/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ClosedPositionId*, *params=None*)
Instantiate a ClosedPositionDetails request.

Parameters

- **ClosedPositionId** (*string (required)*) – the ClosedPositionId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClosedPositionId = '212702698-212702774'
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.closedpositions.ClosedPositionDetails(
...                                ClosedPositionId=ClosedPositionId,
...                                params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "ClosedPosition": {
    "AccountId": "9226397",
    "Amount": 80000,
    "AssetType": "FxSpot",
    "BuyOrSell": "Buy",
    "ClientId": "9226397",
    "ClosedProfitLoss": -260,
    "ClosedProfitLossInBaseCurrency": -171.1385,
    "ClosingMarketValue": 0,
    "ClosingMarketValueInBaseCurrency": 0,
    "ClosingMethod": "Fifo",
    "ClosingPositionId": "212702774",
    "ClosingPrice": 1.75612,
    "ConversionRateInstrumentToBaseSettledClosing": false,
    "ConversionRateInstrumentToBaseSettledOpening": false,
    "CostClosing": -7.02,
    "CostClosingInBaseCurrency": -4.62,
    "CostOpening": -7.04,
    "CostOpeningInBaseCurrency": -4.63,
    "ExecutionTimeClose": "2019-03-05T22:57:51.935866Z",
    "ExecutionTimeOpen": "2019-03-05T22:39:43.738721Z",
    "OpeningPositionId": "212702698",
    "OpenPrice": 1.75937,
    "Uic": 23
  },
  "ClosedPositionDetails": {
    "CostClosing": {
      "Commission": -7.02
    },
    "CostClosingInBaseCurrency": {
      "Commission": -4.62
    },
    "CostOpening": {
      "Commission": -7.04
    },
    "CostOpeningInBaseCurrency": {
      "Commission": -4.63
    },
    "CurrencyConversionRateInstrumentToBaseClosing": 0.658225,
    "CurrencyConversionRateInstrumentToBaseOpening": 0.658225,
```


(continues on next page)

(continued from previous page)


```
    "ValueDateClose": "2019-03-08T00:00:00.000000Z",
    "ValueDateOpen": "2019-03-08T00:00:00.000000Z"
  },
  "ClosedPositionUniqueId": "212702698-212702774",
  "DisplayAndFormat": {
    "Currency": "CAD",
    "Decimals": 4,
    "Description": "British Pound/Canadian Dollar",
    "Format": "AllowDecimalPips",
    "Symbol": "GBPCAD"
  },
  "Exchange": {
    "Description": "Inter Bank",
    "ExchangeId": "SBFX",
    "IsOpen": true
  },
  "NetPositionId": "GBPCAD__FxSpot"
}
```


expected_status

response
response - get the response of the request.

status_code

ClosedPositionList

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionList**(*params=None*)
Returns a list of closed positions fulfilling the criteria specified by the query string parameters.

ENDPOINT = 'openapi/port/v1/closedpositions/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate a ClosedPositionList request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.closedpositions.ClosedPositionList(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "__count": 5,
  "Data": [
    {
```


(continues on next page)

(continued from previous page)


```
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": 80000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Buy",
        "ClientId": "9226397",
        "ClosedProfitLoss": -260,
        "ClosedProfitLossInBaseCurrency": -171.1138,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212702774",
        "ClosingPrice": 1.75612,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": false,
        "CostClosing": -7.02,
        "CostClosingInBaseCurrency": -4.62,
        "CostOpening": -7.04,
        "CostOpeningInBaseCurrency": -4.63,
        "ExecutionTimeClose": "2019-03-05T22:57:51.935866Z",
        "ExecutionTimeOpen": "2019-03-05T22:39:43.738721Z",
        "OpeningPositionId": "212702698",
        "OpenPrice": 1.75937,
        "Uic": 23
      },
      "ClosedPositionUniqueId": "212702698-212702774",
      "NetPositionId": "GBPCAD__FxSpot"
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": -100000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Sell",
        "ClientId": "9226397",
        "ClosedProfitLoss": 29,
        "ClosedProfitLossInBaseCurrency": 25.6447,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212702772",
        "ClosingPrice": 1.13025,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": false,
        "CostClosing": -5.65,
        "CostClosingInBaseCurrency": -5,
        "CostOpening": -5.65,
        "CostOpeningInBaseCurrency": -5,
        "ExecutionTimeClose": "2019-03-05T22:57:51.776721Z",
        "ExecutionTimeOpen": "2019-03-05T22:39:43.546536Z",
        "OpeningPositionId": "212702696",
        "OpenPrice": 1.13054,
        "Uic": 21
      },
      "ClosedPositionUniqueId": "212702696-212702772",
      "NetPositionId": "EURUSD__FxSpot"
    },
```


(continues on next page)

(continued from previous page)


```
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": 10000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Buy",
        "ClientId": "9226397",
        "ClosedProfitLoss": -13.2,
        "ClosedProfitLossInBaseCurrency": -11.67276,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212702680",
        "ClosingPrice": 1.31731,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -3,
        "CostClosingInBaseCurrency": -2.65,
        "CostOpening": -3,
        "CostOpeningInBaseCurrency": -2.65,
        "ExecutionTimeClose": "2019-03-05T22:23:38.888231Z",
        "ExecutionTimeOpen": "2019-03-04T17:11:39.129241Z",
        "OpeningPositionId": "212675868",
        "OpenPrice": 1.31863,
        "Uic": 31
      },
      "ClosedPositionUniqueId": "212675868-212702680",
      "NetPositionId": "GBPUSD__FxSpot"
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Buy",
        "ClientId": "9226397",
        "ClosedProfitLoss": 50,
        "ClosedProfitLossInBaseCurrency": 32.9065,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212702664",
        "ClosingPrice": 1.75878,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -8.79,
        "CostClosingInBaseCurrency": -5.78,
        "CostOpening": -8.79,
        "CostOpeningInBaseCurrency": -5.82,
        "ExecutionTimeClose": "2019-03-05T22:22:51.922693Z",
        "ExecutionTimeOpen": "2019-03-03T23:34:51.823660Z",
        "OpeningPositionId": "212550210",
        "OpenPrice": 1.75828,
        "Uic": 23
      },
      "ClosedPositionUniqueId": "212550210-212702664",
      "NetPositionId": "GBPCAD__FxSpot"
```


(continues on next page)

(continued from previous page)


```
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": 400000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Buy",
        "ClientId": "9226397",
        "ClosedProfitLoss": -1800,
        "ClosedProfitLossInBaseCurrency": -1118.124,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212702660",
        "ClosingPrice": 1.85952,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -37.19,
        "CostClosingInBaseCurrency": -23.1,
        "CostOpening": -29.824,
        "CostOpeningInBaseCurrency": -18.62,
        "ExecutionTimeClose": "2019-03-05T22:22:07.523028Z",
        "ExecutionTimeOpen": "2019-03-03T23:35:08.243690Z",
        "OpeningPositionId": "212550212",
        "OpenPrice": 1.86402,
        "Uic": 22
      },
      "ClosedPositionUniqueId": "212550212-212702660",
      "NetPositionId": "GBPAUD__FxSpot"
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

ClosedPositionSubscription

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionSubscription**(*data*, *params=No*

Sets up a subscription and returns an initial snapshot of list of closed positions specified by the parameters in the request.

ENDPOINT = 'openapi/port/v1/closedpositions/subscriptions/'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*, *params=None*)
Instantiate a ClosedPositionSubscription request.

Parameters

- **data** (*dict (required)*) – dict representing the parameters of the data body

- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1551913039211",
          "ReferenceId": "D_975"
        }
```

```
>>> r = pf.closedpositions.ClosedPositionSubscription(
...               data=data,
...               params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "ContextId": "explorer_1551913039211",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "D_975",
  "RefreshRate": 1000,
  "Snapshot": {
    "Data": [
      {
        "ClosedPosition": {
          "AccountId": "9226397",
          "Amount": -40000,
          "AssetType": "FxSpot",
          "BuyOrSell": "Sell",
          "ClientId": "9226397",
          "ClosedProfitLoss": -582.8,
          "ClosedProfitLossInBaseCurrency": -383.389152,
          "ClosingMarketValue": 0,
          "ClosingMarketValueInBaseCurrency": 0,
          "ClosingMethod": "Fifo",
          "ClosingPositionId": "212725160",
          "ClosingPrice": 1.77074,
          "ConversionRateInstrumentToBaseSettledClosing": false,
          "ConversionRateInstrumentToBaseSettledOpening": true,
          "CostClosing": -4.03,
          "CostClosingInBaseCurrency": -2.65,
          "CostOpening": -3.51,
          "CostOpeningInBaseCurrency": -2.32,
          "ExecutionTimeClose": "2019-03-06T23:07:47.040598Z",
          "ExecutionTimeOpen": "2019-03-06T10:24:50.635259Z",
          "OpeningPositionId": "212710176",
```

(continues on next page)

(continued from previous page)


```
        "OpenPrice": 1.75617,
        "Uic": 23
      },
      "ClosedPositionUniqueId": "212710176-212725160",
      "NetPositionId": "GBPCAD__FxSpot"
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": -40000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Sell",
        "ClientId": "9226397",
        "ClosedProfitLoss": -590.8,
        "ClosedProfitLossInBaseCurrency": -388.651872,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212725128",
        "ClosingPrice": 1.77094,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -4.03,
        "CostClosingInBaseCurrency": -2.65,
        "CostOpening": -3.51,
        "CostOpeningInBaseCurrency": -2.32,
        "ExecutionTimeClose": "2019-03-06T23:02:56.295679Z",
        "ExecutionTimeOpen": "2019-03-06T10:24:50.635259Z",
        "OpeningPositionId": "212710176",
        "OpenPrice": 1.75617,
        "Uic": 23
      },
      "ClosedPositionUniqueId": "212710176-212725128",
      "NetPositionId": "GBPCAD__FxSpot"
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": 40000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Buy",
        "ClientId": "9226397",
        "ClosedProfitLoss": 6,
        "ClosedProfitLossInBaseCurrency": 5.30445,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212724952",
        "ClosingPrice": 1.13076,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -3,
        "CostClosingInBaseCurrency": -2.65,
        "CostOpening": -2.26,
        "CostOpeningInBaseCurrency": -2,
        "ExecutionTimeClose": "2019-03-06T22:55:59.228387Z",
        "ExecutionTimeOpen": "2019-03-06T10:24:50.460091Z",
```


(continues on next page)

(continued from previous page)

```
          "OpeningPositionId": "212710174",
          "OpenPrice": 1.13061,
          "Uic": 21
        },
        "ClosedPositionUniqueId": "212710174-212724952",
        "NetPositionId": "EURUSD__FxSpot"
      }
    ],
    "MaxRows": 100000
  },
  "State": "Active"
}
```

expected_status

response
response - get the response of the request.

status_code

ClosedPositionSubscriptionRemoveById

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionSubscriptionRemoveById**

Removes subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/port/v1/closedpositions/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a ClosedPositionSubscriptionRemoveById request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.closedpositions.ClosedPositionSubscriptionRemoveById(
...             ContextId=ContextId,
...             ReferenceId=ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

ClosedPositionSubscriptionUpdate

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionSubscriptionUpdate**(*Con Ref er- en- ceId dat*

Extends or reduces the page size, number of positions shown, on a running closed positions subscription. When expanding the page size, the subsequent closed positions are streamed so to avoid race conditions.

ENDPOINT = 'openapi/port/v1/closedpositions/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 200

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*, *data*)
Instantiate a ClosedPositionSubscriptionUpdate request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId
- **data** (*dict (required)*) – dict representing the parameters of the data body

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "NewPageSize": 25630
        }
```

```
>>> ContextId = ''
>>> ReferenceId = ''
>>> r = pf.closedpositions.ClosedPositionSubscriptionUpdate(
...                                         ContextId=ContextId,
...                                         ReferenceId=ReferenceId,
...                                         data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

ClosedPositionSubscriptionsRemove

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionSubscriptionsRemove**(*Co*
pa

Removes multiple all subscriptions for the current session on this resource, and frees all resources on the server.

ENDPOINT = 'openapi/port/v1/closedpositions/subscriptions/{ContextId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

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

Instantiate a ClosedPositionSubscriptionsRemove request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (required)*) – dict representing the parameters of the querystring

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.closedpositions.ClosedPositionSubscriptionsRemove(
...             ContextId=ContextId,
...             params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

ClosedPositionsMe

class saxo_openapi.endpoints.portfolio.closedpositions.**ClosedPositionsMe**(*params=None*)

Returns a list of closed positions fulfilling the criteria specified by the query string parameters.

ENDPOINT = 'openapi/port/v1/closedpositions/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)

Instantiate a ClosedPositionsMe request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
```

(continues on next page)

(continued from previous page)

```
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.closedpositions.ClosedPositionsMe(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "__count": 3,
  "Data": [
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": -40000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Sell",
        "ClientId": "9226397",
        "ClosedProfitLoss": -582.8,
        "ClosedProfitLossInBaseCurrency": -383.377496,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212725160",
        "ClosingPrice": 1.77074,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -4.03,
        "CostClosingInBaseCurrency": -2.65,
        "CostOpening": -3.51,
        "CostOpeningInBaseCurrency": -2.32,
        "ExecutionTimeClose": "2019-03-06T23:07:47.040598Z",
        "ExecutionTimeOpen": "2019-03-06T10:24:50.635259Z",
        "OpeningPositionId": "212710176",
        "OpenPrice": 1.75617,
        "Uic": 23
      },
      "ClosedPositionUniqueId": "212710176-212725160",
      "NetPositionId": "GBPCAD__FxSpot"
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": -40000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Sell",
        "ClientId": "9226397",
        "ClosedProfitLoss": -590.8,
        "ClosedProfitLossInBaseCurrency": -388.640056,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212725128",
        "ClosingPrice": 1.77094,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
```


(continues on next page)

(continued from previous page)


```
        "CostClosing": -4.03,
        "CostClosingInBaseCurrency": -2.65,
        "CostOpening": -3.51,
        "CostOpeningInBaseCurrency": -2.32,
        "ExecutionTimeClose": "2019-03-06T23:02:56.295679Z",
        "ExecutionTimeOpen": "2019-03-06T10:24:50.635259Z",
        "OpeningPositionId": "212710176",
        "OpenPrice": 1.75617,
        "Uic": 23
      },
      "ClosedPositionUniqueId": "212710176-212725128",
      "NetPositionId": "GBPCAD__FxSpot"
    },
    {
      "ClosedPosition": {
        "AccountId": "9226397",
        "Amount": 40000,
        "AssetType": "FxSpot",
        "BuyOrSell": "Buy",
        "ClientId": "9226397",
        "ClosedProfitLoss": 6,
        "ClosedProfitLossInBaseCurrency": 5.30466,
        "ClosingMarketValue": 0,
        "ClosingMarketValueInBaseCurrency": 0,
        "ClosingMethod": "Fifo",
        "ClosingPositionId": "212724952",
        "ClosingPrice": 1.13076,
        "ConversionRateInstrumentToBaseSettledClosing": false,
        "ConversionRateInstrumentToBaseSettledOpening": true,
        "CostClosing": -3,
        "CostClosingInBaseCurrency": -2.65,
        "CostOpening": -2.26,
        "CostOpeningInBaseCurrency": -2,
        "ExecutionTimeClose": "2019-03-06T22:55:59.228387Z",
        "ExecutionTimeOpen": "2019-03-06T10:24:50.460091Z",
        "OpeningPositionId": "212710174",
        "OpenPrice": 1.13061,
        "Uic": 21
      },
      "ClosedPositionUniqueId": "212710174-212724952",
      "NetPositionId": "EURUSD__FxSpot"
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.4.6 saxo_openapi.endpoints.portfolio.exposure

CreateExposureSubscription

class saxo_openapi.endpoints.portfolio.exposure.**CreateExposureSubscription**(*data*)

Sets up a subscription and returns an initial snapshot of list of instrument exposure specified by the parameters in the request.

ENDPOINT = 'openapi/port/v1/exposure/instruments/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)

Instantiate a CreateExposureSubscription request.

Parameters **data** (*dict (required)*) – dict representing the body with request parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1552035128308",
          "ReferenceId": "Z_807"
        }
```

```
>>> r = pf.exposure.CreateExposureSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "ContextId": "explorer_1552035128308",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "Z_807",
  "RefreshRate": 1000,
  "Snapshot": {
    "Data": [
      {
        "Amount": 60000,
        "AssetType": "FxSpot",
        "AverageOpenPrice": 1.13071,
        "CalculationReliability": "Ok",
        "CanBeClosed": true,
        "DisplayAndFormat": {
          "Currency": "USD",
          "Decimals": 4,
          "Description": "Euro/US Dollar",
          "Format": "AllowDecimalPips",
          "Symbol": "EURUSD"
        },
        "InstrumentPriceDayPercentChange": 0.44,
        "NetPositionId": "EURUSD__FxSpot",
```

(continues on next page)

(continued from previous page)


```
        "ProfitLossOnTrade": -396.6,
        "Uic": 21
      },
      {
        "Amount": -50000,
        "AssetType": "FxSpot",
        "AverageOpenPrice": 8.6839,
        "CalculationReliability": "Ok",
        "CanBeClosed": true,
        "DisplayAndFormat": {
          "Currency": "DKK",
          "Decimals": 4,
          "Description": "British Pound/Danish Krone",
          "Format": "Normal",
          "Symbol": "GBPDKK"
        },
        "InstrumentPriceDayPercentChange": -0.98,
        "NetPositionId": "GBPDKK__FxSpot",
        "ProfitLossOnTrade": 2420,
        "Uic": 25
      }
    ]
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

CurrencyExposureMe

class saxo_openapi.endpoints.portfolio.exposure.**CurrencyExposureMe**
Returns a list of currencies and net exposures.

ENDPOINT = 'openapi/port/v1/exposure/currency/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a CurrencyExposureMe request.

Parameters **None** –

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.exposure.CurrencyExposureMe()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
[
  {
    "Amount": 1057573.99,
    "Currency": "EUR"
  },
  {
    "Amount": -67842.6,
    "Currency": "USD"
  },
  {
    "Amount": -50000,
    "Currency": "GBP"
  },
  {
    "Amount": 434195,
    "Currency": "DKK"
  }
]
```


expected_status

response
response - get the response of the request.

status_code

CurrencyExposureSpecific

class saxo_openapi.endpoints.portfolio.exposure.**CurrencyExposureSpecific**(*params*)
Returns a list of currencies in which there is an exposure.

ENDPOINT = 'openapi/port/v1/exposure/currency/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a CurrencyExposureSpecific request.

Parameters **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.exposure.CurrencyExposureSpecific(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
[
  {
    "Amount": 1057573.99,
    "Currency": "EUR"
  },
  {
    "Amount": -67842.6,
    "Currency": "USD"
  },
  {
    "Amount": -50000,
    "Currency": "GBP"
  },
  {
    "Amount": 434195,
    "Currency": "DKK"
  }
]
```


expected_status

response
response - get the response of the request.

status_code

FxSpotExposureMe

class saxo_openapi.endpoints.portfolio.exposure.**FxSpotExposureMe**
Returns a list of currencies and net exposures.

ENDPOINT = 'openapi/port/v1/exposure/fxspot/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a FxSpotExposureMe request.

Parameters **None** –

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.exposure.FxSpotExposureMe()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:


```
[
  {
    "Amount": 431950,
    "AmountInCalculationEntityCurrency": 57878,
    "Currency": "DKK"
  },
```


(continues on next page)

(continued from previous page)

```
  {
    "Amount": 60000,
    "AmountInCalculationEntityCurrency": 60000,
    "Currency": "EUR"
  },
  {
    "Amount": -50000,
    "AmountInCalculationEntityCurrency": -57878,
    "Currency": "GBP"
  },
  {
    "Amount": -67402.2,
    "AmountInCalculationEntityCurrency": -60000,
    "Currency": "USD"
  }
]
```

expected_status

response
response - get the response of the request.

status_code

FxSpotExposureSpecific

class saxo_openapi.endpoints.portfolio.exposure.**FxSpotExposureSpecific**(*params*)
Returns a list of currencies in which there is an exposure.

ENDPOINT = 'openapi/port/v1/exposure/fxspot/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a FxSpotExposureSpecific request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.exposure.FxSpotExposureSpecific(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
[
  {
```

(continues on next page)

(continued from previous page)

```
    "Amount": 432350,
    "AmountInCalculationEntityCurrency": 57929,
    "Currency": "DKK"
  },
  {
    "Amount": 60000,
    "AmountInCalculationEntityCurrency": 60000,
    "Currency": "EUR"
  },
  {
    "Amount": -50000,
    "AmountInCalculationEntityCurrency": -57929,
    "Currency": "GBP"
  },
  {
    "Amount": -67398,
    "AmountInCalculationEntityCurrency": -60000,
    "Currency": "USD"
  }
]
```

expected_status

response
response - get the response of the request.

status_code

NetInstrumentExposureSpecific

class saxo_openapi.endpoints.portfolio.exposure.**NetInstrumentExposureSpecific**(*params*)
Returns a list instruments and net exposures.

ENDPOINT = 'openapi/port/v1/exposure/instruments/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a NetInstrumentExposureSpecific request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.exposure.NetInstrumentExposureSpecific(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
[
  {
    "Amount": 60000,
    "AssetType": "FxSpot",
    "AverageOpenPrice": 1.13071,
    "CalculationReliability": "Ok",
    "CanBeClosed": true,
    "DisplayAndFormat": {
      "Currency": "USD",
      "Decimals": 4,
      "Description": "Euro/US Dollar",
      "Format": "AllowDecimalPips",
      "Symbol": "EURUSD"
    },
    "InstrumentPriceDayPercentChange": 0.42,
    "NetPositionId": "EURUSD__FxSpot",
    "ProfitLossOnTrade": -405,
    "Uic": 21
  },
  {
    "Amount": -50000,
    "AssetType": "FxSpot",
    "AverageOpenPrice": 8.6839,
    "CalculationReliability": "Ok",
    "CanBeClosed": true,
    "DisplayAndFormat": {
      "Currency": "DKK",
      "Decimals": 4,
      "Description": "British Pound/Danish Krone",
      "Format": "Normal",
      "Symbol": "GBPDKK"
    },
    "InstrumentPriceDayPercentChange": -1.02,
    "NetPositionId": "GBPDKK__FxSpot",
    "ProfitLossOnTrade": 2600,
    "Uic": 25
  }
]
```


expected_status

response
response - get the response of the request.

status_code

NetInstrumentsExposureMe

class saxo_openapi.endpoints.portfolio.exposure.**NetInstrumentsExposureMe**
Returns a list instruments and net exposures.

ENDPOINT = 'openapi/port/v1/exposure/instruments/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a NetInstrumentsExposureMe request.

Parameters `None` –

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.exposure.NetInstrumentsExposureMe()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:


```
[
  {
    "Amount": 60000,
    "AssetType": "FxSpot",
    "AverageOpenPrice": 1.13071,
    "CalculationReliability": "Ok",
    "CanBeClosed": true,
    "DisplayAndFormat": {
      "Currency": "USD",
      "Decimals": 4,
      "Description": "Euro/US Dollar",
      "Format": "AllowDecimalPips",
      "Symbol": "EURUSD"
    },
    "InstrumentPriceDayPercentChange": 0.42,
    "NetPositionId": "EURUSD__FxSpot",
    "ProfitLossOnTrade": -408.6,
    "Uic": 21
  },
  {
    "Amount": -50000,
    "AssetType": "FxSpot",
    "AverageOpenPrice": 8.6839,
    "CalculationReliability": "Ok",
    "CanBeClosed": true,
    "DisplayAndFormat": {
      "Currency": "DKK",
      "Decimals": 4,
      "Description": "British Pound/Danish Krone",
      "Format": "Normal",
      "Symbol": "GBPDKK"
    },
    "InstrumentPriceDayPercentChange": -1,
    "NetPositionId": "GBPDKK__FxSpot",
    "ProfitLossOnTrade": 2530,
    "Uic": 25
  }
]
```


expected_status

response
response - get the response of the request.

status_code

RemoveExposureSubscription

class saxo_openapi.endpoints.portfolio.exposure.**RemoveExposureSubscription**(*ContextId*, *ReferenceId*)

Removes subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/port/v1/exposure/instruments/subscriptions/{ContextId}/{ReferenceI

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)

Instantiate a RemoveExposureSubscription request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.exposure.RemoveExposureSubscription(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data returned.

expected_status

response

response - get the response of the request.

status_code

RemoveExposureSubscriptionsByTag

class saxo_openapi.endpoints.portfolio.exposure.**RemoveExposureSubscriptionsByTag**(*ContextId*, *params=None*)

Removes multiple all subscriptions for the current session on this resource, and frees all resources on the server.

ENDPOINT = 'openapi/port/v1/exposure/instruments/subscriptions/{ContextId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

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

Instantiate a RemoveExposureSubscriptionsByTag request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> ContextId = 'explorer_1552035128308'
>>> r = pf.exposure.RemoveExposureSubscriptionsByTag(ContextId,
...                                                  params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data returned.

expected_status

response
response - get the response of the request.

status_code

3.4.7 saxo_openapi.endpoints.portfolio.netpositions

NetPositionListSubscription

class saxo_openapi.endpoints.portfolio.netpositions.**NetPositionListSubscription**(*data*)
Create a subscription on a list of net positions and make it active.

Sets up a subscription and returns an initial snapshot of list of net positions specified by the parameters in the request.

ENDPOINT = 'openapi/port/v1/netpositions/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a NetPositionListSubscription request.

