polygon

Release 1.0.8

P S Solanki

Jul 11, 2022

CONTENTS:


GitHub


patreon

CHAPTER

ONE

GETTING STARTED

Welcome to `polygon`. Read this page to quickly install and configure this library to write your first Polygon Python application.

It is highly recommended to read this page for everyone since it contains everything you need to get started with the library

You can see some examples on the github repository after you have done the initial steps. And maybe join our Discord Server while you're at it :D

1.1 What you need to have

1. A polygon.io account and your `API key`. Find your api key on Your Dashboard
2. Python version 3.6 or higher. Don't have it installed? Install python

Once you have these, Proceed to the installation of the library. Skip if already done.

1.2 Installing `polygon`

The recommended method of installation for all users is to install using `pip` from PyPi. A virtual environment is highly recommended but not a necessity.

run the below command in terminal (same for all OS)

```
pip install polygon
```

To confirm the install worked, try importing the package as such

```
import polygon
```

If this doesn't throw any errors, the install worked. You may proceed to next steps now.

1.2.1 Optional Libraries

You can also install the library with **optional dependencies** (you can skip them if you don't need their functionalities)

```
pip install uvloop  # this will install uvloop, see the uvloop section below to know how␣
↪to use uvloop for faster performance on pure async programs

# OR

pip install orjson  # this will install orjson lib. Polygon lib would use orjson if␣
↪available for streaming clients only. This enables fast json decoding of responses

# OR to get both

pip install orjson, uvloop  # Note that uvloop is only available on Unix platforms as of␣
↪now
```

1.3 General guide for clients & functions

This section would provide general guidance on the clients without going into specific endpoints as stocks or options.

As you already know polygon.io has two major classes of APIs. The `REST` APIs and `websockets` streaming APIs.

This library implements all of them.

- For REST HTTP endpoints
  - Regular client is implemented for all endpoints.
  - Support for `async` client is also provided. See *Async Support for REST endpoints* for more.
- For websocket streaming endpoints
  - a `callback` based stream client is implemented. See *Callback Streaming*
  - an async based stream client is also implemented. See *Async Streaming*

Be sure to check out our special section *What the Hell are Enums Anyways* for info on `enums` which will be used in many functions in this library to avoid passing error prone data.

Functions which are standalone (not attached to client objects) are all available at the top level as `polygon.function_name`.

A detailed description of how to use the streaming endpoints is provided in the streamer docs linked above.

Need examples? The github repository has a few you could use.

also feel free to join in our Discord Server **to ask a question or just chat with interesting people**

1.3.1 Creating and Using REST HTTP clients

This section aims to outline the general procedure to create and use the http clients in both regular and async programming methods.

First up, you'd import the library. There are many ways to import names from a library and it is highly recommended to complete fundamental python if you're not aware of them.

```
import polygon

# OR import the name you need
from polygon import StocksClient

# OR import the names you need
from polygon import (StocksClient, ForexClient, StreamClient, build_option_symbol)
```

Now creating a client is as simple as (using stocks and forex clients as examples here)

1. Regular client: `stocks_client = polygon.StocksClient('API_KEY')`
2. Async client: `forex_client = polygon.ForexClient('API_KEY', True)`

Note that It is NOT recommended to hard code your API key or other credentials into your code unless you really have a use case. Instead preferably do one of the following:

1. create a separate python file with credentials, import that file into main file and reference using variable names.
2. Use environment variables.

1.3.2 Request timeouts and limits configuration (optional)

section Only meant for advanced use cases. For most people, default timeouts would be enough.

You can also specify timeouts on requests. By default the timeout is set to 10 seconds for connection, read, write and pool timeouts.

write timeout and pool timeout are only available for async rest client (which is httpx based). They'll be ignored if used with normal client

If you're unsure of what this implies, you probably don't need to change them.

Limits config

Only meant for async rest client (httpx based).

You also have the ability to change httpx connection pool settings when you work with async based rest client. This allows you to better control the behavior of underlying httpx pool, especially in cases where you need highly concurrent async applications. Using uvloop is also a good option in those case

You can change the below configs:

- **max_connections**: the max number of connections in the pool. Defaults to No Limit in the lib.
- **max_keepalive**: max number of keepalive connections in the pool. Defaults to 30.

Example uses:

```
# client with a custom timeout. Default is 10 seconds
client = polygon.StocksClient('api_key', connect_timeout=15)

# another one
```

(continues on next page)

(continued from previous page)

```
client = polygon.StocksClient('api_key', connect_timeout=5, read_timeout=5)

# An async one now
client = polygon.StocksClient('key', True, read_timeout=5, connect_timeout=15)

# another async one
client = polygon.StocksClient('key', True, connect_timeout=15, max_connections=200)
```

Now that you have a client, simply call its methods to get data from the API

```
current_price = stocks_client.get_current_price('AMD')
print(f'Current price for AMD is {current_price}')
```

Note that you can have instances of all 5 different types of http clients together. So you can create client for each one of the stocks, options and other APIs

All the clients in the lib support context managers

```
with polygon.StocksClient('KEY') as client:
    last_quote = client.get_last_quote('AMD)
    print(f'Last quote for AMD: {last_quote}')

# OR for async
async with polygon.StocksClient('key', True) as client:
    last_quote = await client.get_last_quote('AMD')
    print(last_quote)
```

Using context managers ensures that the connections opened up to make requests are closed properly.

You can manually close the connections if you're not using context managers:

1. for regular non-async: `client.close()`
2. for async: `await client.close()`

This is not an absolute necessity but rather a good software practice to close out resources when you don't need them.

1.4 Calling the methods/functions

Most methods and functions have sane default values which can be customized as needed. Required parameters need to be supplied as positional arguments (which just means that the order of arguments matter when passing more than one).

Some options, crypto and forex endpoints expect you to append prefixes `O:`, `C:`, `X:` respectively in front of tickers (on options symbols, forex pairs and crypto pairs). **the library handles this for you** so you can pass in those with or without the prefix.

Parameters which have special values are supplied as python enums. You can however always pass in your own values but it is recommended to use enums as they mitigate the possibilities of an error.

All enums are available in the module `polygon.enums` and can be imported the way you like.

If you're still unsure about enums, see our dedicated section: *What the Hell are Enums Anyways*

1.4.1 Passing dates, datetime values or timestamps

The library allows you to specify your datetime or date values as `datetime.date`, `datetime.datetime` objects or as string `YYYY-MM-DD`. Some endpoints also accept millisecond/nanosecond timestamps (docs will mention this wherever necessary)

- If an endpoint accepts a timestamp, you can either pass a timestamp or a datetime or date object. The lib will do the conversions for you internally
- When you pass a timestamp, library will NOT do any conversions and pass it as is. So make sure you are passing the correct timestamps.
- If you pass a `datetime` object, and the endpoint accepts a timestamp, the lib will convert internally to a timestamp. If there is no timezone info attached to the object, `UTC` will be used.
- If you come across situations where the returned data results are not complete or missing some values (for eg on aggregate bars endpoint), just pass your values as `datetime` values (if possible as a timestamp or with timezone information at least)
- The lib makes its best efforts parsing what the supplied datetime/timestamp/date could mean in context of the relevant endpoint. The behavior is of course different between for example aggs and trades. If you want absolute control, just pass as a unix timestamp or a `datetime` object having timezone information

Here are some **best practices when passing datetime or dates or timestamps**

- If you want complete control over what's passed, pass a timestamp since epoch. The accuracy (i.e milli second or nano second) depends on the endpoint itself (mentioned in the docs of course). Default timestamp accuracy is `ms`
- Passing `datetime` objects is also a good way to pass absolute values and is recommended. Even better if the object has timezone info. If no timezone info is provided, lib assumes UTC. It doesn't make a difference in most cases, but should be taken care of in fine tuning and accurate filtering scenarios

1.5 Return Values

Most methods would by default return a dictionary/list object containing the data from the API. If you need the underlying response object you need to pass in `raw_response=True` in the function call. It might be useful for checking `status_code` or inspecting `headers`.

For 99% users, the default should be good enough.

The underlying response object returned is `requests.models.Response` for regular client and `httpx.Response` for async client. Using `.json()` on the response object gets you the data dict/list

Once you have the response, you can utilize the data in any way that you like. You can push it to a database, create a pandas dataframe, save it to a file or process it the way you like.

Every method's documentation contains a direct link to the corresponding official documentation page where you can see what the keys in the response mean.

1.6 Pagination Support

So quite a few endpoints implement pagination for large responses and hence the library implements a very simple and convenient way to get all the pages and merge responses internally to give you a single response with all the results in it.

The behavior is exactly the same for ALL endpoints which support pagination (docs will mention when an endpoint is paginated). Knowing the functions and parameters once is enough for all endpoints.

To enable pagination

> you simply need to pass `all_pages=True` to enable pagination for the concerned endpoint. You can also pass `max_pages=an integer` to limit how many pages the lib will fetch internally. The default behavior is to fetch all available pages.

You can pass `verbose=True` if you want to know what's happening behind the scenes. It will print out status messages about the pagination process.

You can further customize what kinda output you want to get. **you have three possible options to make use of pagination abilities** in the library

1.6.1 Get a Single Merged Response (recommended)

Recommended for most users. Using this method will give you all the pages, **merged into one single response** internally for your convenience, and you will get all the results from all pages in one single list.

To use, simply pass `all_pages=True`. you can optionally provide `max_pages` number too to limit how many pages to get.

for example, below examples will do the merging of responses internally for you

```
# assuming client is created already

# This will pull ALL available tickers from reference APIs and merge them into a single␣
↪list
data = client.get_tickers(market='stocks', limit=1000, all_pages=True)

# This will pull up to 4 available pages of tickers from reference APIs and merge them␣
↪into a
# single list
data = client.get_tickers(market='stocks', limit=1000, all_pages=True, max_pages=5)
```

1.6.2 Get a List of all pages

Only for people who know they need it. what this method does is provide you with a list of all pages, WITHOUT merging them. so you'll basically get a list of all pages like so `[page1_data, page2_data, page3_data]`.

By default each page element is the corresponding page's data itself. You can also customize it to get the underlying response objects (meant for advanced use cases)

To enable, as usual you'd pass in `all_pages=True`. But this time you'll ask the lib not to merge the pages using `merge_all_pages=False`. That's it. as described above, to get underlying response objects, pass an additional `raw_page_responses=True` too.