Parameters **data** (*dict (required)*) – dict representing the data body parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1551702571343",
          "ReferenceId": "F_20"
        }
```

```
>>> r = pf.netpositions.NetPositionListSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "ContextId": "explorer_1551702571343",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "F_20",
  "RefreshRate": 1000,
  "Snapshot": {
    "Data": [
      {
        "NetPositionBase": {
          "Amount": 100000,
          "AssetType": "FxSpot",
          "CanBeClosed": true,
          "ClientId": "9226397",
          "IsMarketOpen": true,
          "NumberOfRelatedOrders": 0,
          "PositionsAccount": "9226397",
          "SinglePositionId": "212550210",
          "SinglePositionStatus": "Open",
          "Uic": 23,
          "ValueDate": "2019-03-06T00:00:00.000000Z"
        },
        "NetPositionId": "GBPCAD__FxSpot",
        "NetPositionView": {
          "AverageOpenPrice": 1.75824,
          "CalculationReliability": "Ok",
          "CurrentPrice": 1.75313,
          "CurrentPriceDelayMinutes": 0,
          "CurrentPriceType": "Bid",
          "Exposure": 100000,
          "ExposureCurrency": "GBP",
          "ExposureInBaseCurrency": 116179,
          "InstrumentPriceDayPercentChange": -0.17,
          "PositionCount": 1,
          "PositionsNotClosedCount": 1,
          "ProfitLossOnTrade": -511,
          "ProfitLossOnTradeInBaseCurrency": -338.57,
          "Status": "Open",
          "TradeCostsTotal": -17.56,
          "TradeCostsTotalInBaseCurrency": -11.63
        }
      },
      {
        "NetPositionBase": {
          "Amount": 500000,
          "AssetType": "FxSpot",
          "CanBeClosed": true,
          "ClientId": "9226397",
          "IsMarketOpen": true,
          "NumberOfRelatedOrders": 0,
          "PositionsAccount": "9226397",
```


(continues on next page)

(continued from previous page)


```
        "SinglePositionId": "212550212",
        "SinglePositionStatus": "Open",
        "Uic": 22,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "GBPAUD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 1.86391,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.85805,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 500000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 580895,
        "InstrumentPriceDayPercentChange": -0.36,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -2930,
        "ProfitLossOnTradeInBaseCurrency": -1831.62,
        "Status": "Open",
        "TradeCostsTotal": -93.05,
        "TradeCostsTotalInBaseCurrency": -58.17
      }
    },
    {
      "NetPositionBase": {
        "Amount": 0,
        "AssetType": "FxSpot",
        "CanBeClosed": false,
        "ClientId": "9226397",
        "IsMarketOpen": true,
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "EURUSD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 0,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.13416,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 0,
        "ExposureCurrency": "EUR",
        "ExposureInBaseCurrency": 0,
        "InstrumentPriceDayPercentChange": -0.2,
        "PositionCount": 2,
        "PositionsNotClosedCount": 2,
        "ProfitLossOnTrade": 5,
        "ProfitLossOnTradeInBaseCurrency": 4.41,
        "Status": "Open",
        "TradeCostsTotal": -11.38,
        "TradeCostsTotalInBaseCurrency": -10.03
      }
    },
```


(continues on next page)

(continued from previous page)


```
      {
        "NetPositionBase": {
          "Amount": 10000,
          "AssetType": "FxSpot",
          "CanBeClosed": true,
          "ClientId": "9226397",
          "IsMarketOpen": true,
          "NumberOfRelatedOrders": 0,
          "PositionsAccount": "9226397",
          "SinglePositionId": "212675868",
          "SinglePositionStatus": "Open",
          "Uic": 31,
          "ValueDate": "2019-03-06T00:00:00.000000Z"
        },
        "NetPositionId": "GBPUSD__FxSpot",
        "NetPositionView": {
          "AverageOpenPrice": 1.31849,
          "CalculationReliability": "Ok",
          "CurrentPrice": 1.3176,
          "CurrentPriceDelayMinutes": 0,
          "CurrentPriceType": "Bid",
          "Exposure": 10000,
          "ExposureCurrency": "GBP",
          "ExposureInBaseCurrency": 11617.9,
          "InstrumentPriceDayPercentChange": -0.23,
          "PositionCount": 1,
          "PositionsNotClosedCount": 1,
          "ProfitLossOnTrade": -8.9,
          "ProfitLossOnTradeInBaseCurrency": -7.85,
          "Status": "Open",
          "TradeCostsTotal": -6,
          "TradeCostsTotalInBaseCurrency": -5.29
        }
      }
    ]
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

NetPositionSubscriptionRemove

class saxo_openapi.endpoints.portfolio.netpositions.**NetPositionSubscriptionRemove**(*ContextId*, *ReferenceId*)

Removes subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/port/v1/positions/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a NetPositionSubscriptionRemove request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ...
>>> ReferenceId = ...
>>> r = pf.netpositions.NetPositionSubscriptionRemove(
...         ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

NetPositionSubscriptionRemoveMultiple

class saxo_openapi.endpoints.portfolio.netpositions.**NetPositionSubscriptionRemoveMultiple**(

Remove multiple all subscriptions for the current session on this resource, and frees all resources on the server.

ENDPOINT = 'openapi/port/v1/netpositions/subscriptions/{ContextId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params=None*)
Instantiate a NetPositionSubscriptionRemoveMultiple request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
```

(continues on next page)

(continued from previous page)

```
        {
          "Tag": "..."
        }
```

```
>>> ContextId = ...
>>> r = pf.netpositions.NetPositionSubscriptionRemoveMultiple(
...         ContextId,
...         params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

NetPositionsMe

class saxo_openapi.endpoints.portfolio.netpositions.**NetPositionsMe**(*params=None*)
Get netpositions for the logged-in client.

ENDPOINT = 'openapi/port/v1/netpositions/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate a NetPositionsMe request.

Parameters **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.netpositions.NetPositionsMe(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "__count": 4,
  "Data": [
    {
      "NetPositionBase": {
        "Amount": 100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
```

(continues on next page)

(continued from previous page)


```
        "ClientId": "9226397",
        "IsMarketOpen": true,
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "SinglePositionId": "212550210",
        "SinglePositionStatus": "Open",
        "Uic": 23,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "GBPCAD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 1.75824,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.75322,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 100000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 116201.5,
        "InstrumentPriceDayPercentChange": -0.17,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -502,
        "ProfitLossOnTradeInBaseCurrency": -332.64,
        "Status": "Open",
        "TradeCostsTotal": -17.56,
        "TradeCostsTotalInBaseCurrency": -11.64
      }
    },
    {
      "NetPositionBase": {
        "Amount": 500000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "IsMarketOpen": true,
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "SinglePositionId": "212550212",
        "SinglePositionStatus": "Open",
        "Uic": 22,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "GBPAUD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 1.86391,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.85813,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 500000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 581007.5,
        "InstrumentPriceDayPercentChange": -0.35,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -2890,
```


(continues on next page)

(continued from previous page)


```
      "ProfitLossOnTradeInBaseCurrency": -1806.86,
      "Status": "Open",
      "TradeCostsTotal": -93.05,
      "TradeCostsTotalInBaseCurrency": -58.18
    }
  },
  {
    "NetPositionBase": {
      "Amount": 0,
      "AssetType": "FxSpot",
      "CanBeClosed": false,
      "ClientId": "9226397",
      "IsMarketOpen": true,
      "NumberOfRelatedOrders": 0,
      "PositionsAccount": "9226397",
      "Uic": 21,
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "NetPositionId": "EURUSD__FxSpot",
    "NetPositionView": {
      "AverageOpenPrice": 0,
      "CalculationReliability": "Ok",
      "CurrentPrice": 1.13343,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 0,
      "ExposureCurrency": "EUR",
      "ExposureInBaseCurrency": 0,
      "InstrumentPriceDayPercentChange": -0.26,
      "PositionCount": 2,
      "PositionsNotClosedCount": 2,
      "ProfitLossOnTrade": 5,
      "ProfitLossOnTradeInBaseCurrency": 4.41,
      "Status": "Open",
      "TradeCostsTotal": -11.38,
      "TradeCostsTotalInBaseCurrency": -10.04
    }
  },
  {
    "NetPositionBase": {
      "Amount": 10000,
      "AssetType": "FxSpot",
      "CanBeClosed": true,
      "ClientId": "9226397",
      "IsMarketOpen": true,
      "NumberOfRelatedOrders": 0,
      "PositionsAccount": "9226397",
      "SinglePositionId": "212675868",
      "SinglePositionStatus": "Open",
      "Uic": 31,
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "NetPositionId": "GBPUSD__FxSpot",
    "NetPositionView": {
      "AverageOpenPrice": 1.31849,
      "CalculationReliability": "Ok",
      "CurrentPrice": 1.31701,
```


(continues on next page)

(continued from previous page)

```
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 10000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 11620.15,
        "InstrumentPriceDayPercentChange": -0.27,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -14.8,
        "ProfitLossOnTradeInBaseCurrency": -13.06,
        "Status": "Open",
        "TradeCostsTotal": -6,
        "TradeCostsTotalInBaseCurrency": -5.29
      }
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

NetPositionsQuery

class saxo_openapi.endpoints.portfolio.netpositions.**NetPositionsQuery**(*params*)
Get netpositions for a client, account group, account or position. Returns a list of net positions fulfilling the criteria specified by the query string parameters. Each net position may include all related sub positions if fieldGroups includes SubPositions.

ENDPOINT = 'openapi/port/v1/netpositions'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a NetPositionsQuery request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.netpositions.NetPositionsQuery(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "__count": 4,
  "Data": [
    {
      "NetPositionBase": {
        "AccountId": "9226397",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "IsMarketOpen": true,
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "SinglePositionId": "212550210",
        "SinglePositionStatus": "Open",
        "Uic": 23,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "GBPCAD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 1.75824,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.75287,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 100000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 116180.5,
        "InstrumentPriceDayPercentChange": -0.19,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -537,
        "ProfitLossOnTradeInBaseCurrency": -355.85,
        "Status": "Open",
        "TradeCostsTotal": -17.55,
        "TradeCostsTotalInBaseCurrency": -11.63
      }
    },
    {
      "NetPositionBase": {
        "AccountId": "9226397",
        "Amount": 500000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "IsMarketOpen": true,
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "SinglePositionId": "212550212",
        "SinglePositionStatus": "Open",
        "Uic": 22,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "GBPAUD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 1.86391,
        "CalculationReliability": "Ok",
```


(continues on next page)

(continued from previous page)


```
        "CurrentPrice": 1.8575,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 500000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 580902.5,
        "InstrumentPriceDayPercentChange": -0.39,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -3205,
        "ProfitLossOnTradeInBaseCurrency": -2004.09,
        "Status": "Open",
        "TradeCostsTotal": -93.04,
        "TradeCostsTotalInBaseCurrency": -58.18
      }
    },
    {
      "NetPositionBase": {
        "AccountId": "9226397",
        "Amount": 0,
        "AssetType": "FxSpot",
        "CanBeClosed": false,
        "ClientId": "9226397",
        "IsMarketOpen": true,
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "EURUSD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 0,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.13377,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 0,
        "ExposureCurrency": "EUR",
        "ExposureInBaseCurrency": 0,
        "InstrumentPriceDayPercentChange": -0.23,
        "PositionCount": 2,
        "PositionsNotClosedCount": 2,
        "ProfitLossOnTrade": 5,
        "ProfitLossOnTradeInBaseCurrency": 4.41,
        "Status": "Open",
        "TradeCostsTotal": -11.38,
        "TradeCostsTotalInBaseCurrency": -10.04
      }
    },
    {
      "NetPositionBase": {
        "AccountId": "9226397",
        "Amount": 10000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "IsMarketOpen": true,
```


(continues on next page)

(continued from previous page)


```
        "NumberOfRelatedOrders": 0,
        "PositionsAccount": "9226397",
        "SinglePositionId": "212675868",
        "SinglePositionStatus": "Open",
        "Uic": 31,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "NetPositionId": "GBPUSD__FxSpot",
      "NetPositionView": {
        "AverageOpenPrice": 1.31849,
        "CalculationReliability": "Ok",
        "CurrentPrice": 1.31718,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 10000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 11618.05,
        "InstrumentPriceDayPercentChange": -0.26,
        "PositionCount": 1,
        "PositionsNotClosedCount": 1,
        "ProfitLossOnTrade": -13.1,
        "ProfitLossOnTradeInBaseCurrency": -11.55,
        "Status": "Open",
        "TradeCostsTotal": -6,
        "TradeCostsTotalInBaseCurrency": -5.29
      }
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

SingleNetPosition

class saxo_openapi.endpoints.portfolio.netpositions.**SingleNetPosition**(*NetPositionId*, *params*)

Get a single net position.

ENDPOINT = 'openapi/port/v1/netpositions/{NetPositionId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*NetPositionId*, *params*)
Instantiate a SingleNetPosition request.

Parameters

- **NetPositionId** (*string (required)*) – the NetPositionId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> NetPositionId = "GBPCAD__FxSpot"
>>> r = pf.netpositions.SingleNetPosition(NetPositionId=NetPositionId,
...                                       params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "NetPositionBase": {
    "AccountId": "9226397",
    "Amount": 100000,
    "AssetType": "FxSpot",
    "CanBeClosed": true,
    "ClientId": "9226397",
    "IsMarketOpen": true,
    "NumberOfRelatedOrders": 0,
    "PositionsAccount": "9226397",
    "SinglePositionId": "212550210",
    "SinglePositionStatus": "Open",
    "Uic": 23,
    "ValueDate": "2019-03-06T00:00:00.000000Z"
  },
  "NetPositionDetails": {
    "CloseCost": {
      "Commission": 8.77
    },
    "CloseCostInBaseCurrency": {
      "Commission": 5.81
    },
    "MarketValue": -508,
    "MarketValueInBaseCurrency": -336.7,
    "OpenCost": {
      "Commission": 8.79
    },
    "OpenCostInBaseCurrency": {
      "Commission": 5.83
    }
  },
  "NetPositionId": "GBPCAD__FxSpot",
  "NetPositionView": {
    "AverageOpenPrice": 1.75824,
    "CalculationReliability": "Ok",
    "CurrentPrice": 1.75316,
    "CurrentPriceDelayMinutes": 0,
    "CurrentPriceType": "Bid",
    "Exposure": 100000,
    "ExposureCurrency": "GBP",
```


(continues on next page)

(continued from previous page)

```
    "ExposureInBaseCurrency": 116226,
    "InstrumentPriceDayPercentChange": -0.17,
    "PositionCount": 1,
    "PositionsNotClosedCount": 1,
    "ProfitLossOnTrade": -508,
    "ProfitLossOnTradeInBaseCurrency": -336.7,
    "Status": "Open",
    "TradeCostsTotal": -17.56,
    "TradeCostsTotalInBaseCurrency": -11.64
  }
}
```

expected_status

response
response - get the response of the request.

status_code

SingleNetPositionDetails

class saxo_openapi.endpoints.portfolio.netpositions.**SingleNetPositionDetails**(*NetPositionId*, *params*)

Get a single net position details.

ENDPOINT = 'openapi/port/v1/netpositions/{NetPositionId}/details'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*NetPositionId*, *params*)
Instantiate a SingleNetPositionDetails request.

Parameters

- **NetPositionId** (*string (required)*) – the NetPositionId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
      {
        "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
      }
```

```
>>> NetPositionId = "GBPCAD__FxSpot"
>>> r = pf.positions.SingleNetPositionDetails(
...          NetPositionId=NetPositionId,
...          params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "NetPositionBase": {
    "AccountId": "9226397",
    "Amount": 100000,
    "AssetType": "FxSpot",
    "CanBeClosed": true,
    "ClientId": "9226397",
    "IsMarketOpen": true,
    "NumberOfRelatedOrders": 0,
    "PositionsAccount": "9226397",
    "SinglePositionId": "212550210",
    "SinglePositionStatus": "Open",
    "Uic": 23,
    "ValueDate": "2019-03-06T00:00:00.000000Z"
  },
  "NetPositionDetails": {
    "CloseCost": {
      "Commission": 8.77
    },
    "CloseCostInBaseCurrency": {
      "Commission": 5.81
    },
    "MarketValue": -478,
    "MarketValueInBaseCurrency": -316.77,
    "OpenCost": {
      "Commission": 8.79
    },
    "OpenCostInBaseCurrency": {
      "Commission": 5.83
    }
  },
  "NetPositionId": "GBPCAD__FxSpot",
  "NetPositionView": {
    "AverageOpenPrice": 1.75824,
    "CalculationReliability": "Ok",
    "CurrentPrice": 1.75346,
    "CurrentPriceDelayMinutes": 0,
    "CurrentPriceType": "Bid",
    "Exposure": 100000,
    "ExposureCurrency": "GBP",
    "ExposureInBaseCurrency": 116228,
    "InstrumentPriceDayPercentChange": -0.15,
    "PositionCount": 1,
    "PositionsNotClosedCount": 1,
    "ProfitLossOnTrade": -478,
    "ProfitLossOnTradeInBaseCurrency": -316.77,
    "Status": "Open",
    "TradeCostsTotal": -17.56,
    "TradeCostsTotalInBaseCurrency": -11.64
  }
}
```


expected_status

response
response - get the response of the request.

status_code

3.4.8 saxo_openapi.endpoints.portfolio.orders

CreateOpenOrdersSubscription

class saxo_openapi.endpoints.portfolio.orders.**CreateOpenOrdersSubscription**(*data*)

Set up a subscription and returns an initial snapshot of list of orders specified by the parameters in the request.

ENDPOINT = 'openapi/port/v1/orders/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)

Instantiate a CreateOpenOrdersSubscription request.

Parameters **data** (*dict (optional)*) – dict representing the body with request parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1552035128308",
          "ReferenceId": "C_582"
        }
```

```
>>> r = pf.orders.CreateOpenOrdersSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "ContextId": "explorer_1552035128308",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "C_582",
  "RefreshRate": 1000,
  "Snapshot": {
    "Data": [
      {
        "AccountId": "9226397",
        "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "BuySell": "Buy",
        "CalculationReliability": "Ok",
        "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
        "CorrelationKey": "925ab7aa-998b-4f4b-af26-7c1b3f4eee27",
        "CurrentPrice": 7.46175,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Ask",
        "DistanceToMarket": 0.46175,
        "Duration": {
```

(continues on next page)

(continued from previous page)


```
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": false,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76332324",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T14:54:17.423333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 16
    },
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "e684a2e2-f9f9-4b10-91ad-a50e2f8b80ae",
      "CurrentPrice": 7.46175,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.46175,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": false,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76328908",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T12:04:13.493333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 16
    },
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 10000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "ec6ca0f4-8380-4068-a32d-538b7b06208e",
      "CurrentPrice": 1.30179,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.20179,
      "Duration": {
        "DurationType": "GoodTillCancel"
```


(continues on next page)

(continued from previous page)


```
        },
        "IsMarketOpen": false,
        "MarketPrice": 1.30179,
        "OpenOrderType": "Limit",
        "OrderAmountType": "Quantity",
        "OrderId": "76289260",
        "OrderRelation": "StandAlone",
        "OrderTime": "2019-03-04T17:55:59.503333Z",
        "Price": 1.1,
        "RelatedOpenOrders": [],
        "Status": "Working",
        "Uic": 31
      }
    ]
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

GetAllOpenOrders

class saxo_openapi.endpoints.portfolio.orders.**GetAllOpenOrders**(*params=None*)
all the open orders on an account or a client.

ENDPOINT = 'openapi/port/v1/orders/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate a GetAllOpenOrders request.

Parameters **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.orders.GetAllOpenOrders(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "__count": 3,
  "Data": [
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "925ab7aa-998b-4f4b-af26-7c1b3f4eee27",
      "CurrentPrice": 7.46175,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.46175,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": false,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76332324",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T14:54:17.423333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 16
    },
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "e684a2e2-f9f9-4b10-91ad-a50e2f8b80ae",
      "CurrentPrice": 7.46175,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.46175,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": false,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76328908",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T12:04:13.493333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
```


(continues on next page)

(continued from previous page)


```
      "Uic": 16
    },
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 10000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "ec6ca0f4-8380-4068-a32d-538b7b06208e",
      "CurrentPrice": 1.30179,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.20179,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": false,
      "MarketPrice": 1.30179,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76289260",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-04T17:55:59.503333Z",
      "Price": 1.1,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 31
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

GetOpenOrder

class saxo_openapi.endpoints.portfolio.orders.**GetOpenOrder**(*ClientKey*, *OrderId*, *params=None*)

get a specific open order of a client. Unique id of the client.Unique id of the order.Specification of field groups to return. Default is empty.

ENDPOINT = 'openapi/port/v1/orders/{ClientKey}/{OrderId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ClientKey*, *OrderId*, *params=None*)
Instantiate a GetOpenOrder request.

Parameters

- **ClientKey** (*string (required)*) – the ClientKey

- **OrderId** (*string (required)*) – the OrderId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ClientKey = 'Cf4xZWiYL6W1nMKpygBLLA=='
>>> OrderId = '76332324'
>>> params =
        {}
```

```
>>> r = pf.orders.GetOpenOrder(ClientKey=ClientKey,
...                            OrderId=OrderId,
...                            params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "__count": 1,
  "Data": [
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "925ab7aa-998b-4f4b-af26-7c1b3f4eee27",
      "CurrentPrice": 7.46175,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.46175,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": true,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76332324",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T14:54:17.423333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 16
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

GetOpenOrdersMe

class saxo_openapi.endpoints.portfolio.orders.**GetOpenOrdersMe**(*params=None*)
get all the open orders on a client to which the logged in user belongs.

ENDPOINT = 'openapi/port/v1/orders/me/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate a GetOpenOrdersMe request.

Parameters **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.orders.GetOpenOrdersMe(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "__count": 3,
  "Data": [
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "925ab7aa-998b-4f4b-af26-7c1b3f4eee27",
      "CurrentPrice": 7.46175,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.46175,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": true,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76332324",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T14:54:17.423333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 16
```

(continues on next page)

(continued from previous page)


```
    },
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "e684a2e2-f9f9-4b10-91ad-a50e2f8b80ae",
      "CurrentPrice": 7.46175,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.46175,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": true,
      "MarketPrice": 7.46175,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76328908",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-07T12:04:13.493333Z",
      "Price": 7,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 16
    },
    {
      "AccountId": "9226397",
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 10000,
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "CalculationReliability": "Ok",
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "CorrelationKey": "ec6ca0f4-8380-4068-a32d-538b7b06208e",
      "CurrentPrice": 1.30195,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Ask",
      "DistanceToMarket": 0.20195,
      "Duration": {
        "DurationType": "GoodTillCancel"
      },
      "IsMarketOpen": true,
      "MarketPrice": 1.30195,
      "OpenOrderType": "Limit",
      "OrderAmountType": "Quantity",
      "OrderId": "76289260",
      "OrderRelation": "StandAlone",
      "OrderTime": "2019-03-04T17:55:59.503333Z",
      "Price": 1.1,
      "RelatedOpenOrders": [],
      "Status": "Working",
      "Uic": 31
    }
```


(continues on next page)

(continued from previous page)

```
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

OrderDetails

class saxo_openapi.endpoints.portfolio.orders.**OrderDetails**(*OrderId*, *params=None*)
Get detailed information about a single open order as specified by the query parameters.

ENDPOINT = 'openapi/port/v1/orders/{OrderId}/details/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*OrderId*, *params=None*)
Instantiate a OrderDetails request.

Parameters

- **OrderId** (*string (required)*) – the OrderId
- **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> OrderId = '76332324'
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.orders.OrderDetails(OrderId=OrderId, params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:

```
{
  "AccountId": "9226397",
  "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "Amount": 100000,
  "AssetType": "FxSpot",
  "BuySell": "Buy",
  "CalculationReliability": "Ok",
  "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "CorrelationKey": "925ab7aa-998b-4f4b-af26-7c1b3f4eee27",
  "CurrentPrice": 7.46175,
  "CurrentPriceDelayMinutes": 0,
```

(continues on next page)

(continued from previous page)


```
  "CurrentPriceType": "Ask",
  "DisplayAndFormat": {
    "Currency": "DKK",
    "Decimals": 4,
    "Description": "Euro/Danish Krone",
    "Format": "AllowDecimalPips",
    "Symbol": "EURDKK"
  },
  "DistanceToMarket": 0.46175,
  "Duration": {
    "DurationType": "GoodTillCancel"
  },
  "Exchange": {
    "Description": "Inter Bank",
    "ExchangeId": "SBFX",
    "IsOpen": false
  },
  "IsMarketOpen": false,
  "MarketPrice": 7.46175,
  "OpenOrderType": "Limit",
  "OrderAmountType": "Quantity",
  "OrderId": "76332324",
  "OrderRelation": "StandAlone",
  "OrderTime": "2019-03-07T14:54:17.423333Z",
  "Price": 7,
  "RelatedOpenOrders": [],
  "Status": "Working",
  "Uic": 16
}
```


expected_status

response
response - get the response of the request.

status_code

RemoveOpenOrderSubscription

class saxo_openapi.endpoints.portfolio.orders.**RemoveOpenOrderSubscription**(*ContextId*, *ReferenceId*)

Remove a subscription for the current session identified by subscription id

ENDPOINT = 'openapi/port/v1/orders/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a RemoveOpenOrderSubscription request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'explorer_1552035128308'
>>> ReferenceId = 'C_582'
>>> r = pf.orders.RemoveOpenOrderSubscription(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

RemoveOpenOrderSubscriptionsByTag

class saxo_openapi.endpoints.portfolio.orders.**RemoveOpenOrderSubscriptionsByTag**(*ContextId*, *params=None*)

Remove multiple subscriptions for the current session on this resource. Optionally with with specified Tag.

ENDPOINT = 'openapi/port/v1/orders/subscriptions/{ContextId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params=None*)
Instantiate a RemoveOpenOrderSubscriptionsByTag request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'explorer_1552035128308'
>>> params = {_v3_RemoveOpenOrderSubscriptionsByTag_params}
>>> r = pf.orders.RemoveOpenOrderSubscriptionsByTag(ContextId,
...                                                 params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.4.9 saxo_openapi.endpoints.portfolio.positions

PositionListSubscription

class saxo_openapi.endpoints.portfolio.positions.**PositionListSubscription**(*data*, *params=None*)

Sets up a subscription and returns an initial snapshot of list of positions specified by the parameters in the request.

ENDPOINT = 'openapi/port/v1/positions/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*, *params=None*)
Instantiate a PositionListSubscription request.

Parameters

- **params** (*dict (optional)*) – dict representing the querystring parameters
- **data** (*dict (required)*) – dict representing the data body parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> data =
        {
          "Arguments": {
            "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
          },
          "ContextId": "explorer_1551702571343",
          "ReferenceId": "C_702"
        }
```

```
>>> r = pf.positions.PositionListSubscription(data=data,
...                                           params=params)
>>> # without params: pf.positions.PositionListSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "ContextId": "explorer_1551702571343",
  "Format": "application/json",
  "InactivityTimeout": 30,
  "ReferenceId": "C_702",
  "RefreshRate": 1000,
  "Snapshot": {
```

(continues on next page)

(continued from previous page)


```
  "Data": [
    {
      "NetPositionId": "EURUSD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": -100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "46dc6b2a-5b6f-43c8-b747-6b530da9110e",
        "ExecutionTimeOpen": "2019-03-04T00:10:23.040641Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.13715,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271915",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "PositionId": "212561926",
      "PositionView": {
        "CalculationReliability": "Ok",
        "ConversionRateCurrent": 0.883135,
        "ConversionRateOpen": 0.883135,
        "CurrentPrice": 1.13243,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Ask",
        "Exposure": -100000,
        "ExposureCurrency": "EUR",
        "ExposureInBaseCurrency": -100000,
        "InstrumentPriceDayPercentChange": -0.37,
        "ProfitLossOnTrade": 472,
        "ProfitLossOnTradeInBaseCurrency": 416.84,
        "TradeCostsTotal": -11.35,
        "TradeCostsTotalInBaseCurrency": -10.02
      }
    },
    {
      "NetPositionId": "EURUSD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "50fae087-b7d4-49ab-afa2-5145cd56a7c5",
        "ExecutionTimeOpen": "2019-03-04T00:04:11.340151Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.1371,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271912",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 21,
```


(continues on next page)

(continued from previous page)


```
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "PositionId": "212561892",
    "PositionView": {
      "CalculationReliability": "Ok",
      "ConversionRateCurrent": 0.883135,
      "ConversionRateOpen": 0.883135,
      "CurrentPrice": 1.13223,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 100000,
      "ExposureCurrency": "EUR",
      "ExposureInBaseCurrency": 100000,
      "InstrumentPriceDayPercentChange": -0.37,
      "ProfitLossOnTrade": -487,
      "ProfitLossOnTradeInBaseCurrency": -430.09,
      "TradeCostsTotal": -11.35,
      "TradeCostsTotalInBaseCurrency": -10.02
    }
  },
  {
    "NetPositionId": "GBPAUD__FxSpot",
    "PositionBase": {
      "AccountId": "9226397",
      "Amount": 500000,
      "AssetType": "FxSpot",
      "CanBeClosed": true,
      "ClientId": "9226397",
      "CloseConversionRateSettled": false,
      "CorrelationKey": "206cceed-2240-43f8-8c46-840e8b722549",
      "ExecutionTimeOpen": "2019-03-03T23:35:08.243690Z",
      "IsMarketOpen": true,
      "OpenPrice": 1.86391,
      "RelatedOpenOrders": [],
      "SourceOrderId": "76271862",
      "SpotDate": "2019-03-06",
      "Status": "Open",
      "Uic": 22,
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "PositionId": "212550212",
    "PositionView": {
      "CalculationReliability": "Ok",
      "ConversionRateCurrent": 0.625415,
      "ConversionRateOpen": 0.625415,
      "CurrentPrice": 1.86215,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 500000,
      "ExposureCurrency": "GBP",
      "ExposureInBaseCurrency": 582455,
      "InstrumentPriceDayPercentChange": -0.14,
      "ProfitLossOnTrade": -880,
      "ProfitLossOnTradeInBaseCurrency": -550.37,
      "TradeCostsTotal": -93.15,
      "TradeCostsTotalInBaseCurrency": -58.26
    }
```


(continues on next page)

(continued from previous page)


```
      },
      {
        "NetPositionId": "GBPCAD__FxSpot",
        "PositionBase": {
          "AccountId": "9226397",
          "Amount": 100000,
          "AssetType": "FxSpot",
          "CanBeClosed": true,
          "ClientId": "9226397",
          "CloseConversionRateSettled": false,
          "CorrelationKey": "19c44107-6858-4191-805c-764a69d27491",
          "ExecutionTimeOpen": "2019-03-03T23:34:51.823660Z",
          "IsMarketOpen": true,
          "OpenPrice": 1.75824,
          "RelatedOpenOrders": [],
          "SourceOrderId": "76271861",
          "SpotDate": "2019-03-06",
          "Status": "Open",
          "Uic": 23,
          "ValueDate": "2019-03-06T00:00:00.000000Z"
        },
        "PositionId": "212550210",
        "PositionView": {
          "CalculationReliability": "Ok",
          "ConversionRateCurrent": 0.66362,
          "ConversionRateOpen": 0.66362,
          "CurrentPrice": 1.75496,
          "CurrentPriceDelayMinutes": 0,
          "CurrentPriceType": "Bid",
          "Exposure": 100000,
          "ExposureCurrency": "GBP",
          "ExposureInBaseCurrency": 116491,
          "InstrumentPriceDayPercentChange": -0.07,
          "ProfitLossOnTrade": -328,
          "ProfitLossOnTradeInBaseCurrency": -217.67,
          "TradeCostsTotal": -17.56,
          "TradeCostsTotalInBaseCurrency": -11.65
        }
      }
    ],
    "MaxRows": 100000
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

PositionSubscriptionPageSize

class saxo_openapi.endpoints.portfolio.positions.**PositionSubscriptionPageSize**(*ContextId*, *ReferenceId*, *data*)

Extends or reduces the page size, number of positions shown, on a running positions subscription. When expanding the page size, the new positions are streamed so to avoid race conditions.

ENDPOINT = 'openapi/port/v1/positions/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*, *data*)

Instantiate a PositionSubscriptionPageSize request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId
- **data** (*dict (required)*) – dict representing the data body parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ...
>>> ReferenceId = ...
>>> data =
        {
          "NewPageSize": 25630
        }
```

```
>>> r = pf.positions.PositionSubscriptionPageSize(ContextId,
...                                               ReferenceId,
...                                               data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

PositionSubscriptionRemove

class saxo_openapi.endpoints.portfolio.positions.**PositionSubscriptionRemove**(*ContextId*, *ReferenceId*)

Removes subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/port/v1/positions/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)

Instantiate a PositionSubscriptionPageSize request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ...
>>> ReferenceId = ...
>>> r = pf.positions.PositionSubscriptionRemove(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

PositionSubscriptionRemoveMultiple

class saxo_openapi.endpoints.portfolio.positions.**PositionSubscriptionRemoveMultiple**(*ContextId*, *params=None*)

Remove multiple subscriptions for the given ContextId, optionally marked with a specific tag.

ENDPOINT = 'openapi/port/v1/positions/subscriptions/{ContextId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

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

Instantiate a PositionSubscriptionRemoveMultiple request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "Tag": "..."
        }
```

```
>>> ContextId = ...
>>> r = pf.positions.PositionSubscriptionRemoveMultiple(ContextId,
...                                                     params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

PositionsMe

class saxo_openapi.endpoints.portfolio.positions.**PositionsMe**(*params=None*)
Get positions for the logged-in client.

ENDPOINT = 'openapi/port/v1/positions/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate a PositionsMe request.

Parameters **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> r = pf.positions.PositionsMe()
>>> # or with params: pf.positions.PositionsMe(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "__count": 4,
  "Data": [
```

(continues on next page)

(continued from previous page)


```
    {
      "NetPositionId": "EURUSD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": -100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "46dc6b2a-5b6f-43c8-b747-6b530da9110e",
        "ExecutionTimeOpen": "2019-03-04T00:10:23.040641Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.13715,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271915",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "PositionId": "212561926",
      "PositionView": {
        "CalculationReliability": "Ok",
        "ConversionRateCurrent": 0.882595,
        "ConversionRateOpen": 0.882595,
        "CurrentPrice": 1.13312,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Ask",
        "Exposure": -100000,
        "ExposureCurrency": "EUR",
        "ExposureInBaseCurrency": -100000,
        "InstrumentPriceDayPercentChange": -0.31,
        "ProfitLossOnTrade": 403,
        "ProfitLossOnTradeInBaseCurrency": 355.69,
        "TradeCostsTotal": -11.36,
        "TradeCostsTotalInBaseCurrency": -10.03
      }
    },
    {
      "NetPositionId": "EURUSD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "50fae087-b7d4-49ab-afa2-5145cd56a7c5",
        "ExecutionTimeOpen": "2019-03-04T00:04:11.340151Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.1371,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271912",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
```


(continues on next page)

(continued from previous page)


```
    },
    "PositionId": "212561892",
    "PositionView": {
      "CalculationReliability": "Ok",
      "ConversionRateCurrent": 0.882595,
      "ConversionRateOpen": 0.882595,
      "CurrentPrice": 1.13292,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 100000,
      "ExposureCurrency": "EUR",
      "ExposureInBaseCurrency": 100000,
      "InstrumentPriceDayPercentChange": -0.31,
      "ProfitLossOnTrade": -418,
      "ProfitLossOnTradeInBaseCurrency": -368.92,
      "TradeCostsTotal": -11.35,
      "TradeCostsTotalInBaseCurrency": -10.02
    }
  },
  {
    "NetPositionId": "GBPAUD__FxSpot",
    "PositionBase": {
      "AccountId": "9226397",
      "Amount": 500000,
      "AssetType": "FxSpot",
      "CanBeClosed": true,
      "ClientId": "9226397",
      "CloseConversionRateSettled": false,
      "CorrelationKey": "206cceed-2240-43f8-8c46-840e8b722549",
      "ExecutionTimeOpen": "2019-03-03T23:35:08.243690Z",
      "IsMarketOpen": true,
      "OpenPrice": 1.86391,
      "RelatedOpenOrders": [],
      "SourceOrderId": "76271862",
      "SpotDate": "2019-03-06",
      "Status": "Open",
      "Uic": 22,
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "PositionId": "212550212",
    "PositionView": {
      "CalculationReliability": "Ok",
      "ConversionRateCurrent": 0.6254,
      "ConversionRateOpen": 0.6254,
      "CurrentPrice": 1.85999,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 500000,
      "ExposureCurrency": "GBP",
      "ExposureInBaseCurrency": 581757.5,
      "InstrumentPriceDayPercentChange": -0.25,
      "ProfitLossOnTrade": -1960,
      "ProfitLossOnTradeInBaseCurrency": -1225.78,
      "TradeCostsTotal": -93.1,
      "TradeCostsTotalInBaseCurrency": -58.22
    }
  },
```


(continues on next page)

(continued from previous page)


```
    {
      "NetPositionId": "GBPCAD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "19c44107-6858-4191-805c-764a69d27491",
        "ExecutionTimeOpen": "2019-03-03T23:34:51.823660Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.75824,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271861",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 23,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "PositionId": "212550210",
      "PositionView": {
        "CalculationReliability": "Ok",
        "ConversionRateCurrent": 0.663595,
        "ConversionRateOpen": 0.663595,
        "CurrentPrice": 1.75294,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 100000,
        "ExposureCurrency": "GBP",
        "ExposureInBaseCurrency": 116351.5,
        "InstrumentPriceDayPercentChange": -0.18,
        "ProfitLossOnTrade": -530,
        "ProfitLossOnTradeInBaseCurrency": -351.71,
        "TradeCostsTotal": -17.55,
        "TradeCostsTotalInBaseCurrency": -11.65
      }
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

PositionsQuery

class saxo_openapi.endpoints.portfolio.positions.**PositionsQuery**(*params*)
Get positions for a client, account group, account or position. Returns a list of positions fulfilling the criteria specified by the query string parameters.

ENDPOINT = 'openapi/port/v1/positions'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)

Instantiate a PositionsQuery request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.positions.PositionsQuery(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "__count": 4,
  "Data": [
    {
      "NetPositionId": "EURUSD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": -100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "46dc6b2a-5b6f-43c8-b747-6b530da9110e",
        "ExecutionTimeOpen": "2019-03-04T00:10:23.040641Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.13715,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271915",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "PositionId": "212561926",
      "PositionView": {
        "CalculationReliability": "Ok",
        "ConversionRateCurrent": 0.882905,
        "ConversionRateOpen": 0.882905,
        "CurrentPrice": 1.13273,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Ask",
        "Exposure": -100000,
        "ExposureCurrency": "EUR",
        "ExposureInBaseCurrency": -100000,
        "InstrumentPriceDayPercentChange": -0.34,
        "ProfitLossOnTrade": 442,
        "ProfitLossOnTradeInBaseCurrency": 390.24,
        "TradeCostsTotal": -11.35,
```


(continues on next page)

(continued from previous page)


```
        "TradeCostsTotalInBaseCurrency": -10.02
      }
    },
    {
      "NetPositionId": "EURUSD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": 100000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "50fae087-b7d4-49ab-afa2-5145cd56a7c5",
        "ExecutionTimeOpen": "2019-03-04T00:04:11.340151Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.1371,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271912",
        "SpotDate": "2019-03-06",
        "Status": "Open",
        "Uic": 21,
        "ValueDate": "2019-03-06T00:00:00.000000Z"
      },
      "PositionId": "212561892",
      "PositionView": {
        "CalculationReliability": "Ok",
        "ConversionRateCurrent": 0.882905,
        "ConversionRateOpen": 0.882905,
        "CurrentPrice": 1.13253,
        "CurrentPriceDelayMinutes": 0,
        "CurrentPriceType": "Bid",
        "Exposure": 100000,
        "ExposureCurrency": "EUR",
        "ExposureInBaseCurrency": 100000,
        "InstrumentPriceDayPercentChange": -0.34,
        "ProfitLossOnTrade": -457,
        "ProfitLossOnTradeInBaseCurrency": -403.49,
        "TradeCostsTotal": -11.35,
        "TradeCostsTotalInBaseCurrency": -10.02
      }
    },
    {
      "NetPositionId": "GBPAUD__FxSpot",
      "PositionBase": {
        "AccountId": "9226397",
        "Amount": 500000,
        "AssetType": "FxSpot",
        "CanBeClosed": true,
        "ClientId": "9226397",
        "CloseConversionRateSettled": false,
        "CorrelationKey": "206cceed-2240-43f8-8c46-840e8b722549",
        "ExecutionTimeOpen": "2019-03-03T23:35:08.243690Z",
        "IsMarketOpen": true,
        "OpenPrice": 1.86391,
        "RelatedOpenOrders": [],
        "SourceOrderId": "76271862",
        "SpotDate": "2019-03-06",
```


(continues on next page)

(continued from previous page)


```
      "Status": "Open",
      "Uic": 22,
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "PositionId": "212550212",
    "PositionView": {
      "CalculationReliability": "Ok",
      "ConversionRateCurrent": 0.62534,
      "ConversionRateOpen": 0.62534,
      "CurrentPrice": 1.86127,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 500000,
      "ExposureCurrency": "GBP",
      "ExposureInBaseCurrency": 582115,
      "InstrumentPriceDayPercentChange": -0.19,
      "ProfitLossOnTrade": -1320,
      "ProfitLossOnTradeInBaseCurrency": -825.45,
      "TradeCostsTotal": -93.13,
      "TradeCostsTotalInBaseCurrency": -58.24
    }
  },
  {
    "NetPositionId": "GBPCAD__FxSpot",
    "PositionBase": {
      "AccountId": "9226397",
      "Amount": 100000,
      "AssetType": "FxSpot",
      "CanBeClosed": true,
      "ClientId": "9226397",
      "CloseConversionRateSettled": false,
      "CorrelationKey": "19c44107-6858-4191-805c-764a69d27491",
      "ExecutionTimeOpen": "2019-03-03T23:34:51.823660Z",
      "IsMarketOpen": true,
      "OpenPrice": 1.75824,
      "RelatedOpenOrders": [],
      "SourceOrderId": "76271861",
      "SpotDate": "2019-03-06",
      "Status": "Open",
      "Uic": 23,
      "ValueDate": "2019-03-06T00:00:00.000000Z"
    },
    "PositionId": "212550210",
    "PositionView": {
      "CalculationReliability": "Ok",
      "ConversionRateCurrent": 0.66389,
      "ConversionRateOpen": 0.66389,
      "CurrentPrice": 1.75321,
      "CurrentPriceDelayMinutes": 0,
      "CurrentPriceType": "Bid",
      "Exposure": 100000,
      "ExposureCurrency": "GBP",
      "ExposureInBaseCurrency": 116423,
      "InstrumentPriceDayPercentChange": -0.17,
      "ProfitLossOnTrade": -503,
      "ProfitLossOnTradeInBaseCurrency": -333.94,
      "TradeCostsTotal": -17.56,
```


(continues on next page)

(continued from previous page)

```
        "TradeCostsTotalInBaseCurrency": -11.66
      }
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

SinglePosition

class saxo_openapi.endpoints.portfolio.positions.**SinglePosition**(*PositionId*, *params*)

Get a single position.

ENDPOINT = 'openapi/port/v1/positions/{PositionId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*PositionId*, *params*)
Instantiate a SinglePosition request.

Parameters

- **PositionId** (*string (required)*) – the PositionId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.positions.SinglePosition(PositionId=212561926,
...                                 params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "NetPositionId": "EURUSD__FxSpot",
  "PositionBase": {
    "AccountId": "9226397",
    "Amount": -100000,
    "AssetType": "FxSpot",
    "CanBeClosed": true,
    "ClientId": "9226397",
    "CloseConversionRateSettled": false,
    "CorrelationKey": "46dc6b2a-5b6f-43c8-b747-6b530da9110e",
```

(continues on next page)

(continued from previous page)


```
    "ExecutionTimeOpen": "2019-03-04T00:10:23.040641Z",
    "IsMarketOpen": true,
    "OpenPrice": 1.13715,
    "RelatedOpenOrders": [],
    "SourceOrderId": "76271915",
    "SpotDate": "2019-03-06",
    "Status": "Open",
    "Uic": 21,
    "ValueDate": "2019-03-06T00:00:00.000000Z"
  },
  "PositionId": "212561926",
  "PositionView": {
    "CalculationReliability": "Ok",
    "ConversionRateCurrent": 0.88199,
    "ConversionRateOpen": 0.88199,
    "CurrentPrice": 1.1339,
    "CurrentPriceDelayMinutes": 0,
    "CurrentPriceType": "Ask",
    "Exposure": -100000,
    "ExposureCurrency": "EUR",
    "ExposureInBaseCurrency": -100000,
    "InstrumentPriceDayPercentChange": -0.24,
    "ProfitLossOnTrade": 325,
    "ProfitLossOnTradeInBaseCurrency": 286.65,
    "TradeCostsTotal": -11.36,
    "TradeCostsTotalInBaseCurrency": -10.02
  }
}
```


expected_status

response
response - get the response of the request.

status_code

SinglePositionDetails

class saxo_openapi.endpoints.portfolio.positions.**SinglePositionDetails**(*PositionId*, *params*)

Get a single position details.

ENDPOINT = 'openapi/port/v1/positions/{PositionId}/details'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*PositionId*, *params*)
Instantiate a SinglePositionDetails request.

Parameters

- **PositionId** (*string (required)*) – the PositionId
- **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
          "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.positions.SinglePositionDetails(PositionId=212561926,
...                                        params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "DisplayAndFormat": {
    "Currency": "USD",
    "Decimals": 4,
    "Description": "Euro/US Dollar",
    "Format": "AllowDecimalPips",
    "Symbol": "EURUSD"
  },
  "Exchange": {
    "Description": "Inter Bank",
    "ExchangeId": "SBFX",
    "IsOpen": true
  },
  "NetPositionId": "EURUSD__FxSpot",
  "PositionBase": {
    "AccountId": "9226397",
    "Amount": -100000,
    "AssetType": "FxSpot",
    "CanBeClosed": true,
    "ClientId": "9226397",
    "CloseConversionRateSettled": false,
    "CorrelationKey": "46dc6b2a-5b6f-43c8-b747-6b530da9110e",
    "ExecutionTimeOpen": "2019-03-04T00:10:23.040641Z",
    "IsMarketOpen": true,
    "OpenPrice": 1.13715,
    "RelatedOpenOrders": [],
    "SourceOrderId": "76271915",
    "SpotDate": "2019-03-06",
    "Status": "Open",
    "Uic": 21,
    "ValueDate": "2019-03-06T00:00:00.000000Z"
  },
  "PositionDetails": {
    "CloseCost": {
      "Commission": 5.67
    },
    "CloseCostInBaseCurrency": {
      "Commission": 5
    },
    "CorrelationKey": "46dc6b2a-5b6f-43c8-b747-6b530da9110e",
    "LockedByBackOffice": false,
    "MarketValue": 351,
```


(continues on next page)

(continued from previous page)


```
    "OpenCost": {
      "Commission": 5.69
    },
    "OpenCostInBaseCurrency": {
      "Commission": 5.02
    },
    "SourceOrderId": "76271915"
  },
  "PositionId": "212561926",
  "PositionView": {
    "CalculationReliability": "Ok",
    "ConversionRateCurrent": 0.882195,
    "ConversionRateOpen": 0.882195,
    "CurrentPrice": 1.13364,
    "CurrentPriceDelayMinutes": 0,
    "CurrentPriceType": "Ask",
    "Exposure": -100000,
    "ExposureCurrency": "EUR",
    "ExposureInBaseCurrency": -100000,
    "InsTrumentPriceDayPercentChange": -0.26,
    "ProfitLossOnTrade": 351,
    "ProfitLossOnTradeInBaseCurrency": 309.65,
    "TradeCostsTotal": -11.36,
    "TradeCostsTotalInBaseCurrency": -10.02
  }
}
```


expected_status

response
response - get the response of the request.

status_code

3.4.10 saxo_openapi.endpoints.portfolio.users

UserDetails

class saxo_openapi.endpoints.portfolio.users.**UserDetails**(*UserKey*)
Get the details about a user.

ENDPOINT = 'openapi/port/v1/users/{UserKey}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*UserKey*)
Instantiate a UserDetails request.

Parameters **UserKey** (*string (required)*) – the UserKey

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> UserKey = '...'
```

(continues on next page)

(continued from previous page)

```
>>> r = pf.users.Users(UserKey=UserKey)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "Culture": "en-GB",
  "Language": "en",
  "LastLoginStatus": "Successful",
  "LastLoginTime": "2019-03-05T13:22:22.123000Z",
  "LegalAssetTypes": [
    "FxSpot",
    "FxForwards",
    "FxVanillaOption",
    "FxKnockInOption",
    "FxKnockOutOption",
    "FxOneTouchOption",
    "FxNoTouchOption"
  ],
  "MarketDataViaOpenApiTermsAccepted": false,
  "Name": "F. Brekeveld",
  "TimeZoneId": 26,
  "UserId": "9226397",
  "UserKey": "Cf4xZWiYL6W1nMKpygBLLA=="
}
```

expected_status

response
response - get the response of the request.

status_code

UserUpdate

class saxo_openapi.endpoints.portfolio.users.**UserUpdate**(*data*)
Enables the user to update preferred language, culture and timezone.

ENDPOINT = 'openapi/port/v1/users/me'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*data*)
Instantiate a UserUpdate request.

Parameters **data** (*dict (required)*) – dict representing the data body parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
```

(continues on next page)

(continued from previous page)

```
        {
          "Language": "nl"
        }
```

```
>>> r = pf.users.UserUpdate(data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Users

class saxo_openapi.endpoints.portfolio.users.**Users**(*params*)
Get all users under a particular owner.

ENDPOINT = 'openapi/port/v1/users'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a Users request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = pf.users.Users(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Data": [
    {
      "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Culture": "en-GB",
      "Language": "en",
      "LastLoginStatus": "Successful",
      "LastLoginTime": "2019-03-05T13:22:22.123000Z",
      "LegalAssetTypes": [
```

(continues on next page)

(continued from previous page)


```
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption",
        "FxOneTouchOption",
        "FxNoTouchOption"
      ],
      "MarketDataViaOpenApiTermsAccepted": false,
      "Name": "F. Brekeveld",
      "TimeZoneId": 26,
      "UserId": "9226397",
      "UserKey": "Cf4xZWiYL6W1nMKpygBLLA=="
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

UsersMe

class saxo_openapi.endpoints.portfolio.users.**UsersMe**
Get details about the logged in user.

ENDPOINT = 'openapi/port/v1/users/me'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a UsersMe request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.portfolio as pf
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = pf.users.UsersMe()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "ClientKey": "Cf4xZWiYL6W1nMKpygBLLA==",
  "Culture": "en-GB",
  "Language": "en",
  "LastLoginStatus": "Successful",
  "LastLoginTime": "2019-03-05T13:22:22.123000Z",
  "LegalAssetTypes": [
    "FxSpot",
    "FxForwards",
```


(continues on next page)

(continued from previous page)

```
    "FxVanillaOption",
    "FxKnockInOption",
    "FxKnockOutOption",
    "FxOneTouchOption",
    "FxNoTouchOption"
  ],
  "MarketDataViaOpenApiTermsAccepted": false,
  "Name": "F. Brekeveld",
  "TimeZoneId": 26,
  "UserId": "9226397",
  "UserKey": "Cf4xZWiYL6W1nMKpygBLLA=="
}
```

expected_status

response
response - get the response of the request.

status_code

3.5 saxo_openapi.endpoints.referencedata

3.5.1 saxo_openapi.endpoints.referencedata.algostrategies

AlgoStrategies

class saxo_openapi.endpoints.referencedata.algostrategies.**AlgoStrategies**(*params=None*)
Retrieve a list of strategies with detailed information about each strategy. The response also contains links to other relevant data, such as their parameters.

ENDPOINT = 'openapi/ref/v1/algostrategies/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)
Instantiate an AlgoStrategies request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "$top": "...",
          "$skip": "..."
        }
```

```
>>> r = rd.algostrategies.AlgoStrategies(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "__count": 4,
  "Data": [
    {
      "Description": "Group of VWAP",
      "MinAmountUSD": 0,
      "Name": "VWAP",
      "Parameters": [],
      "SupportedDurationTypes": [
        "DayOrder"
      ],
      "TradableInstrumentTypes": []
    },
    {
      "Description": "Groups of Iceberg Strategies",
      "MinAmountUSD": 0,
      "Name": "Iceberg",
      "Parameters": [],
      "SupportedDurationTypes": [
        "DayOrder"
      ],
      "TradableInstrumentTypes": []
    },
    {
      "Description": "Group of With Volume strategies",
      "MinAmountUSD": 0,
      "Name": "With Volume",
      "Parameters": [],
      "SupportedDurationTypes": [
        "DayOrder"
      ],
      "TradableInstrumentTypes": []
    },
    {
      "Description": "Group of IS strategies",
      "MinAmountUSD": 0,
      "Name": "Implementation Shortfall",
      "Parameters": [],
      "SupportedDurationTypes": [
        "DayOrder"
      ],
      "TradableInstrumentTypes": []
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

AlgoStrategyDetails

class saxo_openapi.endpoints.referencedata.algostrategies.**AlgoStrategyDetails**(*Name*)
Retrieve detailed information about a specific Strategy.

ENDPOINT = 'openapi/ref/v1/algostrategies/{Name}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*Name*)
Instantiate an AlgoStrategyDetails request.

Parameters **Name** (*string (required)*) – Name of the strategy

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> Name = "Implementation Shortfall"
>>> r = rd.algostrategies.AlgoStrategyDetails(Name=Name)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Description": "Group of IS strategies",
  "MinAmountUSD": 0,
  "Name": "Implementation Shortfall",
  "Parameters": [],
  "SupportedDurationTypes": [
    "DayOrder"
  ],
  "TradableInstrumentTypes": []
}
```

expected_status

response
response - get the response of the request.

status_code

3.5.2 saxo_openapi.endpoints.referencedata.countries

Countries

class saxo_openapi.endpoints.referencedata.countries.**Countries**
Retrieve a list all the countries supported by Saxo Bank.

ENDPOINT = 'openapi/ref/v1/countries/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a Countries request.

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
```

(continues on next page)

(continued from previous page)

```
>>> client = saxo_openapi.API(access_token=...)
>>> r = rd.countries.Countries()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "Data": [
    {
      "A3": "AFG",
      "CountryCode": "AF",
      "Name": "Afghanistan",
      "Numeric": 4
    },
    {
      "A3": "ALA",
      "CountryCode": "AX",
      "Name": "Aland Islands",
      "Numeric": 248
    },
    {
      "A3": "ALB",
      "CountryCode": "AL",
      "Name": "Albania",
      "Numeric": 8
    },
    {
      "A3": "..."
    },
    {
      "A3": "VNM",
      "CountryCode": "VN",
      "Name": "Vietnam",
      "Numeric": 704
    },
    {
      "A3": "VIR",
      "CountryCode": "VI",
      "Name": "Virgin Islands (U.S.)",
      "Numeric": 850
    },
    {
      "A3": "WLF",
      "CountryCode": "WF",
      "Name": "Wallis and Futuna Islands",
      "Numeric": 876
    },
    {
      "A3": "ESH",
      "CountryCode": "EH",
      "Name": "Western Sahara",
      "Numeric": 732
    },
    {
      "A3": "YEM",
      "CountryCode": "YE",
      "Name": "Yemen",
```


(continues on next page)

(continued from previous page)


```
      "Numeric": 887
    },
    {
      "A3": "ZMB",
      "CountryCode": "ZM",
      "Name": "Zambia",
      "Numeric": 894
    },
    {
      "A3": "ZWE",
      "CountryCode": "ZW",
      "Name": "Zimbabwe",
      "Numeric": 716
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.5.3 saxo_openapi.endpoints.referencedata.cultures

Cultures

class saxo_openapi.endpoints.referencedata.cultures.**Cultures**
Get a list all the cultures for user preference localization supported by Saxo Bank.

ENDPOINT = 'openapi/ref/v1/cultures/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a Cultures request.

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> r = rd.Cultures()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```


```
{
  "Data": [
    {
      "CultureCode": "af-ZA",
      "Name": "Afrikaans (South Africa)"
    },
    {
      "CultureCode": "am-ET",
```


(continues on next page)

(continued from previous page)


```
      "Name": "Amharic (Ethiopia)"
    },
    {
      "CultureCode": "ar-AE",
      "Name": "Arabic (U.A.E.)"
    },
    {
      "CultureCode": "ar-BH",
      "Name": "Arabic (Bahrain)"
    },
    {
      "CultureCode": "ar-DZ",
      "Name": "Arabic (Algeria)"
    },
    {
      "CultureCode": "...",
      "Name": "..."
    },
    {
      "CultureCode": "zh-CN",
      "Name": "Chinese (Simplified, PRC)"
    },
    {
      "CultureCode": "zh-HK",
      "Name": "Chinese (Traditional, Hong Kong S.A.R.)"
    },
    {
      "CultureCode": "zh-MO",
      "Name": "Chinese (Traditional, Macao S.A.R.)"
    },
    {
      "CultureCode": "zh-SG",
      "Name": "Chinese (Simplified, Singapore)"
    },
    {
      "CultureCode": "zh-TW",
      "Name": "Chinese (Traditional, Taiwan)"
    },
    {
      "CultureCode": "zu-ZA",
      "Name": "isiZulu (South Africa)"
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.5.4 saxo_openapi.endpoints.referencedata.currencies

Currencies

class saxo_openapi.endpoints.referencedata.currencies.**Currencies**

Get a list all supported currencies.

ENDPOINT = 'openapi/ref/v1/currencies/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()

Instantiate a Currencies request.