See examples below

```
# assuming client is created already

# will fetch all available pages, won't merge them and return a list of responses
data = client.get_tickers(market='stocks', limit=1000, all_pages=True, merge_all_
↪pages=False)

# will fetch all available pages, won't merge them and return a list of response objects
data = client.get_tickers(market='stocks', limit=1000, all_pages=True, merge_all_
↪pages=False,
                          raw_page_responses=True)

# will fetch up to 5 available pages, won't merge them and return a list of responses
data = client.get_tickers(market='stocks', limit=1000, all_pages=True, merge_all_
↪pages=False,
                          max_pages=5)
```

1.6.3 Paginate Manually

Only meant for people who really need more manual control over pagination, yet want to make use of available functionality.

Every client has a few core methods which can be used to get next or previous pages by passing in the last response you have.

Note that while using these methods, you'd need to use your own mechanism to combine pages or process them. If any of these methods return False, it means no more pages are available.

Examples Use

```
# assuming a client is created already
data = client.get_trades(<blah-blah>)

next_page_of_data = client.get_next_page(data)  # getting NEXT page
previous_page_of_data = client.get_previous_page(data)  # getting PREVIOUS page

# ASYNC examples
await client.get_next_page(data)
await client.get_previous_page(data)

# It's wise to check if the value returned is not False.
```

In practice, to get all pages (either next or previous), you'll need a while loop An example:

```
all_responses = []

response = client.get_trades_vx(<blah-blah>)  # using get_trades as example. you can use␣
↪it on all methods which support pagination
all_responses.append(response)  # using a list to store all the pages of response. You␣
↪can use your own approach here.

while 1:
    response = client.get_next_page(response)  # change to get_previous_page for␣
↪previous pages.
```

(continues on next page)

(continued from previous page)

```
    if not response:
        break

    all_responses.append(response)  # adding further responses to our list. you can use␣
↪your own approach.

print(f'all pages received. total pages: {len(all_responses)}')
```

1.7 Better Aggregate Bars function

This is a new method added to the library, making it easy to get historical price candles (OCHLV) with ease. The lib does most of the heavy lifting internally, and provides you with a single list which would have ALL the candles.

The functionality is available on both sync (normal) client and also on asyncio based client.

WHY though??

so the aggregate bars endpoints have a weird thing where they don't have any pagination and the number of maximum candles in one response to 50k only. Now usually this is fine if you only seek minute candles for a month for example. But what if you need historical prices for last 10 years?

The library attempts to solve that challenge for you. Depending on whether you tell it to run in parallel or sequentially (info on how to customize behavior is below), the function will grab ALL the responses in the **date range you specify**, will drop duplicates, will drop candles which do not fall under the original time range specified by you. merge the response, return a single list with all the data in there.

For most people, the default values should be enough, but for the ones who hate themselves (:P), it is possible to customize the behavior however they like.

Note that the methods/functions are the same for all aggregate clients (stocks, options, forex and crypto). Knowing it once is enough for all other clients

1.7.1 How the Hell do I use it then

- First things first, the argument to supply to enable the new aggs functionality is passing `full_range=True` to your `client.get_aggregate_bars()` call.

  for example: `stocks_client.get_aggregate_bars('AMD', '2005-06-28', '2021-03-08', full_range=True)`

- The above example will split the larger timeframe into smaller ones, and request them in parallel using a ThreadPool (sync client) or a set of coroutines (async client)
- If you don't want it to run in parallel (recommended to run parallel though), you can just specify `run_parallel=False`. doing that will make the library request data one by one, using the last response received as the new start point until end date is reached. This might be useful if you're running a thread pool of your own and don't want the internal thread pool to mess with your own thread pool. **on async client, always prefer to run parallel**
- The parallel versions (on both threaded and async clients) always split the larger range into smaller ones (45 days for minute frequency, 60 days for hour frequency, close to 10 years for others). If you find yourself dealing with a very highly volatile symbol (eg spy or some crypto symbols which are traded for a high timespan) and the 50k limit is causing some data to be stripped off, you can add the additional argument `high_volatility=True`. This will make the library further reduce its time chunk size

- By default it will also print some warnings if they occur. You can turn off those warnings using `warnings=False`. Only do it if necessary though.
- When working with the parallel versions, you also have the ability to specify how many concurrent threads/coroutines you wish to spawn using `max_concurrent_workers=a new number` ONLY change it if you know you need it. This can sometimes help reduce loads or gain performance boost depending on whether it's increased or decreased. The default is `your cpu core count * 5`
- By default, the results returned will be in ascending order (oldest candles first in the final list). To change that simply specify descending order . You can either pass the enum *polygon.enums.SortOrder* (recommended) or pass a string `sort='desc'`.

1.7.2 I want to do it manually, but could use some help

Oh sure, You can also do that. the function which actually splits large timeframes to smaller ones, can be used to get a list of smaller timeframes with their own start and end times.

Then you can iterate over the list and make requests yourself. Don't do that unless you have to though. It's always better to use built in lib functions

anyways, the function you want to call is `split_date_range()`. You can call this method like so:

```
import polygon

client = polygon.StocksClient('KEY')

time_frames = client.split_date_range(start_date, end_date, timespan='minute')
```

This method also accepts a few more arguments described below:

`Base.`**`split_date_range`**(*start*, *end*, *timespan: str*, *high_volatility: bool = False*, *reverse: bool = True*) → list

Internal helper function to split a BIGGER date range into smaller chunks to be able to easily fetch aggregate bars data. The chunks duration is supposed to be different for time spans. For 1 minute bars, multiplier would be 1, timespan would be 'minute'

Parameters

- **`start`** – start of the time frame. accepts date, datetime objects or a string `YYYY-MM-DD`
- **`end`** – end of the time frame. accepts date, datetime objects or a string `YYYY-MM-DD`
- **`timespan`** – The frequency type. like day or minute. see *polygon.enums.Timespan* for choices
- **`high_volatility`** – Specifies whether the symbol/security in question is highly volatile. If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **`reverse`** – If True (the default), will reverse the order of chunks (chronologically)

Returns

a list of tuples. each tuple is in format `(start, end)` and represents one chunk of time frame

so basically

- By default the list returned will have newer timeframes first. To change that just pass `reverse=False`
- if the symbol you are dealing with is very volatile, so much that the 50k limit per response might be low, you can pass `high_volatility=True` and lib will return timeframe in smaller chunks. (for eg, on minute aggs, 45 day chunks are default, for high volatile symbols it will become 30 days)

1.8 Async Support for REST endpoints

As you saw above in the example, the clients have methods for each endpoint. The usual client is a sync client. However support for async is also provided for all the endpoints on all the clients.

Here is how to make use of it (**This info is applicable to ALL rest clients**)

First up, you'd create a client. Earlier you created a client by passing in just your API key. Here you'd create the client with an additional argument.

so instead of something like: `StocksClient('API_KEY')`, you'd do

```
client = StocksClient('KEY', True)   # or use_async=True for second parameter
```

This gives you an async client. Similar to sync, you can have all 5 different clients together. You can also pass in your timeout values like you did above here too.

ALL the methods you'd use for async client have the same names as their sync counterpart names.

So if a method is named `get_trades()` in usual client, in async client you'd have it as `get_trades()` as well and this behavior is true for all methods

Here is how you can use it grab the current price of a symbol

```
import polygon

async def main():
    stocks_client = polygon.StocksClient('API_KEY', True)

    current_price = await stocks_client.get_current_price('AMD')
    print(current_price)

if __name__ == '__main__':
    import asyncio
    asyncio.run(main())
```

1.9 UVLOOP integration

(for async streamer and async rest client)

unix based Operating systems only, uvloop doesn't have windows support yet

If your use case demands better performance on async streamer or async based applications using rest client than what the usual `asyncio` has to offer, consider using uvloop, a `libuv` based event loop which provides faster execution.

Using it is very simple, install using `pip install uvloop` and then **at the very top of your program**, right below your imports, add:

```
import uvloop

asyncio.set_event_loop_policy(uvloop.EventLoopPolicy())
```

That's it. asyncio will now use uvloop's event loop policy instead of the default one.

1.10 Special Points

- Any method/endpoint having `vX` in its name is deemed experimental by polygon and its name and underlying URL path will be changed to a version number in the future. If you do use one of these, be aware of that name change which is reflected in the docs. If you find the lib doesn't have the changes reflected, let me know through any means mentioned in the help page.
- You would notice some parameters having `lt`, `lte`, `gt` and `gte` in their names. Those parameters are supposed to be filters for `less than`, `less than or equal to`, `greater than`, `greater than or equal to` respectively. To know more see heading **Query Filter Extensions** in This blog post by polygon To explain: imagine a parameter: `fill_date_lt`. now the date you'll supply would be a filter for values less than the given value and hence you'd get results which have fill_date less than your specified value, which in this case is a date.
- Some endpoints may not return a dictionary and instead return a `list`. The number of such endpoints is very low. Similarly get current price returns a float/integer. I'm working towards reflecting the same in individual method's docs.
- It is highly recommended to use the polygon.io documentation website's quick test functionality to play around with the endpoints.
- Type hinting in function/method definitions indicate what data type does that parameter is supposed to be. If you think the type hinting is incomplete/incorrect, let me know. For example you might ses: `cost: int` which means this parameter `cost` is supposed to be an integer. `adjusted: bool` is another example for a boolean (either `True` or `False`)
- You'll notice some type hints having `Union` in them followed by two or more types inside a square bracket. That simply means the parameter could be of any type from that list in bracket . For example: `price: Union[str, float, int]` means the parameter `price` could be either a string, a float or an integer. You'd notice Union type hints more on return types of the functions/methods.

so far so good? Start by taking a look at the complete docs for endpoints you need. Here is a quick list

- *Stocks*
- *Options*
- *Forex* and *Crypto*
- *Callback Streaming* and *Async Streaming*
- *What the Hell are Enums Anyways*

CHAPTER

TWO

STOCKS

So you have completed the initial steps and are ready to dive deep into endpoints. Read this page to know everything you need to know about using the various Stocks HTTP endpoints.

See *Async Support for REST endpoints* for asynchronous use cases.

Docs below assume you have already read getting started page and know how to create the client. If you do not know how to create the client, first see *General guide for clients & functions* and create client as below. As always you can have all 5 different clients together.

```
import polygon

stocks_client = polygon.StocksClient('KEY')  # for usual sync client
async_stock_client = polygon.StocksClient('KEY', True)  # for an async client
```

here is how the client initializer looks like:

polygon.stocks.stocks.**StocksClient**(*api_key: str*, *use_async: bool = False*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all REST Stocks endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **use_async** – Set it to `True` to get async client. Defaults to usual non-async client.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.

- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

Endpoints

To use any of the below method, simply call it on the client you created above. so if you named your client `client`, you'd call the methods as `client.get_trades` and so on. Async methods will need to be awaited, see *Async Support for REST endpoints*.