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> r = rd.currencies.Currencies()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "Data": [
    {
      "CurrencyCode": "USD",
      "Decimals": 2,
      "Name": "US Dollar",
      "Symbol": "$"
    },
    {
      "CurrencyCode": "GBP",
      "Decimals": 2,
      "Name": "British Pound",
      "Symbol": "\u00a3"
    },
    {
      "CurrencyCode": "EUR",
      "Decimals": 2,
      "Name": "Euro",
      "Symbol": "\u20ac"
    },
    {
      "CurrencyCode": "CHF",
      "Decimals": 2,
      "Name": "Swiss Franc",
      "Symbol": "Fr."
    },
    {
      "CurrencyCode": "AUD",
      "Decimals": 2,
      "Name": "Australian Dollar",
      "Symbol": "$"
    },
    {
      "CurrencyCode": "CAD",
      "Decimals": 2,
      "Name": "Canadian Dollar",
      "Symbol": "$"
```

(continues on next page)

(continued from previous page)


```
    },
    {
      "CurrencyCode": "NZD",
      "Decimals": 2,
      "Name": "New Zealand Dollar",
      "Symbol": "$"
    },
    {
      "CurrencyCode": "JPY",
      "Decimals": 0,
      "Name": "Japanese Yen",
      "Symbol": "\u00a5"
    },
    {
      "CurrencyCode": "DKK",
      "Decimals": 2,
      "Name": "Danish Krone",
      "Symbol": "kr."
    },
    {
      "CurrencyCode": "SEK",
      "Decimals": 2,
      "Name": "Swedish Krona",
      "Symbol": "kr"
    },
    {
      "CurrencyCode": "..."
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.5.5 saxo_openapi.endpoints.referencedata.exchanges

ExchangeDetails

class saxo_openapi.endpoints.referencedata.exchanges.**ExchangeDetails**(*ExchangeId*)
Retrieves detailed information about a specific exchange.

ENDPOINT = 'openapi/ref/v1/exchanges/{ExchangeId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ExchangeId*)
Instantiate an ExchangeDetails request.

Parameters **ExchangeId** (*string (required)*) – the ExchangeId

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> ExchangeId = '...'
>>> r = rd.ExchangeDetails(ExchangeId=ExchangeId)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:


```
{
  "AllDay": false,
  "CountryCode": "US",
  "Currency": "USD",
  "ExchangeId": "NYSE_ARCA",
  "ExchangeSessions": [
    {
      "EndTime": "2019-03-04T12:00:00.000000Z",
      "StartTime": "2019-03-01T21:00:00.000000Z",
      "State": "Closed"
    },
    {
      "EndTime": "2019-03-04T14:30:00.000000Z",
      "StartTime": "2019-03-04T12:00:00.000000Z",
      "State": "PreTrading"
    },
    {
      "EndTime": "2019-03-04T21:00:00.000000Z",
      "StartTime": "2019-03-04T14:30:00.000000Z",
      "State": "AutomatedTrading"
    },
    {
      "EndTime": "2019-03-05T12:00:00.000000Z",
      "StartTime": "2019-03-04T21:00:00.000000Z",
      "State": "Closed"
    }
  ],
  "Mic": "ARCX",
  "Name": "New York Stock Exchange (ARCA)",
  "TimeZone": 3,
  "TimeZoneAbbreviation": "EST",
  "TimeZoneOffset": "-05:00:00"
}
```


expected_status

response
response - get the response of the request.

status_code

ExchangeList

class saxo_openapi.endpoints.referencedata.exchanges.**ExchangeList**(*params=None*)
Retrieve a list of exchanges with detailed information about each. The response also contains links to other relevant data, such as their trade statuses.

ENDPOINT = 'openapi/ref/v1/exchanges/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params=None*)

Instantiate an ExchangeList request.

Parameters **params** (*dict (optional)*) – dict representing querystring parameters

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> r = rd.ExchangeList()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "__count": 181,
  "Data": [
    {
      "AllDay": false,
      "CountryCode": "US",
      "Currency": "USD",
      "ExchangeId": "NYSE_ARCA",
      "ExchangeSessions": [
        {
          "EndTime": "2019-03-04T12:00:00.000000Z",
          "StartTime": "2019-03-01T21:00:00.000000Z",
          "State": "Closed"
        },
        {
          "EndTime": "2019-03-04T14:30:00.000000Z",
          "StartTime": "2019-03-04T12:00:00.000000Z",
          "State": "PreTrading"
        },
        {
          "EndTime": "2019-03-04T21:00:00.000000Z",
          "StartTime": "2019-03-04T14:30:00.000000Z",
          "State": "AutomatedTrading"
        },
        {
          "EndTime": "2019-03-05T12:00:00.000000Z",
          "StartTime": "2019-03-04T21:00:00.000000Z",
          "State": "Closed"
        }
      ],
      "Mic": "ARCX",
      "Name": "New York Stock Exchange (ARCA)",
      "TimeZone": 3,
      "TimeZoneAbbreviation": "EST",
      "TimeZoneOffset": "-05:00:00"
    },
    {
      "AllDay": false,
```

(continues on next page)

(continued from previous page)


```
      "CountryCode": "SG",
      "Currency": "SGD",
      "ExchangeId": "SGX-DT",
      "ExchangeSessions": [
        {
          "EndTime": "2019-03-03T23:43:00.000000Z",
          "StartTime": "2019-03-01T11:05:00.000000Z",
          "State": "Closed"
        },
        {
          "EndTime": "2019-03-04T11:05:00.000000Z",
          "StartTime": "2019-03-03T23:43:00.000000Z",
          "State": "AutomatedTrading"
        },
        {
          "EndTime": "2019-03-04T23:43:00.000000Z",
          "StartTime": "2019-03-04T11:05:00.000000Z",
          "State": "Closed"
        },
        {
          "EndTime": "2019-03-05T11:05:00.000000Z",
          "StartTime": "2019-03-04T23:43:00.000000Z",
          "State": "AutomatedTrading"
        }
      ],
      "Mic": "XSES",
      "Name": "Singapore Exchange Derivatives Trading Ltd.",
      "TimeZone": 2,
      "TimeZoneAbbreviation": "SGT",
      "TimeZoneOffset": "08:00:00"
    },
    {
      "AllDay": false,
      "CountryCode": "CH",
      "Currency": "CHF",
      "ExchangeId": "SWX_ETF",
      "ExchangeSessions": [
        {
          "EndTime": "2019-03-04T05:00:00.000000Z",
          "StartTime": "2019-03-01T16:35:00.000000Z",
          "State": "Closed"
        },
        {
          "EndTime": "2019-03-04T08:00:00.000000Z",
          "StartTime": "2019-03-04T05:00:00.000000Z",
          "State": "PreTrading"
        },
        {
          "EndTime": "2019-03-04T16:30:00.000000Z",
          "StartTime": "2019-03-04T08:00:00.000000Z",
          "State": "AutomatedTrading"
        },
        {
          "EndTime": "2019-03-04T16:35:00.000000Z",
          "StartTime": "2019-03-04T16:30:00.000000Z",
          "State": "CallAuctionTrading"
        },
```


(continues on next page)

(continued from previous page)

```
        {
          "EndTime": "2019-03-05T05:00:00.000000Z",
          "StartTime": "2019-03-04T16:35:00.000000Z",
          "State": "Closed"
        }
      ],
      "Mic": "XSWX",
      "Name": "SIX Swiss Exchange (ETFs)",
      "TimeZone": 4,
      "TimeZoneAbbreviation": "CET",
      "TimeZoneOffset": "01:00:00"
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

3.5.6 saxo_openapi.endpoints.referencedata.instruments

ContractoptionSpaces

class saxo_openapi.endpoints.referencedata.instruments.**ContractoptionSpaces**(*OptionRootId*, *params=None*)

Get contractoption data.

ENDPOINT = 'openapi/ref/v1/instruments/contractoptionspaces/{OptionRootId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*OptionRootId*, *params=None*)
Instantiate a ContractoptionSpaces request.

Parameters

- **OptionRootId** (*string (required)*) – the OptionRootId
- **params** (*dict (optional)*) – dict representing querystring parameters

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> OptionRootId = 231
>>> params =
        {
          "ExpiryDates": "2019-05-01",
          "OptionSpaceSegment": "SpecificDates"
        }
```

```
>>> r = rd.instruments.ContractoptionSpaces(
...                           OptionRootId=OptionRootId,
...                           params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "AmountDecimals": 0,
  "AssetType": "StockOption",
  "CanParticipateInMultiLegOrder": false,
  "ContractSize": 100,
  "CurrencyCode": "EUR",
  "DefaultAmount": 1,
  "DefaultExpiry": "2019-04-18T00:00:00Z",
  "DefaultOption": {
    "PutCall": "Call",
    "StrikePrice": 27,
    "Uic": 11897720,
    "UnderlyingUic": 16350
  },
  "Description": "Royal Dutch Shell Plc A",
  "Exchange": {
    "CountryCode": "NL",
    "ExchangeId": "EUR_AMS2",
    "Name": "Euronext Equity & Index Derivatives - AMS"
  },
  "ExerciseStyle": "American",
  "Format": {
    "Decimals": 2,
    "OrderDecimals": 2,
    "StrikeDecimals": 3
  },
  "GroupId": 0,
  "IncrementSize": 1,
  "IsTradable": true,
  "LotSize": 1,
  "LotSizeType": "OddLotsNotAllowed",
  "OptionRootId": 231,
  "OptionSpace": [
    {
      "DisplayDaysToExpiry": 0,
      "DisplayExpiry": "2019-03-01",
      "Expiry": "2019-03-15",
      "LastTradeDate": "2019-03-15T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
          }
        ]
      }
    },
    {
```


(continues on next page)

(continued from previous page)


```
      "DisplayDaysToExpiry": 24,
      "DisplayExpiry": "2019-04-01",
      "Expiry": "2019-04-18",
      "LastTradeDate": "2019-04-18T15:30:00.000000Z",
      "SpecificOptions": [
        {
          "PutCall": "Call",
          "StrikePrice": 24,
          "Uic": 11897711,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Call",
          "StrikePrice": 25,
          "Uic": 11897712,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Call",
          "StrikePrice": 26.5,
          "Uic": 11897717,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Call",
          "StrikePrice": 26,
          "Uic": 11897719,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Call",
          "StrikePrice": 27,
          "Uic": 11897720,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Put",
          "StrikePrice": 25,
          "Uic": 11897721,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Put",
          "StrikePrice": 24,
          "Uic": 11897722,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Put",
          "StrikePrice": 26,
          "Uic": 11897723,
          "UnderlyingUic": 16350
        },
        {
          "PutCall": "Call",
          "StrikePrice": 28,
          "Uic": 11897724,
```


(continues on next page)

(continued from previous page)


```
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Call",
        "StrikePrice": 30,
        "Uic": 11897725,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Call",
        "StrikePrice": 27.5,
        "Uic": 11897728,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Call",
        "StrikePrice": 29,
        "Uic": 11897729,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Put",
        "StrikePrice": 26.5,
        "Uic": 11897730,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Put",
        "StrikePrice": 27,
        "Uic": 11897731,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Put",
        "StrikePrice": 27.5,
        "Uic": 11897732,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Put",
        "StrikePrice": 28,
        "Uic": 11897733,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Put",
        "StrikePrice": 30,
        "Uic": 11897734,
        "UnderlyingUic": 16350
      },
      {
        "PutCall": "Put",
        "StrikePrice": 29,
        "Uic": 11897735,
        "UnderlyingUic": 16350
      },
      {
```


(continues on next page)

(continued from previous page)


```
            "PutCall": "Put",
            "StrikePrice": 23,
            "Uic": 11900544,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Call",
            "StrikePrice": 23,
            "Uic": 11900558,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Call",
            "StrikePrice": 25.5,
            "Uic": 11903077,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Put",
            "StrikePrice": 25.5,
            "Uic": 11903078,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Put",
            "StrikePrice": 22,
            "Uic": 11949922,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Call",
            "StrikePrice": 22,
            "Uic": 11949924,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Call",
            "StrikePrice": 32,
            "Uic": 12003078,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Put",
            "StrikePrice": 32,
            "Uic": 12003081,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Call",
            "StrikePrice": 28.5,
            "Uic": 12007474,
            "UnderlyingUic": 16350
          },
          {
            "PutCall": "Put",
            "StrikePrice": 28.5,
            "Uic": 12007478,
```


(continues on next page)

(continued from previous page)


```
        "UnderlyingUic": 16350
      }
    ],
    "TickSizeScheme": {
      "DefaultTickSize": 0.05,
      "Elements": [
        {
          "HighPrice": 5,
          "TickSize": 0.01
        }
      ]
    }
  },
  {
    "DisplayDaysToExpiry": 54,
    "DisplayExpiry": "2019-05-01",
    "Expiry": "2019-05-17",
    "LastTradeDate": "2019-05-17T15:30:00.000000Z",
    "TickSizeScheme": {
      "DefaultTickSize": 0.05,
      "Elements": [
        {
          "HighPrice": 5,
          "TickSize": 0.01
        }
      ]
    }
  },
  {
    "DisplayDaysToExpiry": 85,
    "DisplayExpiry": "2019-06-01",
    "Expiry": "2019-06-21",
    "LastTradeDate": "2019-06-21T15:30:00.000000Z",
    "TickSizeScheme": {
      "DefaultTickSize": 0.05,
      "Elements": [
        {
          "HighPrice": 5,
          "TickSize": 0.01
        }
      ]
    }
  },
  {
    "DisplayDaysToExpiry": 177,
    "DisplayExpiry": "2019-09-01",
    "Expiry": "2019-09-20",
    "LastTradeDate": "2019-09-20T15:30:00.000000Z",
    "TickSizeScheme": {
      "DefaultTickSize": 0.05,
      "Elements": [
        {
          "HighPrice": 5,
          "TickSize": 0.01
        }
      ]
    }
```


(continues on next page)

(continued from previous page)


```
    },
    {
      "DisplayDaysToExpiry": 268,
      "DisplayExpiry": "2019-12-01",
      "Expiry": "2019-12-20",
      "LastTradeDate": "2019-12-20T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
          }
        ]
      }
    },
    {
      "DisplayDaysToExpiry": 451,
      "DisplayExpiry": "2020-06-01",
      "Expiry": "2020-06-19",
      "LastTradeDate": "2020-06-19T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
          }
        ]
      }
    },
    {
      "DisplayDaysToExpiry": 634,
      "DisplayExpiry": "2020-12-01",
      "Expiry": "2020-12-18",
      "LastTradeDate": "2020-12-18T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
          }
        ]
      }
    },
    {
      "DisplayDaysToExpiry": 999,
      "DisplayExpiry": "2021-12-01",
      "Expiry": "2021-12-17",
      "LastTradeDate": "2021-12-17T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
```


(continues on next page)

(continued from previous page)


```
          }
        ]
      }
    },
    {
      "DisplayDaysToExpiry": 1364,
      "DisplayExpiry": "2022-12-01",
      "Expiry": "2022-12-16",
      "LastTradeDate": "2022-12-16T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
          }
        ]
      }
    },
    {
      "DisplayDaysToExpiry": 1729,
      "DisplayExpiry": "2023-12-01",
      "Expiry": "2023-12-15",
      "LastTradeDate": "2023-12-15T16:30:00.000000Z",
      "TickSizeScheme": {
        "DefaultTickSize": 0.05,
        "Elements": [
          {
            "HighPrice": 5,
            "TickSize": 0.01
          }
        ]
      }
    }
  ],
  "OrderDistances": {
    "EntryDefaultDistance": 0,
    "EntryDefaultDistanceType": "Percentage",
    "StopLimitDefaultDistance": 5,
    "StopLimitDefaultDistanceType": "Pips",
    "StopLossDefaultDistance": 0.5,
    "StopLossDefaultDistanceType": "Percentage",
    "StopLossDefaultEnabled": false,
    "StopLossDefaultOrderType": "Stop",
    "TakeProfitDefaultDistance": 0.5,
    "TakeProfitDefaultDistanceType": "Percentage",
    "TakeProfitDefaultEnabled": false
  },
  "PriceToContractFactor": 100,
  "RelatedInstruments": [
    {
      "AssetType": "CfdOnStock",
      "Uic": 16350
    },
    {
      "AssetType": "Stock",
      "Uic": 16350
```


(continues on next page)

(continued from previous page)


```
    }
  ],
  "RelatedOptionRoots": [],
  "SettlementStyle": "PhysicalDelivery",
  "StandardAmounts": [
    1,
    5,
    10,
    25,
    50,
    100,
    500,
    1000
  ],
  "SupportedOrderTypes": [
    "Limit"
  ],
  "Symbol": "RDSA:xams",
  "TickSize": 0.01,
  "TradableOn": [
    "9300675"
  ],
  "UnderlyingAssetType": "Stock"
}
```


expected_status

response
response - get the response of the request.

status_code

FuturesSpaces

class saxo_openapi.endpoints.referencedata.instruments.**FuturesSpaces**(*ContinuousFuturesUic*)
Get futures spaces data.

ENDPOINT = 'openapi/ref/v1/instruments/futuresspaces/{ContinuousFuturesUic}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*ContinuousFuturesUic*)
Instantiate a ContractoptionSpaces request.

Parameters **ContinuousFuturesUic** (*string (required)*) – the ContinuousFuturesUic

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> ContinuousFuturesUic = '...'
>>> r = rd.instruments.FuturesSpaces(
...                     ContinuousFuturesUic=ContinuousFuturesUic)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{_v3_FuturesSpaces_resp}
```

expected_status

response
response - get the response of the request.

status_code

InstrumentDetails

class saxo_openapi.endpoints.referencedata.instruments.**InstrumentDetails**(*Uic*, *AssetType*, *params=None*)

Get detailed information for a specific instrument.

ENDPOINT = 'openapi/ref/v1/instruments/details/{Uic}/{AssetType}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*Uic*, *AssetType*, *params=None*)
Instantiate an InstrumentDetails request.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument
- **AssetType** (*string (required)*) – the AssetType specification
- **params** (*dict (optional)*) – dict representing querystring parameters

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> Uic = 22
>>> AssetType = 'FxSpot'
>>> params =
        {
          "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> r = rd.instruments.InstrumentDetails(Uic=Uic,
...                                      AssetType=AssetType,
...                                      params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "AmountDecimals": 6,
  "AssetType": "FxForwards",
  "CurrencyCode": "CAD",
```

(continues on next page)

(continued from previous page)


```
  "DefaultAmount": 100000,
  "DefaultSlippage": 0.01,
  "DefaultSlippageType": "Percentage",
  "Description": "British Pound/Canadian Dollar",
  "Exchange": {
    "CountryCode": "DK",
    "ExchangeId": "SBFX",
    "Name": "Inter Bank"
  },
  "Format": {
    "Decimals": 4,
    "Format": "AllowDecimalPips",
    "OrderDecimals": 4
  },
  "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
  "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
  "FxSpotDate": "2019-03-06T00:00:00.000000Z",
  "GroupId": 0,
  "IncrementSize": 5000,
  "IsTradable": true,
  "MinimumTradeSize": 1000,
  "OrderDistances": {
    "EntryDefaultDistance": 0.5,
    "EntryDefaultDistanceType": "Percentage",
    "StopLimitDefaultDistance": 5,
    "StopLimitDefaultDistanceType": "Pips",
    "StopLossDefaultDistance": 50,
    "StopLossDefaultDistanceType": "Pips",
    "StopLossDefaultEnabled": false,
    "StopLossDefaultOrderType": "Stop",
    "TakeProfitDefaultDistance": 50,
    "TakeProfitDefaultDistanceType": "Pips",
    "TakeProfitDefaultEnabled": false
  },
  "StandardAmounts": [
    10000,
    50000,
    100000,
    250000,
    500000,
    1000000,
    2000000,
    5000000,
    10000000,
    20000000
  ],
  "SupportedOrderTypes": [
    "Market",
    "Limit",
    "Stop",
    "TrailingStop",
    "StopLimit"
  ],
  "Symbol": "GBPCAD",
  "TickSize": 0.0001,
  "TickSizeLimitOrder": 0.0001,
  "TickSizeStopOrder": 5e-05,
```


(continues on next page)

(continued from previous page)

```
  "TradableAs": [
    "FxSpot",
    "FxForwards",
    "FxVanillaOption",
    "FxKnockInOption",
    "FxKnockOutOption"
  ],
  "TradableOn": [
    "9226397"
  ],
  "TradingSignals": "NotAllowed",
  "Uic": 23
}
```

expected_status

response
response - get the response of the request.

status_code

Instruments

class saxo_openapi.endpoints.referencedata.instruments.**Instruments**(*params*)
Get a list of summary information for all instruments and options on the Saxo Trading platform restricted by the access rights of the user.

ENDPOINT = 'openapi/ref/v1/instruments/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate an Instruments request.

Parameters **params** (*dict (required)*) – dict reppresenting the querystring parameters

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
          "Uics": "12,22,23"
        }
```

```
>>> r = rd.instruments.Instruments(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Data": [
    {
```

(continues on next page)

(continued from previous page)


```
      "AssetType": "FxSpot",
      "CurrencyCode": "AUD",
      "Description": "Euro/Australian Dollar",
      "ExchangeId": "SBFX",
      "GroupId": 28784,
      "Identifier": 12,
      "SummaryType": "Instrument",
      "Symbol": "EURAUD",
      "TradableAs": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ]
    },
    {
      "AssetType": "FxSpot",
      "CurrencyCode": "AUD",
      "Description": "British Pound/Australian Dollar",
      "ExchangeId": "SBFX",
      "GroupId": 28867,
      "Identifier": 22,
      "SummaryType": "Instrument",
      "Symbol": "GBPAUD",
      "TradableAs": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ]
    },
    {
      "AssetType": "FxSpot",
      "CurrencyCode": "CAD",
      "Description": "British Pound/Canadian Dollar",
      "ExchangeId": "SBFX",
      "GroupId": 28871,
      "Identifier": 23,
      "SummaryType": "Instrument",
      "Symbol": "GBPCAD",
      "TradableAs": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ]
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

InstrumentsDetails

class saxo_openapi.endpoints.referencedata.instruments.**InstrumentsDetails**(*params*)

Get detailed information on a list of instruments.

ENDPOINT = 'openapi/ref/v1/instruments/details'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)

Instantiate an InstrumentsDetails request.

Parameters **params** (*dict (required)*) – dict reppresenting the querystring parameters

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
          "Uics": "23"
        }
```

```
>>> r = rd.instruments.InstrumentsDetails(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "__count": 6,
  "Data": [
    {
      "AmountDecimals": 6,
      "AssetType": "FxSwap",
      "CurrencyCode": "CAD",
      "DefaultAmount": 100000,
      "DefaultSlippage": 0.01,
      "DefaultSlippageType": "Percentage",
      "Description": "British Pound/Canadian Dollar",
      "Exchange": {
        "CountryCode": "DK",
        "ExchangeId": "SBFX",
        "Name": "Inter Bank"
      },
      "Format": {
        "Decimals": 4,
        "Format": "AllowTwoDecimalPips",
        "OrderDecimals": 4
      },
      "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
      "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
      "FxSpotDate": "2019-03-06T00:00:00.000000Z",
```

(continues on next page)

(continued from previous page)


```
      "GroupId": 0,
      "IncrementSize": 5000,
      "IsTradable": true,
      "MinimumTradeSize": 1000,
      "OrderDistances": {
        "EntryDefaultDistance": 0.5,
        "EntryDefaultDistanceType": "Percentage",
        "StopLimitDefaultDistance": 5,
        "StopLimitDefaultDistanceType": "Pips",
        "StopLossDefaultDistance": 50,
        "StopLossDefaultDistanceType": "Pips",
        "StopLossDefaultEnabled": false,
        "StopLossDefaultOrderType": "Stop",
        "TakeProfitDefaultDistance": 50,
        "TakeProfitDefaultDistanceType": "Pips",
        "TakeProfitDefaultEnabled": false
      },
      "StandardAmounts": [
        10000,
        50000,
        100000,
        250000,
        500000,
        1000000,
        2000000,
        5000000,
        10000000,
        20000000
      ],
      "SupportedOrderTypes": [
        "Market",
        "Limit"
      ],
      "Symbol": "GBPCAD",
      "TickSize": 0.0001,
      "TickSizeLimitOrder": 0.0001,
      "TickSizeStopOrder": 5e-05,
      "TradableAs": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ],
      "TradableOn": [],
      "TradingSignals": "NotAllowed",
      "Uic": 23
    },
    {
      "AmountDecimals": 6,
      "AssetType": "FxKnockOutOption",
      "CurrencyCode": "CAD",
      "DefaultAmount": 100,
      "Description": "British Pound/Canadian Dollar",
      "Exchange": {
        "CountryCode": "DK",
        "ExchangeId": "SBFX",
```


(continues on next page)

(continued from previous page)


```
      "Name": "Inter Bank"
    },
    "Format": {
      "Decimals": 4,
      "OrderDecimals": 4,
      "StrikeDecimals": 4
    },
    "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
    "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
    "GroupId": 0,
    "IncrementSize": 100,
    "IsTradable": true,
    "MinimumTradeSize": 1000,
    "OptionsChainSubscriptionAllowed": true,
    "OrderDistances": {
      "EntryDefaultDistance": 0.25,
      "EntryDefaultDistanceType": "Percentage",
      "StopLimitDefaultDistance": 5,
      "StopLimitDefaultDistanceType": "Pips",
      "StopLossDefaultDistance": 0.5,
      "StopLossDefaultDistanceType": "Percentage",
      "StopLossDefaultEnabled": false,
      "StopLossDefaultOrderType": "Stop",
      "TakeProfitDefaultDistance": 0.5,
      "TakeProfitDefaultDistanceType": "Percentage",
      "TakeProfitDefaultEnabled": false
    },
    "StandardAmounts": [
      10000,
      30000,
      50000,
      80000,
      100000
    ],
    "SupportedOrderTypes": [
      "Market",
      "Limit",
      "Stop",
      "TrailingStop",
      "StopLimit"
    ],
    "Symbol": "GBPCAD",
    "TickSize": 0.0001,
    "TickSizeLimitOrder": 0.0001,
    "TickSizeStopOrder": 5e-05,
    "TradableAs": [
      "FxSpot",
      "FxForwards",
      "FxVanillaOption",
      "FxKnockInOption",
      "FxKnockOutOption"
    ],
    "TradableOn": [
      "9226397"
    ],
    "TradingSignals": "NotAllowed",
    "Uic": 23
```


(continues on next page)

(continued from previous page)


```
    },
    {
      "AmountDecimals": 6,
      "AssetType": "FxKnockInOption",
      "CurrencyCode": "CAD",
      "DefaultAmount": 100,
      "Description": "British Pound/Canadian Dollar",
      "Exchange": {
        "CountryCode": "DK",
        "ExchangeId": "SBFX",
        "Name": "Inter Bank"
      },
      "Format": {
        "Decimals": 4,
        "OrderDecimals": 4,
        "StrikeDecimals": 4
      },
      "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
      "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
      "GroupId": 0,
      "IncrementSize": 100,
      "IsTradable": true,
      "MinimumTradeSize": 1000,
      "OptionsChainSubscriptionAllowed": true,
      "OrderDistances": {
        "EntryDefaultDistance": 0.25,
        "EntryDefaultDistanceType": "Percentage",
        "StopLimitDefaultDistance": 5,
        "StopLimitDefaultDistanceType": "Pips",
        "StopLossDefaultDistance": 0.5,
        "StopLossDefaultDistanceType": "Percentage",
        "StopLossDefaultEnabled": false,
        "StopLossDefaultOrderType": "Stop",
        "TakeProfitDefaultDistance": 0.5,
        "TakeProfitDefaultDistanceType": "Percentage",
        "TakeProfitDefaultEnabled": false
      },
      "StandardAmounts": [
        10000,
        30000,
        50000,
        80000,
        100000
      ],
      "SupportedOrderTypes": [
        "Market",
        "Limit",
        "Stop",
        "TrailingStop",
        "StopLimit"
      ],
      "Symbol": "GBPCAD",
      "TickSize": 0.0001,
      "TickSizeLimitOrder": 0.0001,
      "TickSizeStopOrder": 5e-05,
      "TradableAs": [
        "FxSpot",
```


(continues on next page)