2.1 Get Trades

`SyncStocksClient.get_trades`(*symbol: str*, *date*, *timestamp: Optional[int] = None*, *timestamp_limit: Optional[int] = None*, *reverse: bool = True*, *limit: int = 5000*, *raw_response: bool = False*)

Get trades for a given ticker symbol on a specified date. The response from polygon seems to have a `map` attribute which gives a mapping of attribute names to readable values. Official Docs

Parameters

- **symbol** – The ticker symbol we want trades for.
- **date** – The date/day of the trades to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **timestamp** – The timestamp offset, used for pagination. Timestamp is the offset at which to start the results. Using the `timestamp` of the last result as the offset will give you the next page of results. Default: None. I'm trying to think of a good way to implement pagination support for this type of pagination.
- **timestamp_limit** – The maximum timestamp allowed in the results. Default: None
- **reverse** – Reverse the order of the results. Default True: oldest first. Make it False for Newest first
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.2 Get Trades v3

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncStocksClient.get_trades_v3`(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for a ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want trades for.
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.StocksTradesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

2.3 Get Quotes

SyncStocksClient.**get_quotes**(*symbol: str*, *date*, *timestamp: Optional[int] = None*, *timestamp_limit: Optional[int] = None*, *reverse: bool = True*, *limit: int = 5000*, *raw_response: bool = False*)

Get Quotes for a given ticker symbol on a specified date. The response from polygon seems to have a `map` attribute which gives a mapping of attribute names to readable values. Official Docs

Parameters

- **symbol** – The ticker symbol we want quotes for.
- **date** – The date/day of the quotes to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **timestamp** – The timestamp offset, used for pagination. Timestamp is the offset at which to start the results. Using the `timestamp` of the last result as the offset will give you the next page of results. Default: None. Thinking of a good way to implement this pagination here.
- **timestamp_limit** – The maximum timestamp allowed in the results. Default: None
- **reverse** – Reverse the order of the results. Default True: oldest first. Make it False for Newest first
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.4 Get Quotes v3

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

SyncStocksClient.**get_quotes_v3**(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get NBBO Quotes for a ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want quotes for.
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.StocksQuotesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.

- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

2.5 Get Last Trade

`SyncStocksClient.`**`get_last_trade`**(*symbol: str*, *raw_response: bool = False*)

Get the most recent trade for a given stock. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.6 Get last Quote

SyncStocksClient.**get_last_quote**(*symbol: str*, *raw_response: bool = False*)

Get the most recent NBBO (Quote) tick for a given stock. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

2.7 Get Daily Open Close

SyncStocksClient.**get_daily_open_close**(*symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the OCHLV and after-hours prices of a stock symbol on a certain date. Official Docs

Parameters

- **symbol** – The ticker symbol we want daily-OCHLV for.
- **date** – The date/day of the daily-OCHLV to retrieve. Could be datetime or date or string YYYY-MM-DD
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

2.8 Get Aggregate Bars (Candles)

The library added a better aggregate function if you're looking to get data for large time frames at minute/hour granularity.

(for example 15 years historical data , 1 minute candles)

See *Better Aggregate Bars function* for complete details on how to use it well and control how it behaves.

SyncStocksClient.**get_aggregate_bars**(*symbol: str*, *from_date*, *to_date*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a stock over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **from_date** – The start of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. `asc` default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to `day`
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

2.9 Get Grouped daily Bars (Candles)

SyncStocksClient.**get_grouped_daily_bars**(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily OCHLV for the entire stocks/equities markets. Official docs

Parameters

- **date** – The date to get the data for. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.10 Get Previous Close

SyncStocksClient.**get_previous_close**(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's OCHLV for the specified stock ticker. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.11 Get Snapshot

SyncStocksClient.**get_snapshot**(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded stock ticker. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.12 Get Snapshot (All)

`SyncStocksClient.get_snapshot_all(symbols: Optional[list] = None, raw_response: bool = False)`

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded stock symbols. Official Docs

Parameters

- **symbols** – A comma separated list of tickers to get snapshots for. Defaults to ALL tickers
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

2.13 Get Current Price

`SyncStocksClient.get_current_price(symbol: str)` → float

get current market price for the ticker symbol specified.

Uses *get_last_trade()* under the hood Official Docs

Parameters
symbol – The ticker symbol of the stock/equity.

Returns
The current price. A `KeyError` indicates the request wasn't successful.

2.14 Get Gainers & Losers

`SyncStocksClient.get_gainers_and_losers(direction='gainers', raw_response: bool = False)`

Get the current top 20 gainers or losers of the day in stocks/equities markets. Official Docs

Parameters

- **direction** – The direction of results. Defaults to gainers. See *polygon.enums.SnapshotDirection* for choices
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

CHAPTER

THREE

OPTIONS

Read this page to know everything you need to know about using the various Options HTTP endpoints.

See *Async Support for REST endpoints* for asynchronous use cases.

Docs below assume you have already read getting started page and know how to create the client. If you do not know how to create the client, first see *General guide for clients & functions* and create client as below. As always you can have all 5 different clients together.

```
import polygon

options_client = polygon.OptionsClient('KEY')  # for usual sync client
async_options_client = polygon.OptionsClient('KEY', True)  # for an async client
```

here is how the client initializer looks like:

polygon.options.options.**OptionsClient**(*api_key: str*, *use_async: bool = False*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all REST options endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **use_async** – Set it to `True` to get async client. Defaults to usual non-async client.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.

- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

NOTE: if you don't want to use the option symbol helper functions, then you can just go to the desired endpoint documentation from the list to left

3.1 Working with Option Symbols

So when you're working with options (rest/websockets), you'll certainly need the option symbols which contain the information about their underlying symbol, expiry, call_or_put and the strike price in a certain format. There are many formats to represent them and every data source/brokerage uses a different format to represent them.

for example Polygon.io tends to use This Format . For those who want to understand how this formatting works, Here is a guide (thanks to Ian from polygon support team).

The library is equipped with a few functions to make it easier for you to **build, parse, convert, and detect format** of option symbols without worrying about how the structure works.

This section has been written again following many changes in v1.0.8. If you were using option symbology in v1.0.7 or older, the documentation for that version is available [here](https://polygon.readthedocs.io/en/1.0.7/) although I'd suggest upgrading and making required (small) changes

The library supports the following symbol formats at the moment

Full Name	Shorthand String
Polygon.io	polygon
Tradier	tradier
Trade Station	trade_station
Interactive Brokers	ibkr
TD Ameritrade	tda
Think Or Swim	tos

This section on option symbols is divided into these sections below.

1. **Creating** Option symbols from info as underlying, expiry, strike price, option type
2. **Parsing** Option symbols to **extract** info as underlying, expiry, strike price, option type
3. **Converting** an option symbol from one format to another. Works between all supported formats.
4. **Detecting** format of an option symbol. Basic detection based on some simple rules.

3.1.1 Creating Option Symbols

The function in this sub-section helps you to build option symbols from info as underlying symbol, expiry, strike price & option type (call/put). The function to use is `polygon.build_option_symbol`

- Since the default format is polygon.io you don't need to specify a format if you're only working with polygon option symbols.
- Polygon has a rest endpoint in reference client to get all active contracts which you can filter based on many values such as underlying symbol and expiry dates.
- In polygon format, If you wonder whether you need to worry about the `O:` prefix which some/all option endpoints expect, then to your ease, the library handles that for you (So you can pass a symbol without prefix to let's

say Option Snapshot function and the prefix will be added internally). If you want to be explicit, just pass `prefix_o=True` when building symbol.

- Note that both tradier and polygon happen to use the exact same symbol format and hence can be used interchangeably.

Example Code & Output for polygon/tradier format

```
import polygon

symbol1 = polygon.build_option_symbol('AMD', datetime.date(2022, 6, 28), 'call', 546.56)
symbol2 = polygon.build_option_symbol('TSLA', '220628', 'c', 546, _format='polygon')
symbol3 = polygon.build_option_symbol('A', '220628', 'put', 66.01, prefix_o=True)

# Outputs
# symbol1 -> AMD220628C00546560
# symbol2 -> TSLA220628C00546000
# symbol3 -> O:A220628P00066010
```

The same function can be used to create option symbols for any of the supported formats, just pass in the format you need, either as a shorthand string from the table above, or use an enum from *polygon.enums.OptionSymbolFormat*

- Using enums (like `OptionSymbolFormat.POLYGON` in example below) is a good way to ensure you only pass in correct shorthand strings. Your IDE auto completion would make your life much easier when working with enums.

Example code & outputs for multiple formats

```
from polygon import build_option_symbol  # you can import the way you like, just showing␣
↪the alternates
from polygon.enums import OptionSymbolFormat  # optional, you can pass in shorthand␣
↪strings too

symbol1 = polygon.build_option_symbol('AMD', datetime.date(2022, 6, 28), 'call', 546.56,␣
↪_format='tda')
symbol2 = polygon.build_option_symbol('NVDA', '220628', 'c', 546, _format='tos')
symbol3 = polygon.build_option_symbol('TSLA', datetime.date(2022, 6, 28), 'put', 46.01, _
↪format='tradier')
symbol4 = polygon.build_option_symbol('A', datetime.date(2022, 6, 28), 'p', 46.1, _
↪format='ibkr')
symbol5 = polygon.build_option_symbol('AB', datetime.date(2022, 6, 28), 'p', 46.01, _
↪format='trade_station')
symbol6 = polygon.build_option_symbol('PTON', '220628', 'p', 46, _
↪format=OptionSymbolFormat.POLYGON)  # using enum

# outputs
# symbol1 -> AMD_062822C546.56
# symbol2 -> .NVDA062822C546
# symbol3 -> TSLA220628P00046010
# symbol4 -> A 220628P00046100
# symbol5 -> AB 220628P46.01
# symbol5 -> PTON220628P00046000
```

For those who want more control, here is how the function signature and arguments look

polygon.options.options.**build_option_symbol**(*underlying_symbol: str*, *expiry*, *call_or_put*, *strike_price*, *_format='polygon'*, *prefix_o: bool = False*) → str

Generic function to build option symbols for ALL supported formats: *polygon.enums.OptionSymbolFormat*. Default format is `polygon`.

Parameters

- **underlying_symbol** – The underlying stock ticker symbol.
- **expiry** – The expiry date for the option. You can pass this argument as `datetime.datetime` or `datetime.date` object. Or a string in format: `YYMMDD`. Using datetime objects is recommended.
- **call_or_put** – The option type. You can specify: `c` or `call` or `p` or `put`. Capital letters are also supported.
- **strike_price** – The strike price for the option. ALWAYS pass this as one number. `145`, `240.5`, `15.003`, `56`, `129.02` are all valid values. Try to keep up to 3 digits after the decimal point
- **_format** – The format to use when building symbol. Defaults to `polygon`. Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*
- **prefix_o** – Whether to prefix the symbol with `O:`. It is needed by polygon endpoints. However, all the library functions will automatically add this prefix if you pass in symbols without this prefix. This parameter is **ignored** if format is not `polygon`

Returns

The option symbols string in the format specified

3.1.2 Parsing Option Symbols

The function in this sub-section helps you to extract info as underlying symbol, expiry, strike price & option type (call/put) from an existing option symbol. Parsing is available on all supported formats. The function to use is `polygon.build_option_symbol`

- Since the default format is `polygon`, you don't need to specify a format if you're only working with polygon option symbols.
- Polygon symbols can be passed in with or without the prefix `O:`. Library will handle both internally
- Note that both tradier and polygon happen to use the exact same symbol format and hence can be used interchangeably.
- It is observed that some option symbols as returned by polygon endpoints happen to have a **correction number** within the symbol. The additional number is always between the underlying symbol and expiry. **The lib handles that for you** & parses the symbol accordingly.
- An example of the corrected polygon symbol could be `XY1221015C00234000`. Notice the extra 1 after `XY` and before expiry `221015`. The library would parse this symbol as `XY221015C00234000`. The number could be any number according to a response from polygon support team.

NOTE: The parse function takes another optional argument, `output_format`, defaulting to `'object'`. Here is what it is for and how you can use it to your advantage.

Output Format

The library provides 3 possible output formats when getting parsed info from an option symbol. They are

- An object of class *polygon.options.options.OptionSymbol* (Default). You can access info as
- `obj.strike_price`

- `obj.underlying_symbol`
- `obj.expiry`
- `obj.call_or_put`
- `obj.option_symbol`
- As a list having elements: `[underlying_symbol, expiry, call_or_put, strike_price, option_symbol]` in this fixed order
- As a dict having the following keys:
- `underlying_symbol`
- `expiry`
- `call_or_put`
- `strike_price`
- `option_symbol`

Example code and output for polygon/tradier formats

```
import polygon

parsed_details1 = polygon.parse_option_symbol('AMD211205C00156000')
parsed_details2 = polygon.parse_option_symbol('AMD211205C00156000', output_format=list)
parsed_details3 = polygon.parse_option_symbol('AMD211205C00156000', output_format=dict)

# outputs
# parsed_details1 would be an object having info as attributes as described in output␣
↪format sub-section above
# parsed_details2 -> ['AMD', dt.date(2021, 12, 5), 'C', 156, 'AMD211205C00156000']
# parsed_details3 -> {'underlying_symbol': 'AMD', 'expiry': dt.date(2021, 12, 5), 'call_or_put
↪': 'C',  'strike_price': 156, 'option_symbol': 'AMD211205C00156000'}
```

The same function can be used to parse option symbols in any of the supported formats, just pass in the format you need, either as a shorthand string from the table above, or use an enum from *polygon.enums.OptionSymbolFormat*

- Using enums (like `OptionSymbolFormat.POLYGON` in example below) is a good way to ensure you only pass in correct shorthand strings. Your IDE auto completion would make your life much easier when working with enums.

Example code & outputs for multiple formats

```
import polygon

parsed_details1 = polygon.parse_option_symbol('AMD211205C00156000', _format='tradier')
parsed_details2 = polygon.parse_option_symbol('AMD_062822P587.56', _format='tda', output_
↪format=list)
parsed_details3 = polygon.parse_option_symbol('AB 220628P46.01', _format='trade_station',
↪ output_format=dict)

# outputs
# parsed_details1 would be an object having info as attributes as described in output␣
↪format sub-section above
# parsed_details2 -> ['AMD', dt.date(2022, 6, 28), 'P', 587.56, 'AMD_062822P587.56']
# parsed_details3 -> {'underlying_symbol': 'AB', 'expiry': dt.date(2022, 6, 28), 'call_or_put
↪': 'P', 'strike_price': 46.01, 'option_symbol': 'AB 220628P46.01'}
```

(continues on next page)

(continued from previous page)

For those who want more control, here is how the function signature and arguments look

polygon.options.options.**parse_option_symbol**(*option_symbol: str*, *_format='polygon'*, *output_format='object'*)

Generic function to build option symbols for ALL supported formats: *polygon.enums.OptionSymbolFormat*. Default format is `polygon`.

Parameters

- **option_symbol** – the option symbol you want to parse
- **_format** – What format the symbol is in. If you don't know the format you can use the `detect_option_symbol_format` function to detect the format (best effort detection). Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*. Default: `polygon`
- **output_format** – Output format of the result. defaults to object. Set it to `dict` or `list` as needed.

Returns

The parsed info from symbol either as an object, list or a dict as indicated by `output_format`.

3.1.3 Converting Option Symbol Formats

The function in this sub-section helps you to convert an option symbol from one format to another. So if you want to convert a polygon option symbol to TD Ameritrade symbol (say to place an order), pass the symbol in this function, specify the formats and the library will do the conversions for you.

Example code and outputs

```
import polygon

symbol1 = polygon.convert_option_symbol_formats('AMD220628P00096050', from_format=
↪'polygon', to_format='tda')
symbol2 = polygon.convert_option_symbol_formats('AB 220628P46.01', from_format='trade_
↪station', to_format='polygon')
symbol2 = polygon.convert_option_symbol_formats('NVDA220628C00546000', 'tradier', 'tos')

# outputs
# symbol1 -> AMD_062822P96.05
# symbol2 -> AB220628P00046010
# symbol3 -> .NVDA062822C546
```

For those who want more control, here is how the function signature and arguments look

polygon.options.options.**convert_option_symbol_formats**(*option_symbol: str*, *from_format: str*, *to_format: str*) → str

Convert an option symbol from one format to another within supported formats: *polygon.enums.OptionSymbolFormat*

Parameters

- **option_symbol** – The option symbol you want to convert

- **from_format** – The format in which the option symbol is currently in. If you don't know the format you can use the `detect_option_symbol_format` function to detect the format (best effort detection). Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*
- **to_format** – The format to which you want to convert the option symbol. Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*

Returns
The converted option symbol as a string

3.1.4 Detecting Option Symbol Format

The function in this sub-section helps you to detect the symbol format of an option symbol programmatically. The function does basic detection according to some simple rules so test well before using this in production setting. It is almost always recommended to be explicit about formats.

Example code and outputs

```
import polygon

format1 = polygon.detect_option_symbol_format('AMD_062822P96.05')
format2 = polygon.detect_option_symbol_format('AB220628P00046010')
format3 = polygon.detect_option_symbol_format('.NVDA062822C546')
format4 = polygon.detect_option_symbol_format('AB 220628P46.01')
format5 = polygon.detect_option_symbol_format('AB 220628P00046045')

# outputs
# format1 -> 'tda'
# format2 -> 'polygon'  # this also means tradier since both use exact same format
# format3 -> 'tos'
# format4 -> 'trade_station'
# format5 -> ['ibkr', 'trade_station']
```

For those who want more control, here is how the function signature and arguments look

`polygon.options.options.`**detect_option_symbol_format**(*option_symbol: str*) → Union[str, bool, list]

Detect what format a symbol is formed in. Supported formats are *polygon.enums.OptionSymbolFormat*. This function does basic detection according to some simple rules. Test well before using in production.

Parameters
option_symbol – The option symbol to check the format of

Returns
Format's shorthand string or list of strings if able to recognize the format. `False` otherwise. Possible shorthand strings are `polygon, tda, tos, ibkr, tradier, trade_station`

Endpoints:

To use any of the below method, simply call it on the client you created above. so if you named your client `client`, you'd call the methods as `client.get_trades` and so on. Async methods will need to be awaited, see *Async Support for REST endpoints*.

3.2 Get Trades

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncOptionsClient.get_trades`(*option_symbol: str*, *timestamp=None*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *sort='timestamp'*, *limit: int = 5000*, *order='asc'*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for an options ticker symbol in a given time range. Note that you need to have an option symbol in correct format for this endpoint. You can use `ReferenceClient.get_option_contracts` to query option contracts using many filter parameters such as underlying symbol etc. Official Docs

Parameters

- **option_symbol** – The options ticker symbol to get trades for. for eg `O:TSLA210903C00700000`. you can pass the symbol with or without the prefix `O:`
- **timestamp** – Query by trade timestamp. You can supply a `date`, `datetime` object or a `nanosecond UNIX timestamp` or a string in format: `YYYY-MM-DD`.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **sort** – Sort field used for ordering. Defaults to timestamp. See *polygon.enums.OptionTradesSort* for available choices.
- **limit** – Limit the number of results returned. Defaults to 5000. max is 50000.
- **order** – order of the results. Defaults to `asc`. See *polygon.enums.SortOrder* for info and available choices.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

3.3 Get Quotes

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncOptionsClient.get_quotes`(*option_symbol: str*, *timestamp=None*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *sort='timestamp'*, *limit: int = 5000*, *order='asc'*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get quotes for an options ticker symbol in a given time range. Note that you need to have an option symbol in correct format for this endpoint. You can use `ReferenceClient.get_option_contracts` to query option contracts using many filter parameters such as underlying symbol etc. Official Docs

Parameters

- **option_symbol** – The options ticker symbol to get quotes for. for eg `O:TSLA210903C00700000`. you can pass the symbol with or without the prefix `O:`
- **timestamp** – Query by quote timestamp. You can supply a `date`, `datetime` object or a `nanosecond UNIX timestamp` or a string in format: `YYYY-MM-DD`.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **sort** – Sort field used for ordering. Defaults to timestamp. See *polygon.enums.OptionQuotesSort* for available choices.
- **limit** – Limit the number of results returned. Defaults to 5000. max is 50000.
- **order** – order of the results. Defaults to `asc`. See *polygon.enums.SortOrder* for info and available choices.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or

decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True

- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

3.4 Get Last Trade

`SyncOptionsClient.get_last_trade`(*ticker: str*, *raw_response: bool = False*)

Get the most recent trade for a given options contract. Official Docs

Parameters

- **ticker** – The ticker symbol of the options contract. Eg: `O:TSLA210903C00700000`
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of `raw_response`. Defaults to Dict.

3.5 Get Daily Open Close

`SyncOptionsClient.get_daily_open_close`(*symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the OCHLV and after-hours prices of a contract on a certain date. Official Docs

Parameters

- **symbol** – The option symbol we want daily-OCHLV for. eg `O:FB210903C00700000`. You can pass it with or without the prefix `O:`
- **date** – The date/day of the daily-OCHLV to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

3.6 Get Aggregate Bars

The library added a better aggregate function if you're looking to get data for large time frames at minute/hour granularity.

(for example 15 years historical data , 1 minute candles)

See *Better Aggregate Bars function* for complete details on how to use it well and control how it behaves.