(continued from previous page)


```
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ],
      "TradableOn": [
        "9226397"
      ],
      "TradingSignals": "NotAllowed",
      "Uic": 23
    },
    {
      "AmountDecimals": 6,
      "AssetType": "FxVanillaOption",
      "CurrencyCode": "CAD",
      "DefaultAmount": 100000,
      "Description": "British Pound/Canadian Dollar",
      "Exchange": {
        "CountryCode": "DK",
        "ExchangeId": "SBFX",
        "Name": "Inter Bank"
      },
      "Format": {
        "Decimals": 4,
        "Format": "AllowDecimalPips",
        "StrikeDecimals": 4
      },
      "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
      "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
      "GroupId": 0,
      "IncrementSize": 10000,
      "IsTradable": true,
      "MinimumTradeSize": 1000,
      "OptionsChainSubscriptionAllowed": true,
      "OrderDistances": {
        "EntryDefaultDistance": 0.5,
        "EntryDefaultDistanceType": "Percentage",
        "StopLimitDefaultDistance": 5,
        "StopLimitDefaultDistanceType": "Pips",
        "StopLossDefaultDistance": 50,
        "StopLossDefaultDistanceType": "Pips",
        "StopLossDefaultEnabled": false,
        "StopLossDefaultOrderType": "Stop",
        "TakeProfitDefaultDistance": 50,
        "TakeProfitDefaultDistanceType": "Pips",
        "TakeProfitDefaultEnabled": false
      },
      "StandardAmounts": [
        100000,
        250000,
        500000,
        1000000,
        2000000,
        3000000,
        4000000,
        5000000,
        10000000,
```


(continues on next page)

(continued from previous page)


```
        20000000
      ],
      "SupportedOrderTypes": [
        "Market",
        "Limit",
        "Stop",
        "TrailingStop",
        "StopLimit"
      ],
      "Symbol": "GBPCAD",
      "TickSize": 0.0005,
      "TickSizeLimitOrder": 0.0001,
      "TickSizeStopOrder": 5e-05,
      "TradableAs": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ],
      "TradableOn": [
        "9226397"
      ],
      "TradingSignals": "NotAllowed",
      "Uic": 23
    },
    {
      "AmountDecimals": 6,
      "AssetType": "FxSpot",
      "CurrencyCode": "CAD",
      "DefaultAmount": 100000,
      "DefaultSlippage": 0.01,
      "DefaultSlippageType": "Percentage",
      "Description": "British Pound/Canadian Dollar",
      "Exchange": {
        "CountryCode": "DK",
        "ExchangeId": "SBFX",
        "Name": "Inter Bank"
      },
      "Format": {
        "Decimals": 4,
        "Format": "AllowDecimalPips",
        "OrderDecimals": 4
      },
      "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
      "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
      "FxSpotDate": "2019-03-06T00:00:00.000000Z",
      "GroupId": 0,
      "IncrementSize": 5000,
      "IsTradable": true,
      "MinimumTradeSize": 1000,
      "OrderDistances": {
        "EntryDefaultDistance": 0.5,
        "EntryDefaultDistanceType": "Percentage",
        "StopLimitDefaultDistance": 5,
        "StopLimitDefaultDistanceType": "Pips",
        "StopLossDefaultDistance": 50,
```


(continues on next page)

(continued from previous page)


```
        "StopLossDefaultDistanceType": "Pips",
        "StopLossDefaultEnabled": false,
        "StopLossDefaultOrderType": "Stop",
        "TakeProfitDefaultDistance": 50,
        "TakeProfitDefaultDistanceType": "Pips",
        "TakeProfitDefaultEnabled": false
      },
      "StandardAmounts": [
        10000,
        50000,
        100000,
        250000,
        500000,
        1000000,
        2000000,
        5000000,
        10000000,
        20000000
      ],
      "SupportedOrderTypes": [
        "Market",
        "Limit",
        "Stop",
        "TrailingStop",
        "StopLimit"
      ],
      "Symbol": "GBPCAD",
      "TickSize": 5e-05,
      "TickSizeLimitOrder": 0.0001,
      "TickSizeStopOrder": 5e-05,
      "TradableAs": [
        "FxSpot",
        "FxForwards",
        "FxVanillaOption",
        "FxKnockInOption",
        "FxKnockOutOption"
      ],
      "TradableOn": [
        "9226397"
      ],
      "TradingSignals": "Allowed",
      "Uic": 23
    },
    {
      "AmountDecimals": 6,
      "AssetType": "FxForwards",
      "CurrencyCode": "CAD",
      "DefaultAmount": 100000,
      "DefaultSlippage": 0.01,
      "DefaultSlippageType": "Percentage",
      "Description": "British Pound/Canadian Dollar",
      "Exchange": {
        "CountryCode": "DK",
        "ExchangeId": "SBFX",
        "Name": "Inter Bank"
      },
      "Format": {
```


(continues on next page)

(continued from previous page)


```
      "Decimals": 4,
      "Format": "AllowDecimalPips",
      "OrderDecimals": 4
    },
    "FxForwardMaxForwardDate": "2020-03-10T00:00:00.000000Z",
    "FxForwardMinForwardDate": "2019-03-06T00:00:00.000000Z",
    "FxSpotDate": "2019-03-06T00:00:00.000000Z",
    "GroupId": 0,
    "IncrementSize": 5000,
    "IsTradable": true,
    "MinimumTradeSize": 1000,
    "OrderDistances": {
      "EntryDefaultDistance": 0.5,
      "EntryDefaultDistanceType": "Percentage",
      "StopLimitDefaultDistance": 5,
      "StopLimitDefaultDistanceType": "Pips",
      "StopLossDefaultDistance": 50,
      "StopLossDefaultDistanceType": "Pips",
      "StopLossDefaultEnabled": false,
      "StopLossDefaultOrderType": "Stop",
      "TakeProfitDefaultDistance": 50,
      "TakeProfitDefaultDistanceType": "Pips",
      "TakeProfitDefaultEnabled": false
    },
    "StandardAmounts": [
      10000,
      50000,
      100000,
      250000,
      500000,
      1000000,
      2000000,
      5000000,
      10000000,
      20000000
    ],
    "SupportedOrderTypes": [
      "Market",
      "Limit",
      "Stop",
      "TrailingStop",
      "StopLimit"
    ],
    "Symbol": "GBPCAD",
    "TickSize": 0.0001,
    "TickSizeLimitOrder": 0.0001,
    "TickSizeStopOrder": 5e-05,
    "TradableAs": [
      "FxSpot",
      "FxForwards",
      "FxVanillaOption",
      "FxKnockInOption",
      "FxKnockOutOption"
    ],
    "TradableOn": [
      "9226397"
    ],
```


(continues on next page)

(continued from previous page)

```
      "TradingSignals": "NotAllowed",
      "Uic": 23
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

TradingSchedule

class saxo_openapi.endpoints.referencedata.instruments.**TradingSchedule**(*Uic*, *AssetType*)

Get TradingSchedule data.

ENDPOINT = 'openapi/ref/v1/instruments/tradingschedule/{Uic}/{AssetType}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*Uic*, *AssetType*)
Instantiate a TradingSchedule request.

Parameters

- **Uic** (*string (required)*) – the Uic of the instrument
- **AssetType** (*string (required)*) – the AssetType of the instrument
- **one Uic multiple assettypes can be available trading** (*For*) –
- **different times.** (*on*) –

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> Uic = 21
>>> AssetType = "FxSpot"
>>> r = rd.instruments.ContractoptionSpaces(
...                           Uic=Uic,
...                           AssetType=AssetType)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{_v3_TradingSchedule_resp}
```

expected_status

response
response - get the response of the request.

status_code

3.5.7 saxo_openapi.endpoints.referencedata.languages

Languages

class saxo_openapi.endpoints.referencedata.languages.**Languages**

Get a list containing all the languages supported by Saxo Bank.

ENDPOINT = 'openapi/ref/v1/languages/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()

Instantiate a Languages request.

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> r = rd.languages.Languages()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

```
{
  "Data": [
    {
      "LanguageCode": "ar",
      "LanguageName": "Arabic",
      "NativeName": "Arabic - \u0639\u0631\u0628\u064a"
    },
    {
      "LanguageCode": "bg",
      "LanguageName": "Bulgarian",
      "NativeName": "Bulgarian -␣
↪\u0411\u044a\u043b\u0433\u0430\u0440\u0441\u043a\u0438"
    },
    {
      "LanguageCode": "cs",
      "LanguageName": "Czech",
      "NativeName": "Cesky"
    },
    {
      "LanguageCode": "da",
      "LanguageName": "Danish",
      "NativeName": "Dansk"
    },
    {
      "LanguageCode": "de",
      "LanguageName": "German",
      "NativeName": "Deutsch"
    },
    {
      "LanguageCode": "el",
      "LanguageName": "Greek",
```

(continues on next page)

(continued from previous page)


```
      "NativeName": "Greek - \u0395\u03bb\u03bb\u03b7\u03bd\u03b9\u03ba\u03ac"
    },
    {
      "LanguageCode": "en",
      "LanguageName": "English",
      "NativeName": "English"
    },
    {
      "LanguageCode": "es",
      "LanguageName": "Spanish",
      "NativeName": "Espa\u00f1ol"
    },
    {
      "LanguageCode": "fi",
      "LanguageName": "Finnish",
      "NativeName": "Suomi"
    },
    {
      "LanguageCode": "fr",
      "LanguageName": "French",
      "NativeName": "Fran\u00e7ais"
    },
    {
      "LanguageCode": "...",
      "LanguageName": "...",
      "NativeName": "..."
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.5.8 saxo_openapi.endpoints.referencedata.standarddates

FXOptionExpiryDates

class saxo_openapi.endpoints.referencedata.standarddates.**FXOptionExpiryDates**(*Uic*)
Get a list of FX option expiry dates for an UIC.

ENDPOINT = 'openapi/ref/v1/standarddates/fxoptionexpiry/{Uic}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*Uic*)
Instantiate a FXOptionExpiryDates request.

Parameters **Uic** (*int (required)*) – the Uic code of the instrument

```
>>> import json
>>> import saxo_openapi
```

(continues on next page)

(continued from previous page)

```
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> Uic = 22
>>> r = rd.FXOptionExpiryDates(Uic=Uic)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=2))
```

Output:


```
{
  "Data": [
    {
      "Date": "2019-03-07",
      "Unit": "Days",
      "Value": 1
    },
    {
      "Date": "2019-03-13",
      "Unit": "Weeks",
      "Value": 1
    },
    {
      "Date": "2019-03-20",
      "Unit": "Weeks",
      "Value": 2
    },
    {
      "Date": "2019-03-27",
      "Unit": "Weeks",
      "Value": 3
    },
    {
      "Date": "2019-04-04",
      "Unit": "Months",
      "Value": 1
    },
    {
      "Date": "2019-05-06",
      "Unit": "Months",
      "Value": 2
    },
    {
      "Date": "2019-06-06",
      "Unit": "Months",
      "Value": 3
    },
    {
      "Date": "2019-09-05",
      "Unit": "Months",
      "Value": 6
    },
    {
      "Date": "2019-12-05",
      "Unit": "Months",
      "Value": 9
    },
    {
```


(continues on next page)

(continued from previous page)

```
      "Date": "2020-03-05",
      "Unit": "Years",
      "Value": 1
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

ForwardTenorDates

class saxo_openapi.endpoints.referencedata.standarddates.**ForwardTenorDates**(*Uic*, *params=None*)

Get a list of forward tenor dates for an UIC.

ENDPOINT = 'openapi/ref/v1/standarddates/forwardtenor/{Uic}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*Uic*, *params=None*)
Instantiate a ForwardTenorDates request.

Parameters

- **Uic** (*int (required)*) – the Uic code of the instrument
- **params** (*dict (required)*) – dict with parameters representing the querystring

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA=="
        }
```

```
>>> Uic = 22
>>> r = rd.ForwardTenorDates(Uic=Uic, params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Data": [
    {
      "Date": "2019-03-06",
      "Unit": "Days",
      "Value": 0
    },
```

(continues on next page)

(continued from previous page)


```
    {
      "Date": "2019-03-13",
      "Unit": "Weeks",
      "Value": 1
    },
    {
      "Date": "2019-03-20",
      "Unit": "Weeks",
      "Value": 2
    },
    {
      "Date": "2019-03-27",
      "Unit": "Weeks",
      "Value": 3
    },
    {
      "Date": "2019-04-08",
      "Unit": "Months",
      "Value": 1
    },
    {
      "Date": "2019-05-07",
      "Unit": "Months",
      "Value": 2
    },
    {
      "Date": "2019-06-06",
      "Unit": "Months",
      "Value": 3
    },
    {
      "Date": "2019-09-06",
      "Unit": "Months",
      "Value": 6
    },
    {
      "Date": "2019-12-06",
      "Unit": "Months",
      "Value": 9
    },
    {
      "Date": "2020-03-06",
      "Unit": "Years",
      "Value": 1
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.5.9 saxo_openapi.endpoints.referencedata.timezones

TimeZones

class saxo_openapi.endpoints.referencedata.timezones.**TimeZones**

Get a list all the time zones supported by Saxo Bank.

ENDPOINT = 'openapi/ref/v1/timezones/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()

Instantiate a TimeZones request.

```
>>> import json
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.referencedata as rd
>>> client = saxo_openapi.API(access_token=...)
>>> r = rd.timezones.TimeZones()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```


```
{
  "Data": [
    {
      "DisplayName": "GMT a.k.a. UTC",
      "TimeZoneId": 0,
      "ZoneName": "Etc/UTC"
    },
    {
      "DisplayName": "British Time",
      "TimeZoneId": 1,
      "ZoneName": "Europe/London"
    },
    {
      "DisplayName": "Singapore Time",
      "TimeZoneId": 2,
      "ZoneName": "Asia/Singapore"
    },
    {
      "DisplayName": "US Eastern Time",
      "TimeZoneId": 3,
      "ZoneName": "America/New_York"
    },
    {
      "DisplayName": "Central European Time",
      "TimeZoneId": 4,
      "ZoneName": "Europe/Paris"
    },
    {
      "DisplayName": "US Central Time",
      "TimeZoneId": 5,
      "ZoneName": "America/Chicago"
    },
    {
      "DisplayName": "US Pacific Time",
      "TimeZoneId": 6,
      "ZoneName": "America/Los_Angeles"
    },
```


(continues on next page)

(continued from previous page)


```
    {
      "DisplayName": "Hong Kong Time",
      "TimeZoneId": 7,
      "ZoneName": "Asia/Hong_Kong"
    },
    {
      "DisplayName": "Sydney Time",
      "TimeZoneId": 8,
      "ZoneName": "Australia/Sydney"
    },
    {
      "DisplayName": "New Zealand Time",
      "TimeZoneId": 9,
      "ZoneName": "Pacific/Auckland"
    },
    {
      "DisplayName": "GMT +9 No Daylight S.",
      "TimeZoneId": 10,
      "ZoneName": "Etc/GMT-9"
    },
    {
      "DisplayName": "GMT +7 No Daylight S.",
      "TimeZoneId": 11,
      "ZoneName": "Etc/GMT-7"
    },
    {
      "DisplayName": "Russia Zone 2",
      "TimeZoneId": 12,
      "ZoneName": "Europe/Moscow"
    },
    {
      "DisplayName": "GMT +8 No Daylight S.",
      "TimeZoneId": 13,
      "ZoneName": "Etc/GMT-8"
    },
    {
      "DisplayName": "Eastern European Time",
      "TimeZoneId": 14,
      "ZoneName": "Europe/Helsinki"
    },
    {
      "DisplayName": "Hawaii Time",
      "TimeZoneId": 15,
      "ZoneName": "Pacific/Honolulu"
    },
    {
      "DisplayName": "South African Time",
      "TimeZoneId": 16,
      "ZoneName": "Africa/Johannesburg"
    },
    {
      "DisplayName": "GMT +10 No Daylight S.",
      "TimeZoneId": 17,
      "ZoneName": "Etc/GMT-10"
    },
    {
      "DisplayName": "GMT+3",
```


(continues on next page)

(continued from previous page)


```
      "TimeZoneId": 18,
      "ZoneName": "Etc/GMT-3"
    },
    {
      "DisplayName": "GMT+4",
      "TimeZoneId": 19,
      "ZoneName": "Etc/GMT-4"
    },
    {
      "DisplayName": "Brazil Sao Paulo",
      "TimeZoneId": 20,
      "ZoneName": "America/Sao_Paulo"
    },
    {
      "DisplayName": "Africa/Cairo",
      "TimeZoneId": 33,
      "ZoneName": "Africa/Cairo"
    },
    {
      "DisplayName": "America/Caracas",
      "TimeZoneId": 104,
      "ZoneName": "America/Caracas"
    },
    {
      "DisplayName": "America/Halifax",
      "TimeZoneId": 130,
      "ZoneName": "America/Halifax"
    },
    {
      "DisplayName": "America/La_Paz",
      "TimeZoneId": 147,
      "ZoneName": "America/La_Paz"
    },
    {
      "DisplayName": "Asia/Kolkata",
      "TimeZoneId": 256,
      "ZoneName": "Asia/Kolkata"
    },
    {
      "DisplayName": "Atlantic/Azores",
      "TimeZoneId": 298,
      "ZoneName": "Atlantic/Azores"
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

3.6 saxo_openapi.endpoints.rootservices

3.6.1 saxo_openapi.endpoints.rootservices.diagnostics

Delete

class saxo_openapi.endpoints.rootservices.diagnostics.**Delete**

Send a DELETE request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/delete/'

EXPECTED_STATUS = 200

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__()

Instantiate a Delete request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Delete()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

Echo

class saxo_openapi.endpoints.rootservices.diagnostics.**Echo**

Send a any request and get a 200 OK response with verb, url, headers and body in the response body.

ENDPOINT = 'openapi/root/v1/diagnostics/echo/'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_DATA = None

__init__()

Instantiate an Echo request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Echo()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Get

class saxo_openapi.endpoints.rootservices.diagnostics.**Get**
Send a GET request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/get/'

EXPECTED_STATUS = 200

METHOD = 'GET'

RESPONSE_DATA = None

__init__()
Instantiate a Get request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Get()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Head

class saxo_openapi.endpoints.rootservices.diagnostics.**Head**
Send a HEAD request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/head/'

EXPECTED_STATUS = 200

METHOD = 'HEAD'

RESPONSE_DATA = None

__init__()
Instantiate a Head request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Head()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Options

class saxo_openapi.endpoints.rootservices.diagnostics.**Options**
Send a OPTIONS request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/options/'

EXPECTED_STATUS = 200

METHOD = 'OPTIONS'

RESPONSE_DATA = None

__init__()
Instantiate a Options request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Options()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Patch

class saxo_openapi.endpoints.rootservices.diagnostics.**Patch**
Send a PATCH request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/patch/'

EXPECTED_STATUS = 200

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__()
Instantiate a Patch request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Patch()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Post

class saxo_openapi.endpoints.rootservices.diagnostics.**Post**
Send a POST request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/post/'

EXPECTED_STATUS = 200

METHOD = 'POST'

RESPONSE_DATA = None

__init__()
Instantiate a Post request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Post()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

Put

class saxo_openapi.endpoints.rootservices.diagnostics.**Put**
Send a PUT request and get a 200 OK response back.

ENDPOINT = 'openapi/root/v1/diagnostics/put/'

EXPECTED_STATUS = 200

METHOD = 'PUT'

RESPONSE_DATA = None

__init__()
Instantiate a Put request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.diagnostics.Put()
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.6.2 saxo_openapi.endpoints.rootservices.features

Availability

class saxo_openapi.endpoints.rootservices.features.**Availability**
Get the availability of all features.

ENDPOINT = 'openapi/root/v1/features/availability/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate an Availability request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices.features as rsft
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rsft.Availability()
>>> rv = client.request(r)
>>> print(rv)
```

Output:

```
[
  {
    "Available": true,
    "Feature": "News"
  },
  {
    "Available": true,
```

(continues on next page)

(continued from previous page)

```
    "Feature": "GainersLosers"
  },
  {
    "Available": true,
    "Feature": "Calendar"
  },
  {
    "Available": true,
    "Feature": "Chart"
  }
]
```

expected_status

response
response - get the response of the request.

status_code

CreateAvailabilitySubscription

class saxo_openapi.endpoints.rootservices.features.**CreateAvailabilitySubscription**(*data*)
Create a feature availability subscription.

ENDPOINT = 'openapi/root/v1/features/availability/subscriptions'

EXPECTED_STATUS = 201

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

METHOD = 'POST'

__init__(*data*)
Instantiate an CreateAvailabilitySubscription request.

Parameters **data** (*JSON (required)*) – json body to send

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices.features as rsft
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "RefreshRate": 5000,
          "ReferenceId": "Features",
          "ContextId": "20190209072629616"
        }
```

```
>>> r = rsft.CreateAvailabilitySubscription(data=data)
>>> rv = client.request(r)
>>> print(rv)
```

Output:

```
{
  "ContextId": "20190209072629616",
  "InactivityTimeout": 30,
  "ReferenceId": "Features",
```

(continues on next page)

(continued from previous page)

```
  "RefreshRate": 0,
  "Snapshot": [
    {
      "Available": true,
      "Feature": "News"
    },
    {
      "Available": true,
      "Feature": "GainersLosers"
    },
    {
      "Available": true,
      "Feature": "Calendar"
    },
    {
      "Available": true,
      "Feature": "Chart"
    }
  ],
  "State": "Active"
}
```

expected_status

response
response - get the response of the request.

status_code

RemoveAvailabilitySubscription

class saxo_openapi.endpoints.rootservices.features.**RemoveAvailabilitySubscription**(*ContextId*, *ReferenceId*)

Removes the subscription identified by the specified reference id (and streaming context id).

ENDPOINT = 'openapi/root/v1/features/availability/subscriptions/{ContextId}/{Reference

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a RemoveAvailabilitySubscription request.

Parameters

- **ContextId** (*string (required)*) – the context-id
- **ReferenceId** (*string (required)*) – the reference-id

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices.features as rsft
>>> import json
>>> client = saxo_openapi.API(access_token=...)
```

(continues on next page)

(continued from previous page)

```
>>> ContextId = '...'
>>> ReferenceId = '...'
>>> r = rsft.RemoveAvailabilitySubscription(ContextId=ContextId,
...                                         ReferenceId=ReferenceId)
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.6.3 saxo_openapi.endpoints.rootservices.sessions

ChangeSessionCapabilities

class saxo_openapi.endpoints.rootservices.sessions.**ChangeSessionCapabilities**(*data*)
Change sessions capabilities.

ENDPOINT = 'openapi/root/v1/sessions/capabilities/'

EXPECTED_STATUS = 202

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*data*)
Instantiate a ChangeSessionCapabilities request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "TradeLevel": "OrdersOnly"
        }
```

```
>>> r = rs.sessions.ChangeSessionCapabilities(data=data)
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned

expected_status

response
response - get the response of the request.

status_code

CreateSessionCapabilitiesSubscription

class saxo_openapi.endpoints.rootservices.sessions.**CreateSessionCapabilitiesSubscription**(*da*

Set up a new session capabilities subscription. The data stream will deliver updates from this point.

ENDPOINT = 'openapi/root/v1/sessions/events/subscriptions/'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)

Instantiate a ChangeSessionCapabilitiesSubscription request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "ContextId": "20190314053019906",
          "ReferenceId": "S56886",
          "RefreshRate": 1000,
          "Tag": "PAGE1"
        }
```

```
>>> r = rs.sessions.ChangeSessionCapabilitiesSubscription(data=data)
>>> rv = client.request(r)
>>> print(rv)
```

Output:

```
{
  "ContextId": "20190314053019906",
  "Format": "application/json",
  "InactivityTimeout": 120,
  "ReferenceId": "S56886",
  "RefreshRate": 1000,
  "Snapshot": {
    "DataLevel": "Standard",
    "TradeLevel": "OrdersOnly"
  },
  "State": "Active"
}
```

expected_status

response

response - get the response of the request.

status_code

GetSessionCapabilities

class saxo_openapi.endpoints.rootservices.sessions.**GetSessionCapabilities**

Get the sessions capabilities.

ENDPOINT = 'openapi/root/v1/sessions/capabilities/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a GetSessionCapabilities request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.sessions.GetSessionCapabilities()
>>> rv = client.request(r)
>>> print(rv)
```

Output:

```
{
  "DataLevel": "Standard",
  "TradeLevel": "OrdersOnly"
}
```

expected_status

response
response - get the response of the request.

status_code

RemoveSessionCapabilitiesSubscription

class saxo_openapi.endpoints.rootservices.sessions.**RemoveSessionCapabilitiesSubscription**(*Co Re er en ce*

Removes the subscription identified by the specified reference id. (and streaming context id).

ENDPOINT = 'openapi/root/v1/sessions/events/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a RemoveSessionCapabilitiesSubscription request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.sessions.RemoveSessionCapabilitiesSubscripion(
...       ContextId=ContextId,
...       ReferenceId=ReferenceId)
>>> rv = client.request(r)
>>> assert rv.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.6.4 saxo_openapi.endpoints.rootservices.subscriptions

RemoveMultipleActiveSubscriptions

class saxo_openapi.endpoints.rootservices.subscriptions.**RemoveMultipleActiveSubscriptions**(

Removes multiple subscriptions for the current session, and frees all resources on the server.

ENDPOINT = 'openapi/root/subscriptions/{ContextId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params*)
Instantiate a User request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ''
>>> params =
        {
          "Tag": "TAB9"
        }
```

```
>>> r = rs.subscriptions.RemoveMultipleActiveSubscriptions(
...           ContextId, params=params)
>>> rv = client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.6.5 saxo_openapi.endpoints.rootservices.user

User

class saxo_openapi.endpoints.rootservices.user.**User**
Get information of current user.

ENDPOINT = 'openapi/root/v1/user/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a User request.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.rootservices as rs
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = rs.user.User()
>>> rv = client.request(r)
>>> print(rv)
```

Output:

```
{
  "AccessRights": {
    "CanManageCashTransfers": true,
    "CanTakePriceSession": true,
    "CanTakeTradeSession": true,
    "CanTrade": true,
    "CanViewAnyClient": true
  },
  "ClientId": 467989277,
  "EmployeeId": "472298225",
  "Roles": [
    "OAPI.Roles.RetailClient"
  ],
  "UserId": 983293960
}
```

expected_status

response
response - get the response of the request.

status_code

3.7 saxo_openapi.endpoints.trading

3.7.1 saxo_openapi.endpoints.trading.allocationkeys

CreateAllocationKey

class saxo_openapi.endpoints.trading.allocationkeys.**CreateAllocationKey**(*data*)
Create an allocation key.

ENDPOINT = 'openapi/trade/v1/allocationkeys'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a CreateAllocationKey request.

Parameters **data** (*dict (required)*) – dict representing the parameters of the request-body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AllocationKeyName": "MyAllocation_Key",
          "AllocationUnitType": "Percentage",
          "MarginHandling": "Reduce",
          "OneTime": true,
          "OwnerAccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
          "ParticipatingAccountsInfo": [
            {
              "AcceptRemainderAmount": true,
              "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
              "Priority": 1,
              "UnitValue": 10.0
            },
            {
              "AcceptRemainderAmount": false,
              "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
              "Priority": 1,
              "UnitValue": 90.0
            }
          ]
        }
```

```
>>> r = tr.allocationkeys.CreateAllocationKey(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "AllocationKeyId": "227"
}
```

expected_status

response
response - get the response of the request.

status_code

DeleteAllocationKey

class saxo_openapi.endpoints.trading.allocationkeys.**DeleteAllocationKey**(*AllocationKeyId*)
Delete an allocation key.

ENDPOINT = 'openapi/trade/v1/allocationkeys/{AllocationKeyId}'

EXPECTED_STATUS = 204

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__ (*AllocationKeyId*)
Instantiate a DeleteAllocationKey request.