`SyncOptionsClient.`**`get_aggregate_bars`**(*symbol: str*, *from_date*, *to_date*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for an option contract over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then 5-minute bars will be returned. Official Docs

Parameters

- **`symbol`** – The ticker symbol of the contract. eg `O:FB210903C00700000`. You can pass in with or without the prefix `O:`
- **`from_date`** – The start of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **`to_date`** – The end of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **`adjusted`** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **`sort`** – Sort the results by timestamp. See *`polygon.enums.SortOrder`* for choices. `asc` default.
- **`limit`** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000. see this article for more info.
- **`multiplier`** – The size of the timespan multiplier. Must be a positive whole number. defaults to 1.
- **`timespan`** – The size of the time window. See *`polygon.enums.Timespan`* for choices. defaults to `day`
- **`full_range`** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **`run_parallel`** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **`max_concurrent_workers`** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool

- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

3.7 Get Previous Close

`SyncOptionsClient.get_previous_close(`*ticker: str*, *adjusted: bool = True*, *raw_response: bool = False*`)`

Get the previous day's open, high, low, and close (OHLC) for the specified option contract. Official Docs

Parameters

- **ticker** – The ticker symbol of the options contract. Eg: `O:TSLA210903C00700000`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of `raw_response`. Defaults to Dict.

3.8 Get Snapshot

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncOptionsClient.get_snapshot(`*underlying_symbol: str*, *option_symbol: str*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*`)`

Get the snapshot of an option contract for a stock equity. Official Docs

Parameters

- **underlying_symbol** – The underlying ticker symbol of the option contract. eg `AMD`
- **option_symbol** – the option symbol. You can use use the *Working with Option Symbols* section to make it easy to work with option symbols in polygon or tda formats.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.

- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

CHAPTER

FOUR

REFERENCE APIS

Read this page to know everything you need to know about using the various References HTTP endpoints.

See *Async Support for REST endpoints* for asynchronous use cases.

Docs below assume you have already read getting started page and know how to create the client. If you do not know how to create the client, first see *General guide for clients & functions* and create client as below. As always you can have all 5 different clients together.

```
import polygon

reference_client = polygon.ReferenceClient('KEY')  # for usual sync client
async_reference_client = polygon.ReferenceClient('KEY', True)  # for an async client
```

here is how the client initializer looks like:

polygon.reference_apis.reference_api.**ReferenceClient**(*api_key: str*, *use_async: bool = False*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all REST References endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **use_async** – Set it to `True` to get async client. Defaults to usual non-async client.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.

- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

Endpoints

To use any of the below method, simply call it on the client you created above. so if you named your client `client`, you'd call the methods as `client.get_tickers` and so on. Async methods will need to be awaited, see *Async Support for REST endpoints*.

4.1 Get Tickers

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncReferenceClient.get_tickers`(*symbol: str = ''*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *symbol_type=''*, *market=''*, *exchange: str = ''*, *cusip: Optional[str] = None*, *cik: str = ''*, *date=None*, *search: Optional[str] = None*, *active: bool = True*, *sort='ticker'*, *order='asc'*, *limit: int = 1000*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Query all ticker symbols which are supported by Polygon.io. This API currently includes Stocks/Equities, Crypto, and Forex. Official Docs

Parameters

- **symbol** – Specify a ticker symbol. Defaults to empty string which queries all tickers.
- **ticker_lt** – Return results where this field is less than the value given
- **ticker_lte** – Return results where this field is less than or equal to the value given
- **ticker_gt** – Return results where this field is greater than the value given
- **ticker_gte** – Return results where this field is greater than or equal to the value given
- **symbol_type** – Specify the type of the tickers. See *polygon.enums.TickerType* for common choices. Find all supported types via the Ticker Types API Defaults to empty string which queries all types.
- **market** – Filter by market type. By default all markets are included. See *polygon.enums.TickerMarketType* for available choices.
- **exchange** – Specify the primary exchange of the asset in the ISO code format. Find more information about the ISO codes at the ISO org website. Defaults to empty string which queries all exchanges.
- **cusip** – Specify the `CUSIP` code of the asset you want to search for. Find more information about CUSIP codes on their website Defaults to empty string which queries all CUSIPs
- **cik** – Specify the `CIK` of the asset you want to search for. Find more information about CIK codes at their website Defaults to empty string which queries all CIKs.
- **date** – Specify a point in time to retrieve tickers available on that date. Defaults to the most recent available date. Could be `datetime`, `date` or a string `YYYY-MM-DD`
- **search** – Search for terms within the ticker and/or company name. for eg `MS` will match matching symbols

- **active** – Specify if the tickers returned should be actively traded on the queried date. Default is True
- **sort** – The field to sort the results on. Default is ticker. If the search query parameter is present, sort is ignored and results are ordered by relevance. See *polygon.enums.TickerSortType* for available choices.
- **order** – The order to sort the results on. Default is asc. See *polygon.enums.SortOrder* for available choices.
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

4.2 Get Ticker Types

`SyncReferenceClient.get_ticker_types`(*asset_class=None*, *locale=None*, *raw_response: bool = False*)

Get a mapping of ticker types to their descriptive names. Official Docs

Parameters

- **asset_class** – Filter by asset class. see *polygon.enums.AssetClass* for choices
- **locale** – Filter by locale. See *polygon.enums.Locale* for choices
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

4.3 Get Ticker Details

SyncReferenceClient.**get_ticker_details**(*symbol: str*, *date=None*, *raw_response: bool = False*)

Get a single ticker supported by Polygon.io. This response will have detailed information about the ticker and the company behind it. Official Docs

Parameters

- **symbol** – The ticker symbol of the asset.
- **date** – Specify a point in time to get information about the ticker available on that date. When retrieving information from SEC filings, we compare this date with the period of report date on the SEC filing. Defaults to the most recent available date.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

4.4 Get Option Contract

SyncReferenceClient.**get_option_contract**(*ticker: str*, *as_of_date=None*, *raw_response: bool = False*)

get Info about an option contract Official Docs

Parameters

- **ticker** – An option ticker in standard format. The lib provides easy functions to build and work with option symbols
- **as_of_date** – Specify a point in time for the contract. You can pass a datetime or date object or a string in format YYYY-MM-DD. Defaults to today's date
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

4.5 Get Option Contracts

This endpoint supports pagination. Passing all_pages=True enables it. See *Pagination Support* for better info

SyncReferenceClient.**get_option_contracts**(*underlying_ticker: Optional[str] = None*, *ticker: Optional[str] = None*, *contract_type=None*, *expiration_date=None*, *expiration_date_lt=None*, *expiration_date_lte=None*, *expiration_date_gt=None*, *expiration_date_gte=None*, *order='asc'*, *sort='expiration_date'*, *limit=1000*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

List currently active options contracts Official Docs

Parameters

- **underlying_ticker** – Query for contracts relating to an underlying stock ticker.
- **ticker** – Query for a contract by option ticker.
- **contract_type** – Query by the type of contract. see *polygon.enums.OptionsContractType* for choices
- **expiration_date** – Query by contract expiration date. either `datetime`, `date` or string `YYYY-MM-DD`
- **expiration_date_lt** – expiration date less than given value
- **expiration_date_lte** – expiration date less than equal to given value
- **expiration_date_gt** – expiration_date greater than given value
- **expiration_date_gte** – expiration_date greater than equal to given value
- **order** – Order of results. See *polygon.enums.SortOrder* for choices.
- **sort** – Sort field for ordering. See *polygon.enums.OptionsContractsSortType* for choices. defaults to expiration_date
- **limit** – Limit the size of the response, default is 1000. `Pagination` is supported by the pagination function below
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

4.6 Get Ticker News

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncReferenceClient.get_ticker_news`(*symbol: Optional[str] = None*, *limit: int = 1000*, *order='desc'*, *sort='published_utc'*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *published_utc=None*, *published_utc_lt=None*, *published_utc_lte=None*, *published_utc_gt=None*, *published_utc_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get the most recent news articles relating to a stock ticker symbol, including a summary of the article and a link to the original source. Official Docs

Parameters

- **symbol** – To get news mentioning the name given. Defaults to empty string which doesn't filter tickers
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below
- **order** – Order the results. See *polygon.enums.SortOrder* for choices.
- **sort** – The field key to sort. See *polygon.enums.TickerNewsSort* for choices.
- **ticker_lt** – Return results where this field is less than the value.
- **ticker_lte** – Return results where this field is less than or equal to the value.
- **ticker_gt** – Return results where this field is greater than the value
- **ticker_gte** – Return results where this field is greater than or equal to the value.
- **published_utc** – A date string `YYYY-MM-DD` or `datetime` for published date time filters.
- **published_utc_lt** – Return results where this field is less than the value given
- **published_utc_lte** – Return results where this field is less than or equal to the value given
- **published_utc_gt** – Return results where this field is greater than the value given
- **published_utc_gte** – Return results where this field is greater than or equal to the value given
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.

- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

4.7 Get Stock dividends

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncReferenceClient.get_stock_dividends`(*ticker: Optional[str] = None*, *ex_dividend_date=None*, *record_date=None*, *declaration_date=None*, *pay_date=None*, *frequency: Optional[int] = None*, *limit: int = 1000*, *cash_amount=None*, *dividend_type=None*, *sort: str = 'pay_date'*, *order: str = 'asc'*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *ex_dividend_date_lt=None*, *ex_dividend_date_lte=None*, *ex_dividend_date_gt=None*, *ex_dividend_date_gte=None*, *record_date_lt=None*, *record_date_lte=None*, *record_date_gt=None*, *record_date_gte=None*, *declaration_date_lt=None*, *declaration_date_lte=None*, *declaration_date_gt=None*, *declaration_date_gte=None*, *pay_date_lt=None*, *pay_date_lte=None*, *pay_date_gt=None*, *pay_date_gte=None*, *cash_amount_lt=None*, *cash_amount_lte=None*, *cash_amount_gt=None*, *cash_amount_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get a list of historical cash dividends, including the ticker symbol, declaration date, ex-dividend date, record date, pay date, frequency, and amount. Official Docs

Parameters

- **ticker** – Return the dividends that contain this ticker.
- **ex_dividend_date** – Query by ex-dividend date. could be a date, datetime object or a string `YYYY-MM-DD`
- **record_date** – Query by record date. could be a date, datetime object or a string `YYYY-MM-DD`
- **declaration_date** – Query by declaration date. could be a date, datetime object or a string `YYYY-MM-DD`
- **pay_date** – Query by pay date. could be a date, datetime object or a string `YYYY-MM-DD`
- **frequency** – Query by the number of times per year the dividend is paid out. No default value applied. see *polygon.enums.PayoutFrequency* for choices
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below

- **cash_amount** – Query by the cash amount of the dividend.
- **dividend_type** – Query by the type of dividend. See *polygon.enums.DividendType* for choices
- **sort** – sort key used for ordering. See *polygon.enums.DividendSort* for choices.
- **order** – orders of results. defaults to asc. see *polygon.enums.SortOrder* for choices
- **ticker_lt** – filter where ticker is less than given value (alphabetically)
- **ticker_lte** – filter where ticker is less than or equal to given value (alphabetically)
- **ticker_gt** – filter where ticker is greater than given value (alphabetically)
- **ticker_gte** – filter where ticker is greater than or equal to given value (alphabetically)
- **ex_dividend_date_lt** – filter where ex-div date is less than given date
- **ex_dividend_date_lte** – filter where ex-div date is less than or equal to given date
- **ex_dividend_date_gt** – filter where ex-div date is greater than given date
- **ex_dividend_date_gte** – filter where ex-div date is greater than or equal to given date
- **record_date_lt** – filter where record date is less than given date
- **record_date_lte** – filter where record date is less than or equal to given date
- **record_date_gt** – filter where record date is greater than given date
- **record_date_gte** – filter where record date is greater than or equal to given date
- **declaration_date_lt** – filter where declaration date is less than given date
- **declaration_date_lte** – filter where declaration date is less than or equal to given date
- **declaration_date_gt** – filter where declaration date is greater than given date
- **declaration_date_gte** – filter where declaration date is greater than or equal to given date
- **pay_date_lt** – filter where pay date is less than given date
- **pay_date_lte** – filter where pay date is less than or equal to given date
- **pay_date_gt** – filter where pay date is greater than given date
- **pay_date_gte** – filter where pay date is greater than or equal to given date
- **cash_amount_lt** – filter where cash amt is less than given value
- **cash_amount_lte** – filter where cash amt is less than or equal to given value
- **cash_amount_gt** – filter where cash amt is greater than given value
- **cash_amount_gte** – filter where cash amt is greater than or equal to given value
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True

- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

4.8 Get Stock financials vX

`SyncReferenceClient.get_stock_financials_vx`(*ticker: Optional[str] = None*, *cik: Optional[str] = None*, *company_name: Optional[str] = None*, *company_name_search: Optional[str] = None*, *sic: Optional[str] = None*, *filing_date=None*, *filing_date_lt=None*, *filing_date_lte=None*, *filing_date_gt=None*, *filing_date_gte=None*, *period_of_report_date=None*, *period_of_report_date_lt=None*, *period_of_report_date_lte=None*, *period_of_report_date_gt=None*, *period_of_report_date_gte=None*, *time_frame=None*, *include_sources: bool = False*, *order='asc'*, *limit: int = 50*, *sort='filing_date'*, *raw_response: bool = False*)

Get historical financial data for a stock ticker. The financials data is extracted from XBRL from company SEC filings using this methodology Official Docs

This API is experimental and will replace `get_stock_financials()` in future.

Parameters

- **ticker** – Filter query by company ticker.
- **cik** – filter the Query by `central index key (CIK)` Number
- **company_name** – filter the query by company name
- **company_name_search** – partial match text search for company names
- **sic** – Query by `standard industrial classification (SIC)`
- **filing_date** – Query by the date when the filing with financials data was filed. `datetime/date` or string `YYYY-MM-DD`
- **filing_date_lt** – filter for filing date less than given value
- **filing_date_lte** – filter for filing date less than equal to given value
- **filing_date_gt** – filter for filing date greater than given value
- **filing_date_gte** – filter for filing date greater than equal to given value
- **period_of_report_date** – query by The period of report for the filing with financials data. `datetime/date` or string in format: `YYY-MM-DD`.

- **period_of_report_date_lt** – filter for period of report date less than given value
- **period_of_report_date_lte** – filter for period of report date less than equal to given value
- **period_of_report_date_gt** – filter for period of report date greater than given value
- **period_of_report_date_gte** – filter for period of report date greater than equal to given value
- **time_frame** – Query by timeframe. Annual financials originate from 10-K filings, and quarterly financials originate from 10-Q filings. Note: Most companies do not file quarterly reports for Q4 and instead include those financials in their annual report, so some companies my not return quarterly financials for Q4. See *polygon.enums.StockFinancialsTimeframe* for choices.
- **include_sources** – Whether or not to include the xpath and formula attributes for each financial data point. See the xpath and formula response attributes for more info. `False` by default
- **order** – Order results based on the sort field. 'asc' by default. See *polygon.enums.SortOrder* for choices.
- **limit** – number of max results to obtain. defaults to 50.
- **sort** – Sort field key used for ordering. 'filing_date' default. see *polygon.enums.StockFinancialsSortKey* for choices.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

4.9 Get Stock Splits

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncReferenceClient.`**`get_stock_splits`**(*ticker: Optional[str] = None*, *execution_date=None*, *reverse_split: Optional[bool] = None*, *order: str = 'asc'*, *sort: str = 'execution_date'*, *limit: int = 1000*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *execution_date_lt=None*, *execution_date_lte=None*, *execution_date_gt=None*, *execution_date_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get a list of historical stock splits, including the ticker symbol, the execution date, and the factors of the split ratio. Official Docs

Parameters

- **ticker** – Return the stock splits that contain this ticker. defaults to no ticker filter returning all.
- **execution_date** – query by execution date. could be a date, datetime object or a string `YYYY-MM-DD`

- **reverse_split** – Query for reverse stock splits. A split ratio where split_from is greater than split_to represents a reverse split. By default this filter is not used.
- **order** – Order results based on the sort field. defaults to ascending. See *polygon.enums.SortOrder* for choices
- **sort** – Sort field used for ordering. Defaults to 'execution_date'. See *polygon.enums.SplitsSortKey* for choices.
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below
- **ticker_lt** – filter where ticker name is less than given value (alphabetically)
- **ticker_lte** – filter where ticker name is less than or equal to given value (alphabetically)
- **ticker_gt** – filter where ticker name is greater than given value (alphabetically)
- **ticker_gte** – filter where ticker name is greater than or equal to given value (alphabetically)
- **execution_date_lt** – filter where execution date is less than given value
- **execution_date_lte** – filter where execution date is less than or equal to given value
- **execution_date_gt** – filter where execution date is greater than given value
- **execution_date_gte** – filter where execution date is greater than or equal to given value
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

4.10 Get Market Holidays

SyncReferenceClient.**get_market_holidays**(*raw_response: bool = False*)

Get upcoming market holidays and their open/close times. Official Docs

Parameters

raw_response – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

4.11 Get Market Status

SyncReferenceClient.**get_market_status**(*raw_response: bool = False*)

Get the current trading status of the exchanges and overall financial markets. Official Docs

Parameters

raw_response – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

4.12 Get Conditions

SyncReferenceClient.**get_conditions**(*asset_class=None*, *data_type=None*, *condition_id=None*, *sip=None*, *order=None*, *limit: int = 50*, *sort='name'*, *raw_response: bool = False*)

List all conditions that Polygon.io uses. Official Docs

Parameters

- **asset_class** – Filter for conditions within a given asset class. See *polygon.enums.AssetClass* for choices. Defaults to all assets.
- **data_type** – Filter by data type. See *polygon.enums.ConditionsDataType* for choices. defaults to all.
- **condition_id** – Filter for conditions with a given ID
- **sip** – Filter by SIP. If the condition contains a mapping for that SIP, the condition will be returned.
- **order** – Order results. See *polygon.enums.SortOrder* for choices.
- **limit** – limit the number of results. defaults to 50.
- **sort** – Sort field used for ordering. Defaults to 'name'. See *polygon.enums.ConditionsSortKey* for choices.

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

4.13 Get Exchanges

`SyncReferenceClient.`**`get_exchanges`**(*asset_class=None*, *locale=None*, *raw_response: bool = False*)

List all exchanges that Polygon.io knows about. Official Docs

Parameters

- **asset_class** – filter by asset class. See *polygon.enums.AssetClass* for choices.
- **locale** – Filter by locale name. See *polygon.enums.Locale*
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

CHAPTER

FIVE

FOREX

Read this page to know everything you need to know about using the various Forex HTTP endpoints.

See *Async Support for REST endpoints* for asynchronous use cases.

Docs below assume you have already read getting started page and know how to create the client. If you do not know how to create the client, first see *General guide for clients & functions* and create client as below. As always you can have all 5 different clients together.

```
import polygon

forex_client = polygon.ForexClient('KEY')  # for usual sync client
async_forex_client = polygon.ForexClient('KEY', True)  # for an async client
```

Note that most endpoints require you to specify the currency pairs as separate symbols (a `from_symbol` and a `to_symbol`).

however a few endpoints require you to supply them as one combined symbol. An example would be the `get_aggregates_bars` method. In those methods, the symbol is expected to have a prefix `C:` before the currency symbol names. **but the library allows you to specify the symbol with or without the prefix**. See the relevant method's docs for more information on what the parameters expect.

here is how the client initializer looks like:

polygon.forex.forex_api.**ForexClient**(*api_key: str*, *use_async: bool = False*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all REST Forex endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **use_async** – Set it to `True` to get async client. Defaults to usual non-async client.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies

- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

Endpoints

To use any of the below method, simply call it on the client you created above. so if you named your client `client`, you'd call the methods as `client.get_historic_forex_ticks` and so on. Async methods will need to be awaited, see *Async Support for REST endpoints*.

5.1 Get Historic forex ticks

`SyncForexClient.get_historic_forex_ticks`(*from_symbol: str*, *to_symbol: str*, *date*, *offset: Optional[Union[str, int]] = None*, *limit: int = 500*, *raw_response: bool = False*)

Get historic trade ticks for a forex currency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the forex currency pair.
- **to_symbol** – The "to" symbol of the forex currency pair.
- **date** – The date/day of the historic ticks to retrieve. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **offset** – The timestamp offset, used for pagination. This is the offset at which to start the results. Using the timestamp of the last result as the offset will give you the next page of results. I'm thinking about a good way to implement this type of pagination in the lib which doesn't have a `next_url` in the response attributes.
- **limit** – Limit the size of the response, max 10000. Default 500
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

5.2 Get Quotes (NBBO)

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncForexClient.get_quotes`(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get NBBO Quotes for a forex ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want quotes for. eg: `C:EUR-USD`. you can pass with or without prefix C:
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.ForexQuotesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

5.3 Get Last Quote

SyncForexClient.**get_last_quote**(*from_symbol: str*, *to_symbol: str*, *raw_response: bool = False*)

Get the last trade tick for a forex currency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the forex currency pair.
- **to_symbol** – The "to" symbol of the forex currency pair.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

5.4 Get Aggregate Bars (Candles)

The library added a better aggregate function if you're looking to get data for large time frames at minute/hour granularity.