Parameters **AllocationKeyId** (*string (required)*) – the allocation key id.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> AllocationKeyId = 227
>>> r = tr.allocationkeys.DeleteAllocationKey(
...     AllocationKeyId=AllocationKeyId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

expected_status

response
response - get the response of the request.

status_code

GetAllocationKeyDetails

class saxo_openapi.endpoints.trading.allocationkeys.**GetAllocationKeyDetails** (*AllocationKeyId*)
Get detailed information about an allocation key.

ENDPOINT = 'openapi/trade/v1/allocationkeys/{AllocationKeyId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__ (*AllocationKeyId*)
Instantiate a GetAllocationKeyDetails request.

Parameters **AllocationKeyId** (*string (required)*) – the allocation key id.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> AllocationKeyId = 227
>>> r = tr.allocationkeys.GetAllocationKeyDetails(
...     AllocationKeyId=AllocationKeyId)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "AllocationKeyId": "227",
  "AllocationKeyName": "MyAllocation_Key",
  "AllocationUnitType": "Percentage",
  "MarginHandling": "Reduce",
  "OwnerAccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
  "ParticipatingAccountsInfo": [
    {
      "AcceptRemainderAmount": true,
```

(continues on next page)

(continued from previous page)

```
      "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
      "Priority": 1,
      "UnitValue": 10.0
    },
    {
      "AcceptRemainderAmount": false,
      "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
      "Priority": 1,
      "UnitValue": 90.0
    }
  ],
  "Status": "Active"
}
```

expected_status

response
response - get the response of the request.

status_code

GetAllocationKeys

class saxo_openapi.endpoints.trading.allocationkeys.**GetAllocationKeys**(*params*)
Get a list of existing allocation keys. By default only Active allocation keys for the current client are returned.

ENDPOINT = 'openapi/trade/v1/allocationkeys'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a GetAllocationKeys request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "$top": 601,
          "$skip": 22968,
          "AccountKey": "...",
          "ClientKey": "...",
          "Statuses": "Active,OneTime"
        }
```

```
>>> r = tr.allocationkeys.GetAllocationKeys(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Data": [
    {
      "AllocationKeyId": "227",
      "AllocationKeyName": "MyAllocation_Key",
      "OwnerAccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
      "Status": "Active"
    },
    {
      "AllocationKeyId": "227_2",
      "AllocationKeyName": "MyAllocation_Key_2",
      "OwnerAccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
      "Status": "OneTime"
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

3.7.2 saxo_openapi.endpoints.trading.infoprices

CreateInfoPriceSubscription

class saxo_openapi.endpoints.trading.infoprices.**CreateInfoPriceSubscription**(*data*)
Sets up a subscription and returns an initial snapshot of an info price list specified by the parameters in the request.

ENDPOINT = 'openapi/trade/v1/infoprices/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a CreateInfoPriceSubscription request.

Parameters **data** (*dict (required)*) – dict representing the parameters of the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
            "AssetType": "FxSpot",
            "FieldGroups": [
              "DisplayAndFormat",
              "HistoricalChanges",
              "InstrumentPriceDetails",
              "PriceInfo",
              "PriceInfoDetails",
```

(continues on next page)

(continued from previous page)

```
            "Quote"
          ],
          "Uics": "22,23"
        },
        "ContextId": "20190307094456688",
        "ReferenceId": "IP17820",
        "RefreshRate": 1000
      }
```

```
>>> r = tr.infoprices.CreateInfoPriceSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "ContextId": "20190307094456688",
  "Format": "application/json",
  "InactivityTimeout": 60,
  "ReferenceId": "IP17820",
  "RefreshRate": 1000,
  "Snapshot": {
    "Data": [
      {
        "AssetType": "FxSpot",
        "DisplayAndFormat": {
          "Currency": "AUD",
          "Decimals": 4,
          "Description": "British Pound/Australian Dollar",
          "Format": "AllowDecimalPips",
          "OrderDecimals": 4,
          "Symbol": "GBPAUD"
        },
        "HistoricalChanges": {
          "PercentChange1Month": 1.21,
          "PercentChange2Months": 2.95,
          "PercentChange3Months": 1.85,
          "PercentChange6Months": -1.83,
          "PercentChangeWeekly": 1.67
        },
        "InstrumentPriceDetails": {
          "IsMarketOpen": true,
          "ShortTradeDisabled": false,
          "ValueDate": "2017-05-19"
        },
        "LastUpdated": "0001-01-01T00:00:00Z",
        "PriceInfo": {
          "High": 1.09117,
          "Low": 1.08853,
          "NetChange": 0.00048,
          "PercentChange": 0.04
        },
        "PriceInfoDetails": {
          "AskSize": 1000000.0,
          "BidSize": 1000000.0,
          "LastClose": 1.08932,
```

(continues on next page)

(continued from previous page)


```
          "LastTraded": 0.0,
          "LastTradedSize": 0.0,
          "Open": 0.0,
          "Volume": 0.0
        },
        "Quote": {
          "Amount": 100000,
          "Ask": 1.74948,
          "Bid": 1.74858,
          "DelayedByMinutes": 15,
          "ErrorCode": "None",
          "Mid": 1.74903,
          "PriceTypeAsk": "Indicative",
          "PriceTypeBid": "Indicative"
        },
        "Uic": 22
      },
      {
        "AssetType": "FxSpot",
        "DisplayAndFormat": {
          "Currency": "CAD",
          "Decimals": 4,
          "Description": "British Pound/Canadian Dollar",
          "Format": "AllowDecimalPips",
          "OrderDecimals": 4,
          "Symbol": "GBPCAD"
        },
        "InstrumentPriceDetails": {
          "IsMarketOpen": true,
          "ShortTradeDisabled": false,
          "ValueDate": "2017-05-19"
        },
        "LastUpdated": "0001-01-01T00:00:00Z",
        "Quote": {
          "Amount": 100000,
          "Ask": 1.76278,
          "Bid": 1.76198,
          "DelayedByMinutes": 15,
          "ErrorCode": "None",
          "Mid": 1.76238,
          "PriceTypeAsk": "Indicative",
          "PriceTypeBid": "Indicative"
        },
        "Uic": 23
      }
    ]
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

InfoPrice

class saxo_openapi.endpoints.trading.infoprices.**InfoPrice**(*params*)

Gets an info price for an instrument using the specified parameters.

ENDPOINT = 'openapi/trade/v1/infoprices'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)

Instantiate an InfoPrice request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "Uic": 22,
          "AccountKey": "81ef1924-c25f-43fe-90ff-028e3fe249f2",
          "AssetType": "...",
          "<other-parms>": "..."
        }
```

```
>>> r = tr.infoprices.InfoPrice(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "AssetType": "FxSpot",
  "DisplayAndFormat": {
    "Currency": "AUD",
    "Decimals": 4,
    "Description": "British Pound/Australian Dollar",
    "Format": "AllowDecimalPips",
    "OrderDecimals": 4,
    "Symbol": "GBPAUD"
  },
  "HistoricalChanges": {
    "PercentChange1Month": 1.21,
    "PercentChange2Months": 2.95,
    "PercentChange3Months": 1.85,
    "PercentChange6Months": -1.83,
    "PercentChangeWeekly": 1.67
  },
  "InstrumentPriceDetails": {
    "IsMarketOpen": true,
    "ShortTradeDisabled": false,
    "ValueDate": "2017-05-19"
  },
  "LastUpdated": "0001-01-01T00:00:00Z",
  "PriceInfo": {
    "High": 1.09117,
```

(continues on next page)

(continued from previous page)

```
    "Low": 1.08853,
    "NetChange": 0.00048,
    "PercentChange": 0.04
  },
  "PriceInfoDetails": {
    "AskSize": 1000000.0,
    "BidSize": 1000000.0,
    "LastClose": 1.08932,
    "LastTraded": 0.0,
    "LastTradedSize": 0.0,
    "Open": 0.0,
    "Volume": 0.0
  },
  "Quote": {
    "Amount": 100000,
    "Ask": 1.74948,
    "Bid": 1.74858,
    "DelayedByMinutes": 15,
    "ErrorCode": "None",
    "Mid": 1.74903,
    "PriceTypeAsk": "Indicative",
    "PriceTypeBid": "Indicative"
  },
  "Uic": 22
}
```

expected_status

response
response - get the response of the request.

status_code

InfoPrices

class saxo_openapi.endpoints.trading.infoprices.**InfoPrices**(*params*)
Gets a list of info prices for a list of instruments using the specified parameters.

ENDPOINT = 'openapi/trade/v1/infoprices/list'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate an InfoPrices request.

Parameters **params** (*dict (required)*) – dict representing the querystring parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "Uics": "22,23",
          "AccountKey": "1a463418-88d4-4555-92e3-e6004d675245",
```

(continues on next page)

(continued from previous page)

```
            "<other-parms>": "..."
        }
```

```
>>> r = tr.infoprices.InfoPrices(params=params)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:


```
{
  "Data": [
    {
      "AssetType": "FxSpot",
      "DisplayAndFormat": {
        "Currency": "AUD",
        "Decimals": 4,
        "Description": "British Pound/Australian Dollar",
        "Format": "AllowDecimalPips",
        "OrderDecimals": 4,
        "Symbol": "GBPAUD"
      },
      "HistoricalChanges": {
        "PercentChange1Month": 1.21,
        "PercentChange2Months": 2.95,
        "PercentChange3Months": 1.85,
        "PercentChange6Months": -1.83,
        "PercentChangeWeekly": 1.67
      },
      "InstrumentPriceDetails": {
        "IsMarketOpen": true,
        "ShortTradeDisabled": false,
        "ValueDate": "2017-05-19"
      },
      "LastUpdated": "0001-01-01T00:00:00Z",
      "PriceInfo": {
        "High": 1.09117,
        "Low": 1.08853,
        "NetChange": 0.00048,
        "PercentChange": 0.04
      },
      "PriceInfoDetails": {
        "AskSize": 1000000.0,
        "BidSize": 1000000.0,
        "LastClose": 1.08932,
        "LastTraded": 0.0,
        "LastTradedSize": 0.0,
        "Open": 0.0,
        "Volume": 0.0
      },
      "Quote": {
        "Amount": 100000,
        "Ask": 1.74948,
        "Bid": 1.74858,
        "DelayedByMinutes": 15,
        "ErrorCode": "None",
        "Mid": 1.74903,
```


(continues on next page)

(continued from previous page)


```
        "PriceTypeAsk": "Indicative",
        "PriceTypeBid": "Indicative"
      },
      "Uic": 22
    },
    {
      "AssetType": "FxSpot",
      "DisplayAndFormat": {
        "Currency": "CAD",
        "Decimals": 4,
        "Description": "British Pound/Canadian Dollar",
        "Format": "AllowDecimalPips",
        "OrderDecimals": 4,
        "Symbol": "GBPCAD"
      },
      "InstrumentPriceDetails": {
        "IsMarketOpen": true,
        "ShortTradeDisabled": false,
        "ValueDate": "2017-05-19"
      },
      "LastUpdated": "0001-01-01T00:00:00Z",
      "Quote": {
        "Amount": 100000,
        "Ask": 1.76278,
        "Bid": 1.76198,
        "DelayedByMinutes": 15,
        "ErrorCode": "None",
        "Mid": 1.76238,
        "PriceTypeAsk": "Indicative",
        "PriceTypeBid": "Indicative"
      },
      "Uic": 23
    }
  ]
}
```


expected_status

response
response - get the response of the request.

status_code

RemoveInfoPriceSubscriptionById

class saxo_openapi.endpoints.trading.infoprices.**RemoveInfoPriceSubscriptionById**(*ContextId*, *ReferenceId*)

Remove an info price subscription on a single instrument.

ENDPOINT = 'openapi/trade/v1/infoprices/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a RemoveInfoPricesSubscriptionById request.

Parameters

- **ContextId** (*string (required)*) – the context id.
- **ReferenceId** (*string (required)*) – the ReferenceId id.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'ctxt_20190316'
>>> ReferenceId = 'pri_01'
>>> r = tr.infoprices.RemoveInfoPriceSubscriptionById(
...     ContextId=ContextId,
...     ReferenceId=ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

RemoveInfoPriceSubscriptionsByTag

class saxo_openapi.endpoints.trading.infoprices.**RemoveInfoPriceSubscriptionsByTag**(*ContextId*, *params=No*

Remove one or more infoprice subscriptions.

ENDPOINT = 'openapi/trade/v1/infoprices/subscriptions/{ContextId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params=None*)
Instantiate a RemoveInfoPriceSubscriptionsByTag request.

Parameters

- **ContextId** (*string (required)*) – the context id.
- **params** (*dict (required)*) – dict representing the querystring parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'ctxt_20190316'
>>> params =
        {
```

(continues on next page)

(continued from previous page)

```
          "Tag": "IP"
        }
```

```
>>> r = tr.infoprices.RemoveInfoPriceSubscriptionsByTag(
...     ContextId=ContextId,
...     params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.7.3 saxo_openapi.endpoints.trading.messages

CreateTradeMessageSubscription

class saxo_openapi.endpoints.trading.messages.**CreateTradeMessageSubscription**(*data*)
Create a subscription on trade messages.

ENDPOINT = 'openapi/trade/v1/messages/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a CreateTradeMessageSubscription request.

Parameters **data** (*dict (required)*) – dict representing the parameters of the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "ContextId": "20190307094456781",
          "Format": "application/json",
          "ReferenceId": "TM90172",
          "RefreshRate": 5,
          "Tag": "PAGE1"
        }
```

```
>>> r = tr.messages.CreateTradeMessageSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "ContextId": "20190307094456781",
  "Format": "application/json",
  "InactivityTimeout": 120,
  "ReferenceId": "TM90172",
  "RefreshRate": 800,
  "Snapshot": {
    "Data": [
      {
        "DateTime": "2014-12-12T09:17:12Z",
        "DisplayName": "Price Alert",
        "DisplayType": "Default",
        "IsDiscardable": false,
        "MessageBody": "Price alert was triggered on EURUSD",
        "MessageHeader": "Price Alert",
        "MessageId": "345322",
        "MessageType": "PriceAlert"
      }
    ]
  },
  "State": "Active",
  "Tag": "PAGE1"
}
```


expected_status

response
response - get the response of the request.

status_code

GetTradeMessages

class saxo_openapi.endpoints.trading.messages.**GetTradeMessages**
Get trade messages for the current user.

ENDPOINT = 'openapi/trade/v1/messages'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a GetTradeMessages request.

Parameters **None** –

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = tr.messages.GetTradeMessages()
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Data": [
    {
      "DateTime": "2014-12-12T09:17:12Z",
      "DisplayName": "Price Alert",
      "DisplayType": "Default",
      "IsDiscardable": false,
      "MessageBody": "Price alert was triggered on EURUSD",
      "MessageHeader": "Price Alert",
      "MessageId": "345322",
      "MessageType": "PriceAlert"
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

MarkMessageAsSeen

class saxo_openapi.endpoints.trading.messages.**MarkMessageAsSeen**(*MessageId*)
Logically this is done by moving the message to the 'seen' collection.

ENDPOINT = 'openapi/trade/v1/messages/seen/{MessageId}'

EXPECTED_STATUS = 204

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*MessageId*)
Instantiate a MarkMessageAsSeen request.

Parameters **MessageId** (*string (required)*) – the message-id of the message.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> MessageId = '...'
>>> r = tr.messages.MarkMessageAsSeen(MessageId=MessageId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

RemoveTradeMessageSubscriptionById

class saxo_openapi.endpoints.trading.messages.**RemoveTradeMessageSubscriptionById**(*ContextId*, *ReferenceId*)

Removes a trade message subscription for the current session.

ENDPOINT = 'openapi/trade/v1/messages/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)

Instantiate a RemoveTradeMessageSubscriptionById request.

Parameters

- **ContextId** (*string (required)*) – the context-id
- **ReferenceId** (*string (required)*) – the reference-id

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'ctxt_20190317'
>>> ReferenceId = '...'
>>> r = tr.messages.RemoveTradeMessageSubscriptionById(
...         ContextId=ContextId,
...         ReferenceId=ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

RemoveTradeMessageSubscriptions

class saxo_openapi.endpoints.trading.messages.**RemoveTradeMessageSubscriptions**(*ContextId*, *params*)

Removes trade message subscriptions for the current session.

ENDPOINT = 'openapi/trade/v1/messages/subscriptions/{ContextId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *params*)

Instantiate a RemoveTradeMessageSubscription request.

Parameters

- **ContextId** (*string (required)*) – the context-id
- **ReferenceId** (*string (required)*) – the reference-id

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = 'ctxt_20190317'
>>> params =
        {
          "Tag": "CORE"
        }
```

```
>>> r = tr.messages.RemoveTradeMessageSubscriptions(
...         ContextId=ContextId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.7.4 saxo_openapi.endpoints.trading.optionschain

OptionsChainSubscriptionCreate

class saxo_openapi.endpoints.trading.optionschain.**OptionsChainSubscriptionCreate**(*data*)
Create an active options chain subscription.

ENDPOINT = 'openapi/trade/v1/optionschain/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a OptionsChainSubscriptionCreate request.

Parameters **data** (*dict*) – the dict representing the parameters of the request body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
            "AssetType": "StockIndexOption",
            "Expiries": [
              {
```

(continues on next page)

(continued from previous page)

```
                "Index": 1,
                "StrikeStartIndex": 0
              }
            ],
            "Identifier": 18,
            "MaxStrikesPerExpiry": 3
          },
          "ContextId": "20190411051355700",
          "ReferenceId": "C0165000",
          "RefreshRate": 1000
        }
```

```
>>> r = tr.optionschain.OptionsChainSubscriptionCreate(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Arguments": {
    "AccountKey": "LZTc7DdejXODf-WS12aCyQ==",
    "AssetType": "StockIndexOption",
    "Expiries": [
      {
        "Index": 1,
        "StrikeStartIndex": 0
      }
    ],
    "Identifier": 18,
    "MaxStrikesPerExpiry": 3
  },
  "ContextId": "20190411051355700",
  "ReferenceId": "C0165000",
  "RefreshRate": 1000
}
```

expected_status

response
response - get the response of the request.

status_code

OptionsChainSubscriptionModify

class saxo_openapi.endpoints.trading.optionschain.**OptionsChainSubscriptionModify**(*ContextId*, *ReferenceId*, *data*)

Modify an existing options chain subscription.

ENDPOINT = 'openapi/trade/v1/optionschain/subscriptions{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*, *data*)
Instantiate a OptionsChainSubscriptionModify request.

Parameters

- **ContextId** (*string*) – the ContextId
- **ReferenceId** (*string*) – the ReferenceId
- **data** (*dict*) – dict representing the parameters of the request body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ReferenceId = ...
>>> ContextId = ...
>>> data =
        {
          "Expiries": [
            {
              "Index": 129,
              "StrikeStartIndex": 232
            }
          ],
          "MaxStrikesPerExpiry": 157
        }
```

```
>>> r = tr.optionschain.OptionsChainSubscriptionModify(
...     ReferenceId=ReferenceId,
...     ContextId=ContextId,
...     data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

OptionsChainSubscriptionRemove

class saxo_openapi.endpoints.trading.optionschain.**OptionsChainSubscriptionRemove**(*ContextId*, *ReferenceId*)

Remove an options chain subscription.

ENDPOINT = 'openapi/trade/v1/optionschain/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a OptionsChainSubscriptionRemove request.

Parameters

- **ContextId** (*string*) – the ContextId
- **ReferenceId** (*string*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ReferenceId = ...
>>> ContextId = ...
>>> r = tr.optionschain.OptionsChainSubscriptionRemove(
...     ReferenceId=ReferenceId,
...     ContextId=ContextId
... )
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

OptionsChainSubscriptionResetATM

class saxo_openapi.endpoints.trading.optionschain.**OptionsChainSubscriptionResetATM**(*ContextId*, *ReferenceId*)

Reset an options chain subscription 'At The Money'.

ENDPOINT = 'openapi/trade/v1/optionschain/subscriptions/{ContextId}/{ReferenceId}/Reset

EXPECTED_STATUS = 204

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate an OptionsChainSubscriptionResetATM request.

Parameters

- **ContextId** (*string*) – the ContextId
- **ReferenceId** (*string*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ReferenceId = ...
>>> ContextId = ...
>>> r = tr.optionschain.OptionsChainSubscriptionResetATM(
...     ReferenceId=ReferenceId,
...     ContextId=ContextId
... )
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

3.7.5 saxo_openapi.endpoints.trading.orders

CancelOrders

class saxo_openapi.endpoints.trading.orders.**CancelOrders**(*OrderIds*, *params*)
Cancel one or more orders.

ENDPOINT = 'openapi/trade/v2/orders/{OrderIds}'

EXPECTED_STATUS = 200

METHOD = 'DELETE'

__init__(*OrderIds*, *params*)
Instantiate a CancelOrders request.

Parameters

- **OrderIds** (*string (required)*) – ',' delimited string with one or more orderId's
- **params** (*dict (required)*) – dict representing the querystring parameters.

params example:

```
params =
    {
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA=="
    }
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> OrderIds="76289286"
>>> params = ...
>>> r = tr.orders.CancelOrders(OrderIds=OrderIds, params=params)
```

(continues on next page)

(continued from previous page)

```
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "Orders": [
    {
      "OrderId": "76289286"
    }
  ]
}
```

expected_status

response
response - get the response of the request.

status_code

ChangeOrder

class saxo_openapi.endpoints.trading.orders.**ChangeOrder**(*data*)
Change one or more existing orders.

ENDPOINT = 'openapi/trade/v2/orders'

EXPECTED_STATUS = 200

METHOD = 'PATCH'

__init__(*data*)
Instantiate a ChangeOrder request.

Parameters **data** (*dict (required)*) – dict representing the data body, in this case an order change spec.

OrderBody example:

```
data =
    {
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": 40000.0,
      "AssetType": "FxSpot",
      "OrderId": "76289286",
      "OrderType": "Limit",
      "ManualOrder": false,
      "OrderDuration": {
        "DurationType": "DayOrder"
      }
    }
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data = ...
```

(continues on next page)

(continued from previous page)

```
>>> r = tr.orders.ChangeOrder(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "OrderId": "76289286"
}
```

expected_status

response
response - get the response of the request.

status_code

Order

class saxo_openapi.endpoints.trading.orders.**Order**(*data*)
Place a new order.

ENDPOINT = 'openapi/trade/v2/orders'

EXPECTED_STATUS = 200

METHOD = 'POST'

__init__(*data*)
Instantiate an Order request.

Parameters **data** (*dict (required)*) – dict representing the data body, in this case an order spec.

OrderBody example:

```
data =
    {
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "ManualOrder": false,
      "Amount": "10000",
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "OrderType": "Market",
      "Uic": 31
    }
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data = ...
>>> r = tr.orders.Order(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "OrderId": "76288494"
}
```

expected_status

response
response - get the response of the request.

status_code

PrecheckOrder

class saxo_openapi.endpoints.trading.orders.**PrecheckOrder**(*data*)
Precheck an order.

ENDPOINT = 'openapi/trade/v2/orders/precheck'

EXPECTED_STATUS = 200

METHOD = 'POST'

__init__(*data*)
Instantiate a PrecheckOrder request.

Parameters **data** (*dict (required)*) – dict representing the data body, in this case an order change spec.

data example:

```
data =
    {
      "AccountKey": "Cf4xZWiYL6W1nMKpygBLLA==",
      "Amount": "10000",
      "AssetType": "FxSpot",
      "BuySell": "Buy",
      "OrderType": "Market",
      "Uic": 31
    }
```

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data = ...
>>> r = tr.orders.PrecheckOrder(data=data)
>>> client.request(r)
>>> print(json.dumps(r.response, indent=4))
```

Output:

```
{
  "EstimatedCashRequired": 296.275,
  "EstimatedCashRequiredCurrency": "EUR",
  "InstrumentToAccountConversionRate": 0.88239,
  "PreCheckResult": "Ok"
}
```

expected_status

response
response - get the response of the request.

status_code

3.7.6 saxo_openapi.endpoints.trading.positions

ExerciseAmount

class saxo_openapi.endpoints.trading.positions.**ExerciseAmount**(*data*)
Forces exercise of an amount across all positions for the specified UIC. This is relevant for Futures Options, Stock Options, Stock Index Options.

ENDPOINT = 'openapi/trade/v1/positions/exercise'

EXPECTED_STATUS = 204

METHOD = 'PUT'

__init__(*data*)
Instantiate an ExerciseAmount request.

Parameters **data** (*dict (required)*) – dict representing the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
          "Amount": 100,
          "AssetType": "StockOption",
          "Uic": 5455848
        }
```

```
>>> r = tr.positions.ExerciseAmount(data=data)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "PositionId": "1019942426"
}
```

expected_status

response
response - get the response of the request.

status_code

ExercisePosition

class saxo_openapi.endpoints.trading.positions.**ExercisePosition**(*PositionId*, *data*)
Forces exercise of a position. This is relevant for Futures Options, Stock Options, Stock Index Options.

ENDPOINT = 'openapi/trade/v1/positions/{PositionId}/exercise'

EXPECTED_STATUS = 204

METHOD = 'PUT'

__init__(*PositionId*, *data*)
Instantiate an ExercisePosition request.

Parameters

- **PositionId** (*string (required)*) – the position id
- **data** (*dict (required)*) – dict representing the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
          "Amount": 100,
          "AssetType": "StockOption",
          "Uic": 5455848
        }
```

```
>>> PositionId = 1019942425
>>> r = tr.positions.ExercisePosition(PositionId, data=data)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "PositionId": "1019942426"
}
```

expected_status

response
response - get the response of the request.

status_code

PositionByQuote

class saxo_openapi.endpoints.trading.positions.**PositionByQuote**(*data*)
Creates a new position by accepting a quote.

The quote must be the most recent one and it must be tradable: (Quote.PriceType=PriceType.Tradable).

ENDPOINT = 'openapi/trade/v1/positions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a PositionByQuote request.

Parameters **data** (*dict (required)*) – dict representing the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AppHint": 12,
          "BuySell": "Buy",
          "PriceReferenceId": "P57629",
          "QuoteId": "1232145",
          "UserPrice": 234.1
        }
```

```
>>> r = tr.positions.PositionByQuote(data=data)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "PositionId": "1019942426"
}
```

expected_status

response
response - get the response of the request.

status_code

UpdatePosition

class saxo_openapi.endpoints.trading.positions.**UpdatePosition**(*PositionId*, *data*)
Updates properties of an existing position. This is only relevant for FX Options, where you can update the Exercise method.

ENDPOINT = 'openapi/trade/v1/positions/{PositionId}'

EXPECTED_STATUS = 204

METHOD = 'PATCH'

__init__(*PositionId*, *data*)
Instantiate an UpdatePosition request.

Parameters

- **PositionId** (*string (required)*) – the position id
- **data** (*dict (required)*) – dict representing the data body.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AccountKey": "LZTc7DdejXODf-WSl2aCyQ==",
          "ExerciseMethod": "Spot"
        }
```

```
>>> PositionId = 1019942425
>>> r = tr.positions.UpdatePosition(PositionId, data=data)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "PositionId": "1019942426"
}
```

expected_status

response
response - get the response of the request.

status_code

3.7.7 saxo_openapi.endpoints.trading.prices

CreatePriceSubscription

class saxo_openapi.endpoints.trading.prices.**CreatePriceSubscription**(*data*)
Sets up an active price subscription on an instrument and returns an initial snapshot of the most recent price.

ENDPOINT = 'openapi/trade/v1/prices/subscriptions'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a CreatePriceSubscription request.

Parameters **data** (*dict (required)*) – dict representing the data body, in this case an order spec.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "Arguments": {
            "Uic": 45,
            "AssetType": "FxSpot"
          },
          "ContextId": "explorer_1552305379377",
          "ReferenceId": "USDSGD"
        }
```

```
>>> r = tr.prices.CreatePriceSubscription(data=data)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "ContextId": "explorer_1552305379377",
  "Format": "application/json",
  "InactivityTimeout": 30,
```

(continues on next page)

(continued from previous page)


```
  "ReferenceId": "N_935",
  "RefreshRate": 1000,
  "Snapshot": {
    "AssetType": "FxSpot",
    "LastUpdated": "2019-03-11T21:14:00.578000Z",
    "PriceSource": "SBFX",
    "Quote": {
      "Amount": 100000,
      "Ask": 1.35827,
      "Bid": 1.35754,
      "DelayedByMinutes": 0,
      "ErrorCode": "None",
      "Mid": 1.357905,
      "PriceTypeAsk": "Tradable",
      "PriceTypeBid": "Tradable",
      "RFQState": "None"
    },
    "Uic": 45
  },
  "State": "Active"
}
```


expected_status

response
response - get the response of the request.

status_code

MarginImpactRequest

class saxo_openapi.endpoints.trading.prices.**MarginImpactRequest**(*ContextId*, *ReferenceId*)

Request margin impact to come on one of the next following price updates.