(for example 15 years historical data , 1 minute candles)

See *Better Aggregate Bars function* for complete details on how to use it well and control how it behaves.

SyncForexClient.**get_aggregate_bars**(*symbol: str*, *from_date*, *to_date*, *multiplier: int = 1*, *timespan='day'*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a forex pair over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then `5-minute` bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the forex pair. eg: `C:EURUSD`. You can supply with or without prefix `C:`
- **from_date** – The start of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **multiplier** – The size of the timespan multiplier
- **timespan** – The size of the time window. Defaults to day candles. see *polygon.enums.Timespan* for choices
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. see *polygon.enums.SortOrder* for available choices. Defaults to `asc` which is oldest at the top.

- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

5.5 Get Grouped Daily Bars (Candles)

`SyncForexClient.get_grouped_daily_bars`(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily open, high, low, and close (OHLC) for the entire forex markets. Official Docs

Parameters

- **date** – The date for the aggregate window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

5.6 Get Previous Close

SyncForexClient.**get_previous_close**(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified forex pair. Official Docs

Parameters

- **symbol** – The ticker symbol of the forex pair.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

5.7 Get Gainers & Losers

SyncForexClient.**get_gainers_and_losers**(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in forex markets. Official docs

Parameters

- **direction** – The direction of the snapshot results to return. See *polygon.enums.SnapshotDirection* for available choices. Defaults to Gainers.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

5.8 Real Time currency conversion

SyncForexClient.**real_time_currency_conversion**(*from_symbol: str*, *to_symbol: str*, *amount: float*, *precision: int = 2*, *raw_response: bool = False*)

Get currency conversions using the latest market conversion rates. Note than you can convert in both directions. For example USD to CAD or CAD to USD. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **amount** – The amount to convert,
- **precision** – The decimal precision of the conversion. Defaults to 2 which is 2 decimal places accuracy.

- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

CHAPTER

SIX

CRYPTO

Read this page to know everything you need to know about using the various Crypto HTTP endpoints.

See *Async Support for REST endpoints* for asynchronous use cases.

Docs below assume you have already read getting started page and know how to create the client. If you do not know how to create the client, first see *General guide for clients & functions* and create client as below. As always you can have all 5 different clients together.

```
import polygon

crypto_client = polygon.CryptoClient('KEY')  # for usual sync client
async_crypto_client = polygon.CryptoClient('KEY', True)  # for an async client
```

Note that most endpoints require you to specify the currency pairs as separate symbols (a `from_symbol` and a `to_symbol`).

however a few endpoints require you to supply them as one combined symbol. An example would be the `get_aggregates_bars` method. In those methods, the symbol is expected to have a prefix `X:` before the currency symbol names. **but the library allows you to specify the symbol with or without the prefix**. See the relevant method's docs for more information on what the parameters expect.

here is how the client initializer looks like:

polygon.crypto.crypto_api.**CryptoClient**(*api_key: str*, *use_async: bool = False*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all REST crypto endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **use_async** – Set it to `True` to get async client. Defaults to usual non-async client.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies

- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

Endpoints

To use any of the below method, simply call it on the client you created above. so if you named your client `client`, you'd call the methods as `client.get_historic_trades` and so on. Async methods will need to be awaited, see *Async Support for REST endpoints*.

6.1 Get Historic Trades

`SyncCryptoClient.get_historic_trades`(*from_symbol: str*, *to_symbol: str*, *date*, *offset: Optional[Union[str, int]] = None*, *limit: int = 500*, *raw_response: bool = False*)

Get historic trade ticks for a cryptocurrency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the crypto pair.
- **to_symbol** – The "to" symbol of the crypto pair.
- **date** – The date/day of the historic ticks to retrieve. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **offset** – The timestamp offset, used for pagination. This is the offset at which to start the results. Using the timestamp of the last result as the offset will give you the next page of results. I'm trying to think of a good way to implement pagination in the library for these endpoints which do not return a `next_url` attribute.
- **limit** – Limit the size of the response, max 10000. Default 500
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.2 Get Trades

This endpoint supports pagination. Passing `all_pages=True` enables it. See *Pagination Support* for better info

`SyncCryptoClient.get_trades`(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for a crypto ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want trades for. eg `X:BTC-USD`. you can pass with or without the prefix `C:`
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.CryptoTradesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

6.3 Get Last Trade

SyncCryptoClient.**get_last_trade**(*from_symbol: str*, *to_symbol: str*, *raw_response: bool = False*)

Get the last trade tick for a cryptocurrency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.4 Get Daily Open Close

SyncCryptoClient.**get_daily_open_close**(*from_symbol: str*, *to_symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the open, close prices of a cryptocurrency symbol on a certain day. Official Docs:

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **date** – The date of the requested open/close. Could be `datetime`, `date` or string `YYYY-MM-DD`.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.5 Get Aggregate Bars (Candles)

The library added a better aggregate function if you're looking to get data for large time frames at minute/hour granularity.

(for example 15 years historical data , 1 minute candles)

See *Better Aggregate Bars function* for complete details on how to use it well and control how it behaves.

SyncCryptoClient.**get_aggregate_bars**(*symbol: str*, *from_date*, *to_date*, *multiplier: int = 1*, *timespan='day'*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a cryptocurrency pair over a given date range in custom time window sizes. For example, if `timespan='minute'` and `multiplier='5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the currency pair. eg: `X:BTCUSD`. You can specify with or without prefix `X:`
- **from_date** – The start of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **multiplier** – The size of the timespan multiplier
- **timespan** – The size of the time window. Defaults to day candles. see *polygon.enums.Timespan* for choices
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **sort** – Order of sorting the results. See *polygon.enums.SortOrder* for available choices. Defaults to `asc` (oldest at the top)
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

6.6 Get Grouped Daily Bars (Candles)

SyncCryptoClient.**get_grouped_daily_bars**(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily open, high, low, and close (OHLC) for the entire cryptocurrency market. Official Docs

Parameters

- **date** – The date for the aggregate window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.7 Get Previous Close

SyncCryptoClient.**get_previous_close**(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified cryptocurrency pair. Official Docs

Parameters

- **symbol** – The ticker symbol of the currency pair. eg: `X:BTCUSD`. You can specify with or without the prefix `X:`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.8 Get Snapshot All

SyncCryptoClient.**get_snapshot_all**(*symbols: list*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded cryptocurrency symbols Official Docs

Parameters

- **symbols** – A list of tickers to get snapshots for.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.9 Get Snapshot

`SyncCryptoClient.`**`get_snapshot`**(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded cryptocurrency symbol. Official Docs

Parameters

- **`symbol`** – Symbol of the currency pair. eg: `X:BTCUSD`. you can specify with or without prefix `X:`
- **`raw_response`** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.10 Get Gainers and Losers

`SyncCryptoClient.`**`get_gainers_and_losers`**(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in cryptocurrency markets. Official docs

Parameters

- **`direction`** – The direction of the snapshot results to return. See *`polygon.enums.SnapshotDirection`* for available choices. Defaults to Gainers.
- **`raw_response`** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

6.11 Get Level 2 Book

`SyncCryptoClient.`**`get_level2_book`**(*symbol: str*, *raw_response: bool = False*)

Get the current level 2 book of a single ticker. This is the combined book from all of the exchanges. Official Docs

Parameters

- **`symbol`** – The cryptocurrency ticker. eg: `X:BTCUSD`. You can specify with or without the prefix `` `X: ``

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

CHAPTER

SEVEN

CALLBACK STREAMING

A convenient wrapper around the Streaming API

IMPORTANT Polygon.io allows one simultaneous connection to one cluster at a time (clusters: stocks, options, forex, crypto). which means 4 total concurrent streams (Of course you need to have subscriptions for them).

Connecting to a cluster which already has an existing stream connected to it would result in existing connection getting dropped and new connection would be established

Note that This page describes the callback based streaming client. If you're looking for async based streaming client, See *Async Streaming*

Also note that callback based streamer is supposed to get a builtin functionality to reconnect in the library. Async streamer has it already. It's on TODO for this client. Have a reconnect mechanism to share? Share in discussions or on the wiki.

7.1 Creating the client

Creating a client is just creating an instance of `polygon.StreamClient`. Note that this expects a few arguments where most of them have default values.

This is how the initializer looks like:

`StreamClient.__init__`(*api_key: str*, *cluster*, *host='socket.polygon.io'*, *on_message=None*, *on_close=None*, *on_error=None*, *enable_connection_logs: bool = False*)

Initializes the callback function based stream client Official Docs

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **cluster** – Which market/cluster to connect to. See *polygon.enums.StreamCluster* for choices. NEVER connect to the same cluster again if there is an existing stream connected to it. The existing connection would be dropped and new one will be established. You can have up to 4 concurrent streams connected to 4 different clusters.
- **host** – Host url to connect to. Default is real time. See *polygon.enums.StreamHost* for choices.
- **on_message** – The function to be called when data is received. This is primary function you'll write to process the data from the stream. The function should accept one and only one `arg` (message). Default handler is *_default_on_msg()*.
- **on_close** – The function to be called when stream is closed. Function should accept two args (close_status_code, close_message). Default handler is *_default_on_close()*

- **on_error** – Function to be called when an error is encountered. Function should accept one arg (exception object). Default handler is *_default_on_error()*
- **enable_connection_logs** – Whether or not to print debug info related to the stream connection. Helpful for debugging.

Example use:

```
import polygon

stream_client = polygon.StreamClient('KEY', 'stocks', on_message=my_own_handler_
↪function)  # in the simplest form
```

Note that you don't have to call login methods as the library does it internally itself.

7.2 Starting the Stream

Once you have a stream client, you can start the stream thread by calling the method: `start_stream_thread`.

This method has default values which should be good enough for most people. For those who need customization, here is how it looks like:

`StreamClient.`**start_stream_thread**(*ping_interval: int = 21*, *ping_timeout: int = 20*, *ping_payload: str = ''*, *skip_utf8_validation: bool = True*)

Starts the Stream. This will not block the main thread and it spawns the streamer in its own thread.

Parameters

- **ping_interval** – client would send a `ping` every specified number of seconds to server to keep connection alive. Set to 0 to disable pinging. Defaults to 21 seconds
- **ping_timeout** – Timeout in seconds if a `pong` (response to ping from server) is not received. The Stream is terminated as it is considered to be dead if no pong is received within the specified timeout. default: 20 seconds
- **ping_payload** – The option message to be sent with the ping. Better to leave it empty string.
- **skip_utf8_validation** – Whether to skip utf validation of messages. Defaults to True. Setting it to False may result in performance downgrade

Returns

None

Example use:

```
import polygon

stream_client = polygon.StreamClient('KEY', 'stocks', on_message=my_own_handler_function)

stream_client.start_stream_thread()

# subscriptions here.
```

7.3 Important Concepts

Important stuff to know before you connect your first stream. Note that when writing applications, you should create the client and start the stream thread before subscribing.