ENDPOINT = 'openapi/trade/v1/prices/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 204

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a MarginImpactRequest request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = "ctxt_20190311"
>>> ReferenceId = "EURUSD"
>>> r = tr.prices.MarginImpactRequest(ContextId=ContextId,
```

(continues on next page)

(continued from previous page)

```
...                                          ReferenceId=ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

PriceSubscriptionRemove

class saxo_openapi.endpoints.trading.prices.**PriceSubscriptionRemove**(*ContextId*, *ReferenceId*)

Removes subscription for the current session identified by subscription id.

ENDPOINT = 'openapi/trade/v1/prices/subscriptions/{ContextId}/{ReferenceId}'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__(*ContextId*, *ReferenceId*)
Instantiate a PriceSubscriptionRemove request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **ReferenceId** (*string (required)*) – the ReferenceId

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> ContextId = ...
>>> ReferenceId = ...
>>> r = tr.prices.PriceSubscriptionRemove(ContextId, ReferenceId)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response
response - get the response of the request.

status_code

PriceSubscriptionRemoveByTag

class saxo_openapi.endpoints.trading.prices.**PriceSubscriptionRemoveByTag**(*ContextId*, *params=None*)

Remove multiple subscriptions for the given ContextId, optionally marked with a specific tag.

ENDPOINT = 'openapi/trade/v1/prices/subscriptions/{ContextId}/'

EXPECTED_STATUS = 202

METHOD = 'DELETE'

RESPONSE_DATA = None

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

Instantiate a PriceSubscriptionRemoveByTag request.

Parameters

- **ContextId** (*string (required)*) – the ContextId
- **params** (*dict (optional)*) – dict representing the querystring parameters

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.trading as tr
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {}
```

```
>>> ContextId = ...
>>> r = tr.prices.PriceSubscriptionRemoveByTag(ContextId,
...                                            params=params)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No data is returned.

expected_status

response

response - get the response of the request.

status_code

3.8 saxo_openapi.endpoints.valueadd

3.8.1 saxo_openapi.endpoints.valueadd.pricealerts

CreatePriceAlert

class saxo_openapi.endpoints.valueadd.pricealerts.**CreatePriceAlert**(*data*)

Create a new price alert definition. The created definition is returned with a couple of more properties, the price alert definition ID being one of them.

ENDPOINT = 'openapi/vas/v1/pricealerts/definitions/'

EXPECTED_STATUS = 201

METHOD = 'POST'

__init__(*data*)
Instantiate a CreatePriceAlert request.

Parameters **data** (*dict (required)*) – dict representing the body parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "AccountId": "13457INET",
          "AssetType": "FxSpot",
          "Comment": "I believe EURUSD will go up within the next few years!",
          "ExpiryDate": "2016-09-30T12:00:00Z",
          "IsRecurring": true,
          "Operator": "GreaterOrEqual",
          "PriceVariable": "AskTick",
          "State": "Enabled",
          "TargetValue": 1.34595,
          "Uic": 21
        }
```

```
>>> r = va.pricealerts.CreatePriceAlert(AlertDefinitionId)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "AccountId": "13457INET",
  "AlertDefinitionId": "30834",
  "AssetType": "FxSpot",
  "Comment": "I believe EURUSD will go up within the next few years!",
  "ExpiryDate": "2016-09-30T12:00:00Z",
  "IsRecurring": true,
  "Operator": "GreaterOrEqual",
  "PriceVariable": "AskTick",
  "State": "Enabled",
  "TargetValue": 1.34595,
  "Uic": 21,
  "UserId": "2361528"
}
```

expected_status

response
response - get the response of the request.

status_code

DeletePriceAlert

class saxo_openapi.endpoints.valueadd.pricealerts.**DeletePriceAlert**(*AlertDefinitionIds*)
Delete the specified price alert definitions. The alerts have to belong to the current user.

ENDPOINT = 'openapi/vas/v1/pricealerts/definitions/{AlertDefinitionIds}'

EXPECTED_STATUS = 204

METHOD = 'DELETE'

RESPONSE_DATA = None

__init__ (*AlertDefinitionIds*)
Instantiate a DeletePriceAlert request.

Parameters **AlertDefinitionIds** (*string (required)*) – string with ‘,’-delimited AlertDefinitionIds

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> AlertDefinitionIds = '30384,30386'
>>> r = va.pricealerts.DeletePriceAlert(AlertDefinitionIds)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response
response - get the response of the request.

status_code

GetAlertDefinition

class saxo_openapi.endpoints.valueadd.pricealerts.**GetAlertDefinition**(*AlertDefinitionId*)
Gets the specified price alert for the current user.

ENDPOINT = 'openapi/vas/v1/pricealerts/definitions/{AlertDefinitionId}'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__ (*AlertDefinitionId*)
Instantiate a GetAlertDefinition request.

Parameters **AlertDefinitionId** (*string (required)*) – the alert definition id.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> AlertDefinitionId = 30384
>>> r = va.pricealerts.GetAlertDefinition(AlertDefinitionId)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "AccountId": "13457INET",
  "AlertDefinitionId": "30834",
  "AssetType": "FxSpot",
  "Comment": "I believe EURUSD will go up within the next few years!",
  "ExpiryDate": "2016-09-30T12:00:00Z",
  "IsRecurring": true,
  "Operator": "GreaterOrEqual",
  "PriceVariable": "AskTick",
```

(continues on next page)

(continued from previous page)

```
  "State": "Enabled",
  "TargetValue": 1.34595,
  "Uic": 21,
  "UserId": "2361528"
}
```

expected_status

response
response - get the response of the request.

status_code

GetAllAlerts

class saxo_openapi.endpoints.valueadd.pricealerts.**GetAllAlerts**(*params*)
Get an unsorted list of all the price alert definitions belonging to the current user where the state matches the specified value. The alerts might belong to different accounts, should the user have more than one.

ENDPOINT = 'openapi/vas/v1/pricealerts/definitions/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__(*params*)
Instantiate a GetAllAlerts request.

Parameters **data** (*dict (required)*) – dict representing the querystring parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> params =
        {
          "$top": 12015,
          "$skip": 27782,
          "$inlinecount": "None",
          "State": "Enabled"
        }
```

```
>>> r = va.pricealerts.GetAllAlerts(params=params)
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```

```
{
  "Data": [
    {
      "AccountId": "13457INET",
      "AlertDefinitionId": "30834",
      "AssetType": "FxSpot",
      "Comment": "I believe EURUSD will go up within the next few years!",
      "ExpiryDate": "2016-09-30T12:00:00Z",
      "IsRecurring": true,
      "Operator": "GreaterOrEqual",
      "PriceVariable": "AskTick",
```

(continues on next page)

(continued from previous page)


```
      "State": "Enabled",
      "TargetValue": 1.34595,
      "Uic": 21,
      "UserId": "2361528"
    }
  ],
  "MaxRows": 206
}
```


expected_status

response
response - get the response of the request.

status_code

GetUserNotificationSettings

class saxo_openapi.endpoints.valueadd.pricealerts.**GetUserNotificationSettings**
Get the current user's price alert notification settings.

ENDPOINT = 'openapi/vas/v1/pricealerts/usersettings/'

EXPECTED_STATUS = 200

METHOD = 'GET'

__init__()
Instantiate a GetUserNotificationSettings request.

Parameters **None** –

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> r = va.pricealerts.GetUserNotificationSettings()
>>> client.request(r)
>>> print(json.dumps(rv, indent=2))
```


```
{
  "EmailAddress": "john.doe@broker.com",
  "EmailAddressValidated": false,
  "NotifyWithMail": true,
  "NotifyWithPopup": true,
  "Sound": "None"
}
```


expected_status

response
response - get the response of the request.

status_code

ModifyUserNotificationSettings

class saxo_openapi.endpoints.valueadd.pricealerts.**ModifyUserNotificationSettings**(*data*)

Modify the current user's price alert notification settings.

ENDPOINT = 'openapi/vas/v1/pricealerts/usersettings/'

EXPECTED_STATUS = 204

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*data*)

Instantiate a ModifyUserNotificationSettings request.

Parameters **data** (*dict (required)*) – dict representing the body parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> data =
        {
          "EmailAddress": "john.doe@broker.com",
          "EmailAddressValidated": false,
          "NotifyWithMail": true,
          "NotifyWithPopup": true,
          "Sound": "None"
        }
```

```
>>> r = va.pricealerts.ModifyUserNotificationSettings(data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response

response - get the response of the request.

status_code

UpdatePriceAlert

class saxo_openapi.endpoints.valueadd.pricealerts.**UpdatePriceAlert**(*AlertDefinitionId*, *data*)

Update a price alert definition for the current user.

ENDPOINT = 'openapi/vas/v1/pricealerts/definitions/{AlertDefinitionId}'

EXPECTED_STATUS = 204

METHOD = 'PUT'

RESPONSE_DATA = None

__init__(*AlertDefinitionId*, *data*)

Instantiate an UpdatePriceAlert request.

Parameters **data** (*dict (required)*) – dict representing the body parameters.

```
>>> import saxo_openapi
>>> import saxo_openapi.endpoints.valueadd as va
>>> import json
>>> client = saxo_openapi.API(access_token=...)
>>> AlertDefinitionId = 30384
>>> data =
        {
          "AccountId": "13457INET",
          "AssetType": "FxSpot",
          "Comment": "I believe EURUSD will go up within the next few years!",
          "ExpiryDate": "2016-09-30T12:00:00Z",
          "IsRecurring": true,
          "Operator": "GreaterOrEqual",
          "PriceVariable": "AskTick",
          "State": "Enabled",
          "TargetValue": 1.34595,
          "Uic": 21
        }
```

```
>>> r = va.pricealerts.UpdatePriceAlert(AlertDefinitionId, data=data)
>>> client.request(r)
>>> assert r.status_code == r.expected_status
```

No response data is returned.

expected_status

response

response - get the response of the request.

status_code

CHAPTER 4

saxo_openapi.definitions

The `saxo_openapi.definitions` module holds all the definitions as in the definitions appear throughout the documentation, see developer.saxo.

4.1 saxo_openapi.definitions.accounthistory

AccountHistory Definitions.

class saxo_openapi.definitions.accounthistory.**InlineCountValue**

Bases: `object`

Definition representation of InlineCountValue

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

```
>>> import saxo_openapi.definitions.accounthistory as defaccounthistory
>>> print(defaccounthistory.InlineCountValue.AllPages)
AllPages
>>> c = defaccounthistory.InlineCountValue()
>>> print(c[c.AllPages])
The results will contain a total count of items in the queried dataset.
>>> # or
>>> print(defaccounthistory.InlineCountValue().definitions[c.AllPages])
>>> # all keys
>>> print(defaccounthistory.InlineCountValue().definitions.keys())
>>> ...
```

AllPages = 'AllPages'

None = 'None'

__getitem__(*definitionID*)

return description for definitionID.

definitions

readonly property holding definition dict.

class saxo_openapi.definitions.accounthistory.**AccountPerformanceStandardPeriod**
Bases: object

Definition representation of AccountPerformanceStandardPeriod

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

```
>>> import saxo_openapi.definitions.accounthistory as defaccounthistory
>>> print(defaccounthistory.AccountPerformanceStandardPeriod.AllTime)
AllTime
>>> c = defaccounthistory.AccountPerformanceStandardPeriod()
>>> print(c[c.AllTime])
All time account performance.
>>> # or
>>> print(defaccounthistory.AccountPerformanceStandardPeriod().definitions[c.
↪AllTime])
>>> # all keys
>>> print(defaccounthistory.AccountPerformanceStandardPeriod().definitions.keys())
>>> ...
```

AllTime = 'AllTime'

Month = 'Month'

Quarter = 'Quarter'

Year = 'Year'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.accounthistory.**AssetType**
Bases: object

Definition representation of AssetType

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

```
>>> import saxo_openapi.definitions.accounthistory as defaccounthistory
>>> print(defaccounthistory.AssetType.Name)
Name
>>> c = defaccounthistory.AssetType()
>>> print(c[c.Name])
Description
>>> # or
>>> print(defaccounthistory.AssetType().definitions[c.Name])
>>> # all keys
>>> print(defaccounthistory.AssetType().definitions.keys())
>>> ...
```

Bond = 'Bond'

Cash = 'Cash'

CfdIndexOption = 'CfdIndexOption'

CfdOnFutures = 'CfdOnFutures'

CfdOnIndex = 'CfdOnIndex'

CfdOnStock = 'CfdOnStock'

ContractFutures = 'ContractFutures'

FuturesOption = 'FuturesOption'

FuturesStrategy = 'FuturesStrategy'

FxBinaryOption = 'FxBinaryOption'

FxForwards = 'FxForwards'

FxKnockInOption = 'FxKnockInOption'

FxKnockOutOption = 'FxKnockOutOption'

FxNoTouchOption = 'FxNoTouchOption'

FxOneTouchOption = 'FxOneTouchOption'

FxSpot = 'FxSpot'

FxVanillaOption = 'FxVanillaOption'

ManagedFund = 'ManagedFund'

MutualFund = 'MutualFund'

Name = 'Name'

Stock = 'Stock'

StockIndex = 'StockIndex'

StockIndexOption = 'StockIndexOption'

StockOption = 'StockOption'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.accounthistory.**AccountPerformanceFieldGroup**
Bases: object

Definition representation of AccountPerformanceFieldGroup

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

```
>>> import saxo_openapi.definitions.accounthistory as defaccounthistory
>>> print(defaccounthistory.AccountPerformanceFieldGroup.AccountSummary)
AccountSummary
>>> c = defaccounthistory.AccountPerformanceFieldGroup()
>>> print(c[c.AccountSummary])
```

```
>>> # or
>>> print(defaccounthistory.AccountPerformanceFieldGroup().definitions[c.
↪AccountSummary])
>>> # all keys
>>> print(defaccounthistory.AccountPerformanceFieldGroup().definitions.keys())
>>> ...
```

AccountSummary = 'AccountSummary'

All = 'All'

Allocation = 'Allocation'

AvailableBenchmarks = 'AvailableBenchmarks'

BalancePerformance = 'BalancePerformance'

BalancePerformance_AccountValueTimeSeries = 'BalancePerformance_AccountValueTimeSeries

BenchMark = 'BenchMark'

BenchmarkPerformance = 'BenchmarkPerformance'

TimeWeightedPerformance = 'TimeWeightedPerformance'

TimeWeightedPerformance_AccumulatedTimeWeightedTimeSeries = 'TimeWeightedPerformance_A

TotalCashBalancePerCurrency = 'TotalCashBalancePerCurrency'

TotalPositionsValuePerCurrency = 'TotalPositionsValuePerCurrency'

TotalPositionsValuePerProductPerSecurity = 'TotalPositionsValuePerProductPerSecurity'

TradeActivity = 'TradeActivity'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.2 saxo_openapi.definitions.orders

Order related definitions.

class saxo_openapi.definitions.orders.**AmountType**
Bases: object

Definition representation of AmountType

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

```
>>> import saxo_openapi.definitions.orders as deforders
>>> print(deforders.AmountType.CashAmount)
CashAmount
>>> c = deforders.AmountType()
>>> print(c[c.CashAmount])
Order amount is specified as a monetary value
>>> # or
>>> print(deforders.AmountType().definitions[c.CashAmount])
>>> # all keys
>>> print(deforders.AmountType().definitions.keys())
>>> ...
```

CashAmount = 'CashAmount'

Quantity = 'Quantity'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.orders.**AssetType**

Bases: object

Definition representation of AssetType

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

```
>>> import saxo_openapi.definitions.orders as deforders
>>> print(deforders.AssetType.Bond)
Bond
>>> c = deforders.AssetType()
>>> print(c[c.Bond])
Bond
>>> # or
>>> print(deforders.AssetType().definitions[c.Bond])
>>> # all keys
>>> print(deforders.AssetType().definitions.keys())
>>> ...
```

Bond = 'Bond'

CfdIndexOption = 'CfdIndexOption'

CfdOnFutures = 'CfdOnFutures'

CfdOnIndex = 'CfdOnIndex'

CfdOnStock = 'CfdOnStock'

ContractFutures = 'ContractFutures'

FuturesOption = 'FuturesOption'

FuturesStrategy = 'FuturesStrategy'

FxBinaryOption = 'FxBinaryOption'

FxForwards = 'FxForwards'

FxKnockInOption = 'FxKnockInOption'

FxKnockOutOption = 'FxKnockOutOption'

FxNoTouchOption = 'FxNoTouchOption'

FxOneTouchOption = 'FxOneTouchOption'

FxSpot = 'FxSpot'

FxVanillaOption = 'FxVanillaOption'

ManagedFund = 'ManagedFund'

MutualFund = 'MutualFund'

Stock = 'Stock'

StockIndex = 'StockIndex'

StockIndexOption = 'StockIndexOption'

StockOption = 'StockOption'

__getitem__(*definitionID*)

return description for definitionID.

definitions

readonly property holding definition dict.

class saxo_openapi.definitions.orders.**Direction**

Bases: object

Definition representation of Direction

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

```
>>> import saxo_openapi.definitions.orders as deforders
>>> print(deforders.Direction.Buy)
Buy
>>> c = deforders.Direction()
>>> print(c[c.Buy])
Buy
>>> # or
>>> print(deforders.Direction().definitions[c.Buy])
>>> # all keys
>>> print(deforders.Direction().definitions.keys())
>>> ...
```

Buy = 'Buy'

Sell = 'Sell'

__getitem__(*definitionID*)

return description for definitionID.

definitions

readonly property holding definition dict.

class saxo_openapi.definitions.orders.**OrderDurationType**

Bases: object

Definition representation of OrderDurationType

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

```
>>> import saxo_openapi.definitions.orders as deforders
>>> print(deforders.OrderDurationType.AtTheClose)
AtTheClose
>>> c = deforders.OrderDurationType()
>>> print(c[c.AtTheClose])
At the close of the trading session
>>> # or
>>> print(deforders.OrderDurationType().definitions[c.AtTheClose])
>>> # all keys
>>> print(deforders.OrderDurationType().definitions.keys())
>>> ...
```

AtTheClose = 'AtTheClose'

AtTheOpening = 'AtTheOpening'

DayOrder = 'DayOrder'

FillOrKill = 'FillOrKill'

GoodForPeriod = 'GoodForPeriod'

GoodTillCancel = 'GoodTillCancel'

GoodTillDate = 'GoodTillDate'

ImmediateOrCancel = 'ImmediateOrCancel'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.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 saxo_openapi.definitions.orders as deforders
>>> print(deforders.OrderType.Algorithmic)
Algorithmic
>>> c = deforders.OrderType()
>>> print(c[c.Algorithmic])
Algo order
>>> # or
>>> print(deforders.OrderType().definitions[c.Algorithmic])
>>> # all keys
>>> print(deforders.OrderType().definitions.keys())
>>> ...
```

Algorithmic = 'Algorithmic'

Limit = 'Limit'

Market = 'Market'

Stop = 'Stop'

StopIfTraded = 'StopIfTraded'

StopLimit = 'StopLimit'

Switch = 'Switch'

TrailingStop = 'TrailingStop'

TrailingStopIfBid = 'TrailingStopIfBid'

TrailingStopIfOffered = 'TrailingStopIfOffered'

TrailingStopIfTraded = 'TrailingStopIfTraded'

Traspaso = 'Traspaso'

TraspasoIn = 'TraspasoIn'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.orders.**ToOpenClose**
Bases: object

Definition representation of ToOpenClose

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

```
>>> import saxo_openapi.definitions.orders as deforders
>>> print(deforders.ToOpenClose.ToClose)
ToClose
>>> c = deforders.ToOpenClose()
>>> print(c[c.ToClose])
Order/Position is ToClose
>>> # or
>>> print(deforders.ToOpenClose().definitions[c.ToClose])
>>> # all keys
>>> print(deforders.ToOpenClose().definitions.keys())
>>> ...
```

ToClose = 'ToClose'

ToOpen = 'ToOpen'

Undefined = 'Undefined'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.3 saxo_openapi.definitions.reportformats

Reportformat related definitions.

class saxo_openapi.definitions.reportformats.**AccountStatement**
Bases: object

Definition representation of AccountStatement

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

```
>>> import saxo_openapi.definitions.reportformats as defreportformats
>>> print(defreportformats.AccountStatement.PDF)
PDF
>>> c = defreportformats.AccountStatement()
>>> print(c[c.PDF])
PDF
>>> # or
>>> print(defreportformats.AccountStatement().definitions[c.PDF])
>>> # all keys
>>> print(defreportformats.AccountStatement().definitions.keys())
>>> ...
```

Excel = 'Excel'

PDF = 'PDF'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.reportformats.**PortfolioReport**
Bases: object

Definition representation of PortfolioReport

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

```
>>> import saxo_openapi.definitions.reportformats as defreportformats
>>> print(defreportformats.PortfolioReport.PDF)
PDF
>>> c = defreportformats.PortfolioReport()
>>> print(c[c.PDF])
PDF
>>> # or
>>> print(defreportformats.PortfolioReport().definitions[c.PDF])
>>> # all keys
>>> print(defreportformats.PortfolioReport().definitions.keys())
>>> ...
```

PDF = 'PDF'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.reportformats.**TradeDetailsReport**
Bases: object

Definition representation of TradeDetailsReport

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

```
>>> import saxo_openapi.definitions.reportformats as defreportformats
>>> print(defreportformats.TradeDetailsReport.PDF)
PDF
>>> c = defreportformats.TradeDetailsReport()
>>> print(c[c.PDF])
PDF
>>> # or
>>> print(defreportformats.TradeDetailsReport().definitions[c.PDF])
>>> # all keys
>>> print(defreportformats.TradeDetailsReport().definitions.keys())
>>> ...
```

PDF = 'PDF'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.reportformats.**TradesExecutedReport**
Bases: object

Definition representation of TradesExecutedReport

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

```
>>> import saxo_openapi.definitions.reportformats as defreportformats
>>> print(defreportformats.TradesExecutedReport.PDF)
PDF
```

(continues on next page)

(continued from previous page)

```
>>> c = defreportformats.TradesExecutedReport()
>>> print(c[c.PDF])
PDF
>>> # or
>>> print(defreportformats.TradesExecutedReport().definitions[c.PDF])
>>> # all keys
>>> print(defreportformats.TradesExecutedReport().definitions.keys())
>>> ...
```

Excel = 'Excel'

PDF = 'PDF'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.reportformats.**TransactionReport**
Bases: object

Definition representation of TransactionReport

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

```
>>> import saxo_openapi.definitions.reportformats as defreportformats
>>> print(defreportformats.TransactionReport.PDF)
PDF
>>> c = defreportformats.TransactionReport()
>>> print(c[c.PDF])
PDF
>>> # or
>>> print(defreportformats.TransactionReport().definitions[c.PDF])
>>> # all keys
>>> print(defreportformats.TransactionReport().definitions.keys())
>>> ...
```

Excel = 'Excel'

PDF = 'PDF'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.reportformats.**TransactionBalanceReport**
Bases: object

Definition representation of TransactionBalanceReport

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

```
>>> import saxo_openapi.definitions.reportformats as defreportformats
>>> print(defreportformats.TransactionBalanceReport.PDF)
PDF
>>> c = defreportformats.TransactionBalanceReport()
>>> print(c[c.PDF])
```

(continues on next page)

(continued from previous page)

```
PDF
>>> # or
>>> print(defreportformats.TransactionBalanceReport().definitions[c.PDF])
>>> # all keys
>>> print(defreportformats.TransactionBalanceReport().definitions.keys())
>>> ...
```

Excel = 'Excel'

PDF = 'PDF'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

4.4 saxo_openapi.definitions.activities

Activity related definitions.

class saxo_openapi.definitions.activities.**ActivityType**
Bases: object

Definition representation of ActivityType

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

```
>>> import saxo_openapi.definitions.activities as defactivities
>>> print(defactivities.ActivityType.AccountDepreciation)
AccountDepreciation
>>> c = defactivities.ActivityType()
>>> print(c[c.AccountDepreciation])
Account depreciation information
>>> # or
>>> print(defactivities.ActivityType().definitions[c.AccountDepreciation])
>>> # all keys
>>> print(defactivities.ActivityType().definitions.keys())
>>> ...
```

AccountDepreciation = 'AccountDepreciation'

AccountFundings = 'AccountFundings'

MarginCalls = 'MarginCalls'

Orders = 'Orders'

PositionDepreciation = 'PositionDepreciation'

Positions = 'Positions'

__getitem__(*definitionID*)
return description for definitionID.

definitions
readonly property holding definition dict.

class saxo_openapi.definitions.activities.**ActivityFieldGroup**

Bases: object

Definition representation of ActivityFieldGroup

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

```
>>> import saxo_openapi.definitions.activities as defactivities
>>> print(defactivities.ActivityFieldGroup.DisplayAndFormat)
DisplayAndFormat
>>> c = defactivities.ActivityFieldGroup()
>>> print(c[c.DisplayAndFormat])
Display and Format
>>> # or
>>> print(defactivities.ActivityFieldGroup().definitions[c.DisplayAndFormat])
>>> # all keys
>>> print(defactivities.ActivityFieldGroup().definitions.keys())
>>> ...
```

DisplayAndFormat = 'DisplayAndFormat'

ExchangeInfo = 'ExchangeInfo'

__getitem__(*definitionID*)

return description for definitionID.

definitions

readonly property holding definition dict.

CHAPTER 5

saxo_openapi contrib

5.1 saxo_openapi.contrib.orders

The *saxo_openapi.contrib.orders* package provides an abstractionlayer to create *orderbodies* with minimum effort.

5.1.1 LimitOrder

class saxo_openapi.contrib.orders.limitorder.**LimitOrder**(*Uic*, *Amount*, *AssetType*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

create a LimitOrder.

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

ALLOWED_DT = ['DayOrder', 'GoodTillDate', 'GoodTillCancel']

__init__(*Uic*, *Amount*, *AssetType*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

Instantiate a LimitOrder.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount

- **OrderPrice** (*decimal (required)*) – the price indicating the limitprice
- **AssetType** (*string (required)*) – the assettype for the Uic
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **AmountType** (*AmountType (optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification
- **OrderDurationType** (*string, default DayOrder*) – the order duration type, check SAXO Bank specs. for details
- **GTDDate** (*datetime string (required if order duration is GoodTillDate)*) – the GTD-datetime

Example

```
>>> import json
>>> from saxo_openapi import API
>>> import saxo_openapi.endpoints.trading as tr
>>> from saxo_openapi.contrib.orders import LimitOrder
>>>
>>> lo = LimitOrder(Uic=21,
...                 AssetType=OD.AssetType.FxSpot,
...                 Amount=10000,
...                 OrderPrice=1.1025)
>>> print(json.dumps(lo.data, indent=2))
{
  "Uic": 21,
  "AssetType": "FxSpot",
  "Amount": 10000,
  "Price": 1.1025,
  "BuySell": "Buy",
  "OrderType": "Limit",
  "ManualOrder": false,
  "AmountType": "Quantity",
  "OrderDuration": {
     "DurationType": "DayOrder"
  }
}
>>> # now we have the order specification, create the order request
>>> r = tr.orders.Order(data=lo.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(rv)
>>> print(json.dumps(rv, indent=4))
{
   "OrderId": "76697286"
}
```

data
data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.2 LimitOrderFxSpot

class saxo_openapi.contrib.orders.limitorder.**LimitOrderFxSpot**(*Uic*, *Amount*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

LimitOrderFxSpot - LimitOrder for FxSpot only.

The LimitOrderFxSpot lacks the AssetType parameter and only serves the AssetType FxSpot.

ALLOWED_DT = ['DayOrder', 'GoodTillDate', 'GoodTillCancel']

__init__(*Uic*, *Amount*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

Instantiate a LimitOrderFxSpot.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount
- **OrderPrice** (*decimal (required)*) – the price indicating the limitprice
- **AmountType** (AmountType *(optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification

- **OrderDurationType** (*string, default DayOrder*) – the order duration type, check SAXO Bank specs. for details
- **GTDDate** (*datetime string (required if order duration is GoodTillDate)*) – the GTD-datetime

Example

```
>>> from saxo_openapi import API
>>> from saxo_openapi.contrib.orders import (
...           tie_account_to_order,
...           LimitOrderFxSpot)
>>> token = "..."
>>> client = API(access_token=token)
>>> order = tie_account_to_order(
...     AccountKey,
...     LimitOrderFxSpot(Uic=21, Amount=25000, OrderPrice=1.1025))
>>> r = tr.orders.Order(data=order)
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
{
  "OrderId": "76703544"
}
```

data

data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.3 LimitOrderStock

class saxo_openapi.contrib.orders.limitorder.**LimitOrderStock**(*Uic*, *Amount*, *OrderPrice*, *AmountType='Quantity'*, *ManualOrder=False*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

LimitOrderStock - LimitOrder for Stock only.

The LimitOrderStock lacks the AssetType parameter and only serves the AssetType Stock.

ALLOWED_DT = ['DayOrder', 'GoodTillDate', 'GoodTillCancel']

__init__(*Uic*, *Amount*, *OrderPrice*, *AmountType='Quantity'*, *ManualOrder=False*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

Instantiate a LimitOrderStock.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount
- **OrderPrice** (*decimal (required)*) – the price indicating the limitprice
- **AmountType** (*AmountType (optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification
- **OrderDurationType** (*string, default DayOrder*) – the order duration type, check SAXO Bank specs. for details
- **GTDDate** (*datetime string (required if order duration is GoodTillDate)*) – the GTD-datetime

Example

```
>>> from saxo_openapi import API
>>> from saxo_openapi.contrib.orders import (
...           tie_account_to_order,
...           LimitOrderStock)
>>> token = "..."
>>> client = API(access_token=token)
>>> order = tie_account_to_order(
...     AccountKey,
...     LimitOrderStock(Uic=16350, Amount=1000, OrderPrice=28.00))
>>> r = tr.orders.Order(data=order)
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
{
  "OrderId": "76703539"
}
```

data

data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.4 MarketOrder

class saxo_openapi.contrib.orders.**MarketOrder**(*Uic*, *Amount*, *AssetType*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

create a MarketOrder.

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

__init__(*Uic*, *Amount*, *AssetType*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

Instantiate a MarketOrder.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount. A value > 0 means 'buy', a value < 0 means 'sell'
- **AssetType** (*string (required)*) – the assettype for the Uic
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **AmountType** (AmountType *(optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification

Example

```
>>> import json
>>> from saxo_openapi import API
>>> import saxo_openapi.endpoints.trading as tr
>>> from saxo_openapi.contrib.orders import MarketOrder
>>> # buy 10k EURUSD (Uic=21)
>>> mo = MarketOrder(Uic=21,
...                  AssetType=OD.AssetType.FxSpot,
...                  Amount=10000)
>>> print(json.dumps(mo.data, indent=4))
{
  "Uic": 21,
  "AssetType": "FxSpot",
  "Amount": 10000,
  "BuySell": "Buy",
  "OrderType": "Market",
  "AmountType": "Quantity",
  "ManualOrder": False,
```

(continues on next page)

(continued from previous page)

```
  "OrderDuration": {
      "DurationType": "DayOrder"
  }
}
>>> # now we have the order specification, create the order request
>>> r = tr.orders.Order(data=mo.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(rv)
>>> print(json.dumps(rv, indent=4))
{
   "OrderId": "76697286"
}
```

data
data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.5 MarketOrderFxSpot

class saxo_openapi.contrib.orders.**MarketOrderFxSpot**(*Uic*, *Amount*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

MarketOrderFxSpot - MarketOrder for FxSpot only.

The MarketOrderFxSpot lacks the AssetType parameter and only serves the AssetType FxSpot.

__init__(*Uic*, *Amount*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)
Instantiate a MarketOrderFxSpot.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount
- **AmountType** (*AmountType (optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification

Example

```
>>> from saxo_openapi import API
>>> from saxo_openapi.contrib.orders import (
...           tie_account_to_order,
...           MarketOrderFxSpot)
>>> token = "..."
>>> client = API(access_token=token)
>>> order = tie_account_to_order(
...                AccountKey,
...                MarketOrderFxSpot(Uic=21, Amount=25000))
>>> r = tr.orders.Order(data=req)
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
{
  "OrderId": "76703544"
}
```

data
data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.6 MarketOrderStock

class saxo_openapi.contrib.orders.**MarketOrderStock**(*Uic*, *Amount*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

MarketOrderStock - MarketOrder for Stock only.

The MarketOrderStock lacks the AssetType parameter and only serves the AssetType Stock.

__init__(*Uic*, *Amount*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)
Instantiate a MarketOrderStock.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount
- **AmountType** (*AmountType (optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the take-profit order specification

- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification

Example

```
>>> from saxo_openapi import API
>>> from saxo_openapi.contrib.orders import (
...          tie_account_to_order,
...          MarketOrderStock)
>>> token = "..."
>>> client = API(access_token=token)
>>> order = tie_account_to_order(
...                AccountKey,
...                MarketOrderStock(Uic=16350, Amount=1000))
>>> r = tr.orders.Order(data=req)
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
{
  "OrderId": "76703539"
}
```

data
data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.7 StopOrder

class saxo_openapi.contrib.orders.stoporder.**StopOrder**(*Uic*, *Amount*, *AssetType*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

create a StopOrder.

StopOrder is used to build the body for a StopOrder. The body can be used to pass to the Order endpoint.

ALLOWED_DT = ['DayOrder', 'GoodTillDate', 'GoodTillCancel']

__init__(*Uic*, *Amount*, *AssetType*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)
Instantiate a StopOrder.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade

- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount
- **OrderPrice** (*decimal (required)*) – the price indicating the limitprice
- **AssetType** (*string (required)*) – the assettype for the Uic
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **AmountType** (AmountType *(optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification
- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification
- **OrderDurationType** (*string, default DayOrder*) – the order duration type, check SAXO Bank specs. for details
- **GTDDate** (*datetime string (required if order duration is GoodTillDate)*) – the GTD-datetime

Example

```
>>> import json
>>> from saxo_openapi import API
>>> import saxo_openapi.endpoints.trading as tr
>>> from saxo_openapi.contrib.orders import StopOrder
>>>
>>> so = StopOrder(Uic=21,
...                AssetType=OD.AssetType.FxSpot,
...                Amount=10000,
...                OrderPrice=1.1025)
>>> print(json.dumps(so.data, indent=2))
{
  "Uic": 21,
  "AssetType": "FxSpot",
  "Amount": 10000,
  "Price": 1.1025,
  "BuySell": "Buy",
  "OrderType": "Stop",
  "ManualOrder": false,
  "AmountType": "Quantity",
  "OrderDuration": {
     "DurationType": "DayOrder"
  }
}
>>> # now we have the order specification, create the order request
>>> r = tr.orders.Order(data=so.data)
>>> # perform the request
>>> rv = client.request(r)
>>> print(rv)
>>> print(json.dumps(rv, indent=4))
```

(continues on next page)

(continued from previous page)

```
{
   "OrderId": "76697286"
}
```

data
data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.8 StopOrderFxSpot

class saxo_openapi.contrib.orders.stoporder.**StopOrderFxSpot**(*Uic*, *Amount*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

StopOrderFxSpot - StopOrder for FxSpot only.

The StopOrderFxSpot lacks the AssetType parameter and only serves the AssetType FxSpot.

ALLOWED_DT = ['DayOrder', 'GoodTillDate', 'GoodTillCancel']

__init__(*Uic*, *Amount*, *OrderPrice*, *ManualOrder=False*, *AmountType='Quantity'*, *TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*, *OrderDurationType='DayOrder'*, *GTDDate=None*)

Instantiate a StopOrderFxSpot.

Parameters

- **Uic** (*int (required)*) – the Uic of the instrument to trade
- **Amount** (*decimal (required)*) – the number of lots/shares/contracts or a monetary value if amountType is set to CashAmount
- **OrderPrice** (*decimal (required)*) – the price indicating the limitprice
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **AmountType** (AmountType *(optional)*) – the amountType, defaults to Quantity, see AmountType for other options
- **TakeProfitOnFill** (*TakeProfitDetails instance or dict*) – the takeprofit order specification
- **StopLosstOnFill** (*StopLossDetails instance or dict*) – the stoploss order specification

- **TrailingStopLosstOnFill** (*TrailingStopLossDetails instance or dict*) – the Trailingstoploss order specification
- **OrderDurationType** (*string, default DayOrder*) – the order duration type, check SAXO Bank specs. for details
- **GTDDate** (*datetime string (required if order duration is GoodTillDate)*) – the GTD-datetime

Example

```
>>> from saxo_openapi import API
>>> from saxo_openapi.contrib.orders import (
...           tie_account_to_order,
...           StopOrderFxSpot)
>>> token = "..."
>>> client = API(access_token=token)
>>> order = tie_account_to_order(
...     AccountKey,
...     StopOrderFxSpot(Uic=21, Amount=25000, OrderPrice=1.1025))
>>> r = tr.orders.Order(data=order)
>>> rv = client.request(r)
>>> print(json.dumps(rv, indent=2))
{
  "OrderId": "76703544"
}
```

data

data property.

return the JSON body.

hndOnFill(*TakeProfitOnFill=None*, *StopLossOnFill=None*, *TrailingStopLossOnFill=None*)

toJSON()

5.1.9 saxo_openapi.contrib.orders.onfill

The saxo_openapi.contrib.orders.onfill module offers some classes to simplify the creation of TakeProfit, StopLoss and TrailingStopLoss OCO-orders.

The next example shows how to create a marketorder for FX:

go LONG 10000 EURUSD place a take profit order after fill, to take profit @1.14 place a stoploss order after fill, to sell in case price drops below 1.12

```
>>> import json
>>> from saxo_openapi import API
>>> import saxo_openapi.endpoints.trading as tr
>>> import saxo_openapi.endpoints.portfolio as pf
>>> from saxo_openapi.contrib.orders import (
...     tie_account_to_order,
...     MarketOrderFxSpot,
...     TakeProfitDetails,
...     StopLossDetails)
>>> token = "..."
>>> client = API(access_token=token)
```

(continues on next page)

(continued from previous page)

```
>>> r = pf.accounts.AccountsMe()
>>> rv = client.request(r)
>>> # take the first account ...
>>> acct = rv['Data'][0]
>>> AccountKey = acct['AccountKey']
```

At time of writing EURUSD = 1.1250, lets take profit at 1.14, GoodTillCancel (default) lets take the loss when the price drops below at 1.12, GoodTillCancel (default)

TP-details can simply be constructed by using TakeProfitDetails:

```
>>> tpDetails = TakeProfitDetails(price=1.14)
>>> print(tpDetails.data)
{
  "OrderType": "Limit",
  "OrderDuration": {
      "DurationType": "GoodTillCancel"
  },
  "ManualOrder": False,
  "OrderPrice": "1.14000"
}
```

So, a MarketOrder with TP-details and SL-details:

```
>>> ordr = MarketOrderFxSpot(
...     Uic=21,
...     Amount=10000,
...     TakeProfitOnFill=TakeProfitDetails(price=1.14),
...     StopLosstOnFill=StopLosstDetails(price=1.12)
... )
>>> print(json.dumps(ordr.data, indent=4))
{
  "Uic": 21,
  "AssetType": "FxSpot",
  "Amount": 10000,
  "BuySell": "Buy",
  "OrderType": "Market",
  "ManualOrder": false,
  "AmountType": "Quantity",
  "OrderDuration": {
      DurationType": "FillOrKill"
  },
  "Orders": [
    {
      "OrderDuration": {
          DurationType": "GoodTillCancel"
      },
      "ManualOrder": false,
      "OrderType": "Limit",
      "OrderPrice": "1.14000",
      "BuySell": "Sell",
      "AssetType": "FxSpot",
      "Amount": 10000
    },
    {
      "OrderDuration": {
          DurationType": "GoodTillCancel"
```

(continues on next page)

(continued from previous page)

```
      },
      "ManualOrder": false,
      "OrderType": "Stop",
      "OrderPrice": "1.12000",
      "BuySell": "Sell",
      "AssetType": "FxSpot",
      "Amount": 10000
    }
  ]
}
```

Before an order can be placed the account needs to be attached to the orderbody also:

```
>>> ordr = tie_account_to_order(AccountKey=AccountKey,
...             MarketOrderFxSpot(
...                 Uic=21,
...                 Amount=10000,
...                 TakeProfitOnFill=TakeProfitDetails(price=1.14),
...                 StopLosstOnFill=StopLosstDetails(price=1.12)
...             ))
>>> print(json.dumps(ordr, indent=4))
{
  "Uic": 21,
  "AssetType": "FxSpot",
  "Amount": 10000,
  "BuySell": "Buy",
  "OrderType": "Market",
  "AmountType": "Quantity",
  "ManualOrder": false,
  "OrderDuration": {
      DurationType": "FillOrKill"
  },
  "Orders": [
    {
      "OrderDuration": {
          DurationType": "GoodTillCancel"
      },
      "ManualOrder": false,
      "OrderType": "Limit",
      "OrderPrice": "1.14000",
      "BuySell": "Sell",
      "AssetType": "FxSpot",
      "Amount": 10000,
      "AccountKey": "fOA0tvOyQqW2aHpWi9P5bw=="
    },
    {
      "OrderDuration": {
          DurationType": "GoodTillCancel"
      },
      "ManualOrder": false,
      "OrderType": "Stop",
      "OrderPrice": "1.12000",
      "BuySell": "Sell",
      "AssetType": "FxSpot",
      "Amount": 10000,
      "AccountKey": "fOA0tvOyQqW2aHpWi9P5bw=="
    }
```

(continues on next page)

(continued from previous page)

```
  ],
  "AccountKey": "fOA0tvOyQqW2aHpWi9P5bw=="
}
```

Now the order can be placed:

```
>>> r = tr.orders.Order(data=ordr)
>>> rv = client.request(r)
{
  "OrderId": "76697286",
  "Orders": [
    {
      "OrderId": "76697287"
    },
    {
      "OrderId": "76697288"
    }
  ]
}
```

StopLossDetails

class saxo_openapi.contrib.orders.onfill.**StopLossDetails**(*price*, *ManualOrder=False*, *OrderDurationType='GoodTillCancel'*, *GTDDate=None*)

Representation of the specification for a StopLossOrder.

It is typically used to specify 'stop loss details' for the 'StopLossOnFill' parameter of an OrderRequest. From the details a Stop order will be created with the specified *price*. The order gets placed when the underlying order gets filled.

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

ALLOWED_DT = ['GoodTillCancel', 'GoodTillDate', 'DayOrder']

__init__(*price*, *ManualOrder=False*, *OrderDurationType='GoodTillCancel'*, *GTDDate=None*)

Instantiate StopLossDetails.

Parameters

- **price** (*float or string (required)*) – the price to trigger take profit order
- **OrderDurationType** (OrderDurationType *(required)*) – the duration, default is: OrderDurationType.GoodTillCancel
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **GTDDate** (*string or datetim (optional)*) – GTD-datetime is required if OrderDurationType.GoodTillDate

data

data - return the JSON body.

The data property returns a dict representing the JSON-body needed for the API-request. All values that are not set will be left out

toJSON()

TakeProfitDetails

class saxo_openapi.contrib.orders.onfill.**TakeProfitDetails**(*price*, *ManualOrder=False*, *OrderDurationType='GoodTillCancel'*, *GTDDate=None*)

Representation of the specification for a TakeProfitOrder.

It is typically used to specify 'take profit details' for the 'TakeProfitOnFill' parameter of an OrderRequest. From the details a Limit order will be created with the specified *price*. The order gets placed when the underlying order gets filled.

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

ALLOWED_DT = ['GoodTillCancel', 'GoodTillDate', 'DayOrder']

__init__(*price*, *ManualOrder=False*, *OrderDurationType='GoodTillCancel'*, *GTDDate=None*)

Instantiate TakeProfitDetails.

Parameters

- **price** (*float or string (required)*) – the price to trigger take profit order
- **OrderDurationType** (OrderDurationType *(required)*) – the duration, default is: OrderDurationType.GoodTillCancel
- **ManualOrder** (*bool (required)*) – flag to identify if an order is from an automated origin, default: False
- **GTDDate** (*string or datetime (optional)*) – GTD-datetime is required if OrderDurationType.GoodTillDate

data

data - return the JSON body.

The data property returns a dict representing the JSON-body needed for the API-request. All values that are not set will be left out

toJSON()

5.2 saxo_openapi.contrib.util

5.2.1 InstrumentToUic

saxo_openapi.contrib.util.**InstrumentToUic**(*client*, *AccountKey*, *spec*, *assettype='FxSpot'*)

replace the Instrument for a Uic in the spec dict. If there is no Instrument in spec the spec gets returned untouched.

In case there are multiple entries returned and ValueError is raised.

Parameters

- **client** (*API client instance (required)*) – the API client instance
- **AccountKey** (*string (required)*) – the AccountKey of the account

- **spec** (*dict (required)*) – the dictionary to process. If it contains an 'Instrument' key, try to replace it by a Uic
- **assettype** (*string (required: default 'FxSpot')*) – the assettype used in the query with the Instrument

Example

```
>>> from saxo_openapi import API
>>> from saxo_openapi.contrib.util import InstrumentToUic
>>> from pprint import pprint
>>> token = "..."
>>> AccountKey = "..."
>>> client = API(access_token=token)
>>> spec = {'Instrument': 'EURUSD', 'Amount': 120000}
>>> # find the Uic for Instrument
>>> pprint(InstrumentToUic(client, AccountKey, spec=spec))
{'Amount': 120000, 'Uic': 21}
```

CHAPTER 6

Examples

6.1 Stream processing

The SAXO OpenAPI offers steaming support using *websockets*.

Several endpoints offer streaming capabilities by creating *subscriptions*. These subscriptions create messages that can be identified by the *referenceId*.

```
#!/usr/bin/env python3.6

"""Simple stream processing."""


import asyncio
import websockets
from saxo_openapi.contrib.ws import stream


async def echo(ContextId, token):
    hdrs = {
        "Authorization": "Bearer {}".format(token),
    }
    URL = "wss://streaming.saxotrader.com/sim/openapi/streamingws/connect?" + \
          "contextId={ContextId}".format(ContextId=ContextId)
    async with websockets.connect(URL, extra_headers=hdrs) as websocket:
        async for message in websocket:
            print(stream.decode_ws_msg(message))

if __name__ == "__main__":
    import sys

    with open("token.txt") as I:
        token = I.read().strip()
```

(continues on next page)

(continued from previous page)

```
    asyncio.get_event_loop().run_until_complete(Echo(ContextId=sys.argv[1],
                                                     token=token))
```

This code will read messages from a message stream. It must be started with a *ContextId*.

```
$ python stream_example.py ctxt_20190311
```

Now the program runs, but no output appears. By using the explorer www.developer.saxo it is possible to create one ore more subscriptions.

To create a subscription for price information:

- click the *Trading* service group
- click *Prices*
- click POST /trade/v1/prices/subscriptions

```
{
    "Arguments": {
        "Uic": 21,
        "AssetType": "FxSpot"
    },
    "ContextId": "ctxt_20190311",
    "ReferenceId": "EUR_USD"
}
```

Click SEND to submit the POST. Quotes for EUR_USD will show up in the stream. Additional subscriptions can be added and the messages will appear in the stream by their *referenceId*. The output shows EUR_USD price information and output of a subscription for *account balance*.

```
{'refid': b'acctbal', 'msgId': 5004, 'msg': {'InitialMargin': {
↪'MarginAvailable': 98669.15, 'NetEquityForMargin': 100169.15},
↪'MarginAvailableForTrading': 98669.15, 'MarginNetExposure': 100000.32,
↪'NetEquityForMargin': 100169.15, 'TotalValue': 100169.15,
↪'UnrealizedMarginOpenProfitLoss': 179.14, 'UnrealizedMarginProfitLoss':␣
↪179.14, 'UnrealizedPositionsValue': 174.14}}
{'refid': b'EUR_USD', 'msgId': 5005, 'msg': {'LastUpdated': '2019-03-
↪11T19:47:54.197000Z', 'Quote': {'Ask': 1.12491, 'Bid': 1.12471, 'Mid': 1.
↪12481}}}
{'refid': b'acctbal', 'msgId': 5006, 'msg': {'MarginNetExposure': 100000.88}}
{'refid': b'acctbal', 'msgId': 5007, 'msg': {'MarginNetExposure': 100001.44}}
{'refid': b'acctbal', 'msgId': 5008, 'msg': {'MarginNetExposure': 100002.0}}
{'refid': b'EUR_USD', 'msgId': 5009, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:04.830000Z'}}
{'refid': b'acctbal', 'msgId': 5010, 'msg': {'MarginNetExposure': 100001.44}}
{'refid': b'acctbal', 'msgId': 5011, 'msg': {'InitialMargin': {
↪'MarginAvailable': 98666.93, 'NetEquityForMargin': 100166.93},
↪'MarginAvailableForTrading': 98666.93, 'MarginNetExposure': 100000,
↪'NetEquityForMargin': 100166.93, 'TotalValue': 100166.93,
↪'UnrealizedMarginOpenProfitLoss': 176.92, 'UnrealizedMarginProfitLoss':␣
↪176.92, 'UnrealizedPositionsValue': 171.92}}
{'refid': b'EUR_USD', 'msgId': 5012, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:10.464000Z', 'Quote': {'Ask': 1.1249, 'Bid': 1.1247, 'Mid': 1.
↪1248}}}
{'refid': b'EUR_USD', 'msgId': 5013, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:12.482000Z', 'Quote': {'Ask': 1.12487, 'Bid': 1.12467, 'Mid': 1.
↪12477}}}
```

(continues on next page)

(continued from previous page)

```
{'refid': b'acctbal', 'msgId': 5014, 'msg': {'InitialMargin': {
↪'MarginAvailable': 98664.71, 'NetEquityForMargin': 100164.71},
↪'MarginAvailableForTrading': 98664.71, 'MarginExposureCoveragePct': 100.16,
↪ 'MarginNetExposure': 100000.93, 'NetEquityForMargin': 100164.71,
↪'TotalValue': 100164.71, 'UnrealizedMarginOpenProfitLoss': 174.7,
↪'UnrealizedMarginProfitLoss': 174.7, 'UnrealizedPositionsValue': 169.7}}
{'refid': b'EUR_USD', 'msgId': 5015, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:15.882000Z'}}
{'refid': b'acctbal', 'msgId': 5016, 'msg': {'InitialMargin': {
↪'MarginAvailable': 98666.93, 'NetEquityForMargin': 100166.93},
↪'MarginAvailableForTrading': 98666.93, 'MarginNetExposure': 100002.59,
↪'NetEquityForMargin': 100166.93, 'TotalValue': 100166.93,
↪'UnrealizedMarginOpenProfitLoss': 176.92, 'UnrealizedMarginProfitLoss':␣
↪176.92, 'UnrealizedPositionsValue': 171.92}}
{'refid': b'EUR_USD', 'msgId': 5017, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:23.501000Z', 'Quote': {'Ask': 1.12486, 'Bid': 1.12466, 'Mid': 1.
↪12476}}}
{'refid': b'acctbal', 'msgId': 5018, 'msg': {'InitialMargin': {
↪'MarginAvailable': 98666.94, 'NetEquityForMargin': 100166.94},
↪'MarginAvailableForTrading': 98666.94, 'MarginNetExposure': 100003.16,
↪'NetEquityForMargin': 100166.94, 'TotalValue': 100166.94,
↪'UnrealizedMarginOpenProfitLoss': 176.93, 'UnrealizedMarginProfitLoss':␣
↪176.93, 'UnrealizedPositionsValue': 171.93}}
{'refid': b'EUR_USD', 'msgId': 5019, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:24.604000Z', 'Quote': {'Ask': 1.12487, 'Bid': 1.12467, 'Mid': 1.
↪12477}}}
{'refid': b'EUR_USD', 'msgId': 5020, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:26.716000Z'}}
{'refid': b'acctbal', 'msgId': 5021, 'msg': {'MarginNetExposure': 100003.72}}
{'refid': b'EUR_USD', 'msgId': 5022, 'msg': {'LastUpdated': '2019-03-
↪11T19:48:27.088000Z', 'Quote': {'Ask': 1.12486, 'Bid': 1.12466, 'Mid': 1.
↪12476}}}
{'refid': b'acctbal', 'msgId': 5023, 'msg': {'MarginNetExposure': 100003.16}}
```

6.1.1 Subscriptions using saxo_openapi

Creating price-subscriptions using the *saxo_openapi* is easy too.

```
#!/usr/bin/env python3.6

"""Simple demo program that looks up the Uic for currencypairs entered by name.
For each pair it creates a subscription for price information with the instrumentname
as  Referenceid.

The program asumes you have a file with the token locally in token.tok.

Usage: price_subscr.py <contextid> EURUSD EURJPY EURGBP
"""
from saxo_openapi import API
import saxo_openapi.endpoints.trading as tr
import saxo_openapi.endpoints.referencedata as rd
import saxo_openapi.contrib.session as session
import json

def subscribe_for_prices(client, ContextId, instruments):
```

(continues on next page)

(continued from previous page)

```
    """fetch instrument data by the name of the instrument and extract the Uic␣
↪(Identifier)
    and use that to subscribe for prices.
    Use the name of the instrument as reference.
    """
    _ai = session.account_info(client=client)

    # body template for price subscription
    body = {
       "Arguments": {
           "Uic": "",
           "AssetType": "FxSpot"
       },
       "ContextId": "",
       "ReferenceId": ""
    }
    body.update({'ContextId': ContextId})

    for instrument in instruments:
        params = {'AccountKey': _ai.AccountKey,
                  'AssetTypes': 'FxSpot',
                  'Keywords': instrument
                 }
        # create the request to fetch Instrument info
        r = rd.instruments.Instruments(params=params)
        rv = client.request(r)
        if len(rv['Data']) == 1:
            body['Arguments'].update({'Uic': rv['Data'][0]['Identifier']})
            body.update({"ReferenceId": instrument})
            # print("Prepping: ")
            # print(json.dumps(body, indent=2))
            # create the request to fetch Instrument info
            r = tr.prices.CreatePriceSubscription(data=body)
            client.request(r)

            status = "succesful" if r.status_code == r.expected_status else "failed"
            print("Subscription for instrument: {} {}".format(instrument, status))

        else:
            print("Got multiple instruments for {}, can't choose...skip".
↪format(instrument))


if __name__ == "__main__":

    import sys
    with open("token.txt") as I:
        token = I.read().strip()
        client = API(access_token=token)
        ContextId = sys.argv[1]
        subscribe_for_prices(client, ContextId, sys.argv[2:])
        print("check the stream for data ...")
```

Now create the price subscriptions with the program above:

```
$ python price_subscr.py ctxt_20190311 EURJPY EURGBP
Subscription for instrument: EURJPY succesful
```

(continues on next page)

(continued from previous page)

```
Subscription for instrument: EURGBP succesful
check the stream for data ...
```

The new instruments will show up in the stream output.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

S

T

U

Y