7.3.1 Subscribing/Unsubscribing to Streams

All subscription methods have names in pattern `subscribe_service_name` and `unsubscribe_service_name` (listed below)

Symbols names must be specified as a list of symbols: `['AMD', 'NVDA', 'LOL']` is the correct way to specify symbols. Not specifying a list of symbols results in the action being applied to `ALL` tickers in that service. Note that either of `[]`, `None`, `['*']` or `'all'` as value of symbols would also results in ALL tickers.

The library allows specifying a string for symbol argument (that string is sent exactly as it is without processing), but only do that if you have the absolute need to. Most people should just specify a list. Note that a list of single ticker is accepted.

Options and Crypto stream endpoints expect prefixes ``O:, X:`` respectively in front of every ticker. The library handles this for you so you can pass symbols with or without those prefixes.

By default, the library will also enforce upper case for all symbols being passed. To disable this enforcement, just pass in `force_uppercase_symbols=False` when subscribing in the methods below.

7.3.2 Handling messages

Your handler function should accept two arguments. You can ignore the first argument which is going to be the websocket instance itself. The second argument is the actual message.

In callback streaming, **the library can't do the json decoding for you internally, and you will always receive a raw string** as received from the websocket server. messages). **You will have to do `json decoding` yourself**.

```
def sample_handler(ws, msg):
    print(msg)  # here msg is the raw string which contains the msg. to convert it to a␣
↪list/dict, it needs to be decoded.

    # DECODING the msg from string to list/dict
    # ensure you have 'import json' at the top of file in imports

    msg = json.loads(msg)  # now msg is a python object which you can use easily to␣
↪access data from.
```

Once you have the message in your callback handler function, you can process it the way you want. print it out, write it to a file, push it to a redis queue, write to a database, offload to a multi-threaded queue. Just whatever.

The default handler for the messages is `_default_on_msg` which does some checks on messages having event as `status`. and prints out other messages. Messages from polygon having the key `ev` equal to `status` are status updates from polygon about login and relevant actions you take (ev indicates event)

The data messages will have different `ev` value than the string 'status'. The ev values for those would match the *polygon.enums.StreamServicePrefix* values.

You can specify your own handlers for other callbacks (`on_error`, `on_close` etc) too or leave those to defaults.

if you choose to override default handlers for `on_error` and `on_close`, here is how they need to be written

`on_error` handler must accept two arguments. You can ignore the first argument which is just the websocket instance itself. The second argument is going to be the actual error

```
def sample_error_handler(ws, error):
    print(error)
```

`on_close` handler must accept three arguments. you can ignore the first arg which is just the websocket instance itself. The second arg is close code, and third would be the close message. note that this handler is only called when the stream is being closed.

```
def sample_close_handler(ws, close_code, close_msg):
    print(f'Stream close with code: {close_code} || msg: {close_msg}')
```

7.3.3 Closing Stream

To turn off the streamer and shut down the websockets connection gracefully, it is advised to call `stream_client.close_stream()` method when closing the application. Not an absolute necessity but a good software practice.

Streams

7.4 Stocks Streams

7.4.1 Stock Trades

`StreamClient.`**`subscribe_stock_trades`**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time trades for given stock ticker symbol(s).

Parameters

- **`symbols`** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_stock_trades`**(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

7.4.2 Stock Quotes

`StreamClient.`**`subscribe_stock_quotes`**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Quotes for given stock ticker symbol(s).

Parameters

- **`symbols`** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_stock_quotes`**(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

7.4.3 Stock Minute Aggregates (OCHLV)

`StreamClient.`**`subscribe_stock_minute_aggregates`**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time minute aggregates for given stock ticker symbol(s).

Parameters

- **`symbols`** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_stock_minute_aggregates`**(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

7.4.4 Stock Second Aggregates (OCHLV)

`StreamClient.`**`subscribe_stock_second_aggregates`**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time second aggregates for given stock ticker symbol(s).

Parameters

- **`symbols`** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_stock_second_aggregates`**(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

7.4.5 Stock Limit Up Limit Down (LULD)

`StreamClient.`**`subscribe_stock_limit_up_limit_down`**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time LULD events for given stock ticker symbol(s).

Parameters

- **`symbols`** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_stock_limit_up_limit_down`**`(`*symbols: Optional[list] = None*`)`

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

7.4.6 Stock Imbalances

`StreamClient.`**`subscribe_stock_imbalances`**`(`*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*`)`

Stream real-time Imbalance Events for given stock ticker symbol(s).

Parameters

- **`symbols`** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_stock_imbalances`**`(`*symbols: Optional[list] = None*`)`

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

7.5 Options Streams

7.5.1 Options Trades

`StreamClient.`**`subscribe_option_trades`**`(`*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*`)`

Stream real-time Options Trades for given Options contract.

Parameters

- **`symbols`** – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`
- **`force_uppercase_symbols`** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`StreamClient.`**`unsubscribe_option_trades`**`(`*symbols: Optional[list] = None*`)`

Unsubscribe real-time Options Trades for given Options contract.

Parameters
`symbols` – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

7.5.2 Options Quotes

StreamClient.**subscribe_option_quotes**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Quotes for given Options contract.

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_option_quotes**(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Quotes for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

7.5.3 Options Minute Aggregates (OCHLV)

StreamClient.**subscribe_option_minute_aggregates**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Minute Aggregates for given Options contract(s).

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL tickers in the market. you can pass **with or without** the prefix `O:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_option_minute_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Minute aggregates for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

7.5.4 Options Second Aggregates (OCHLV)

StreamClient.**subscribe_option_second_aggregates**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Second Aggregates for given Options contract(s).

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL tickers in the market. you can pass **with or without** the prefix `O:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_option_second_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Second Aggregates for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

7.6 Forex Streams

7.6.1 Forex Quotes

StreamClient.**subscribe_forex_quotes**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time forex quotes for given forex pair(s).

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_forex_quotes**(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

Parameters
symbols – A list of forex tickers. Default is * which unsubscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

7.6.2 Forex Minute Aggregates (OCHLV)

StreamClient.**subscribe_forex_minute_aggregates**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time forex Minute Aggregates for given forex pair(s).

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_forex_minute_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

Parameters
symbols – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

7.7 Crypto Streams

7.7.1 Crypto Trades

StreamClient.**subscribe_crypto_trades**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Trades for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_crypto_trades**(*symbols: Optional[list] = None*)

Unsubscribe real-time trades for given cryptocurrency pair(s).

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

7.7.2 Crypto Quotes

StreamClient.**subscribe_crypto_quotes**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Quotes for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_crypto_quotes**(*symbols: Optional[list] = None*)

Unsubscribe real-time quotes for given cryptocurrency pair(s).

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

7.7.3 Crypto Minute Aggregates (OCHLV)

StreamClient.**subscribe_crypto_minute_aggregates**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Minute Aggregates for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_crypto_minute_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe real-time minute aggregates for given cryptocurrency pair(s).

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

7.7.4 Crypto Level 2 Book

StreamClient.**subscribe_crypto_level2_book**(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time level 2 book data for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

StreamClient.**unsubscribe_crypto_level2_book**(*symbols: Optional[list] = None*)

Unsubscribe real-time level 2 book data for given cryptocurrency pair(s).

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

CHAPTER

EIGHT

ASYNC STREAMING

A convenient wrapper around the Streaming API

IMPORTANT Polygon.io allows one simultaneous connection to one cluster at a time (clusters: stocks, options, forex, crypto). which means 4 total concurrent streams (Of course you need to have subscriptions for them).

Connecting to a cluster which already has an existing stream connected to it would result in existing connection getting dropped and new connection would be established

Note that This page describes the asyncio based streaming client. If you're looking for callback based streaming client, See *Callback Streaming*

Also note that async client has a reconnection mechanism built into it already. It is very basic at the moment. It resubscribes to the same set of services it already had before the disconnection and restores the handlers when reconnection establishes. More info in starting the stream below.

It also exposes a few methods which you could use to create your own reconnect mechanism. Method *polygon.streaming.async_streaming.AsyncStreamClient.reconnect()* is one of them

Have a reconnect mechanism to share? Share in discussions or on the wiki.

8.1 Creating the client

Creating a client is just creating an instance of `polygon.AsyncStreamClient`. Note that this expects a few arguments where most of them have default values.

This is how the initializer looks like:

`AsyncStreamClient.__init__`(*api_key: str*, *cluster*, *host='socket.polygon.io'*, *ping_interval: Optional[int] = 20*, *ping_timeout: Optional[int] = 19*, *max_message_size: int = 1048576*, *max_memory_queue: Optional[int] = 32*, *read_limit: int = 65536*, *write_limit: int = 65536*)

Initializes the stream client for async streaming Official Docs

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **cluster** – Which market/cluster to connect to. See *polygon.enums.StreamCluster* for choices. NEVER connect to the same cluster again if there is an existing stream connected to it. The existing connection would be dropped and new one will be established. You can have up to 4 concurrent streams connected to 4 different clusters.
- **host** – Host url to connect to. Default is real time. See *polygon.enums.StreamHost* for choices

- **ping_interval** – Send a ping to server every specified number of seconds to keep the connection alive. Defaults to 20 seconds. Setting to 0 disables pinging.
- **ping_timeout** – The number of seconds to wait after sending a ping for the response (pong). If no response is received from the server in those many seconds, stream is considered dead and exits with code 1011. Defaults to 19 seconds.
- **max_message_size** – The max_size parameter enforces the maximum size for incoming messages in bytes. The default value is 1 MiB (not MB). None disables the limit. If a message larger than the maximum size is received, recv() will raise ConnectionClosedError and the connection will be closed with code 1009
- **max_memory_queue** – sets the maximum length of the queue that holds incoming messages. The default value is 32. None disables the limit. Messages are added to an in-memory queue when they're received; then recv() pops from that queue
- **read_limit** – sets the high-water limit of the buffer for incoming bytes. The low-water limit is half the high-water limit. The default value is 64 KiB, half of asyncio's default. Don't change if you are unsure of what it implies.
- **write_limit** – The write_limit argument sets the high-water limit of the buffer for outgoing bytes. The low-water limit is a quarter of the high-water limit. The default value is 64 KiB, equal to asyncio's default. Don't change if you're unsure what it implies.

Example use:

```
import polygon

stream_client = polygon.AsyncStreamClient('KEY', 'stocks')  # in the simplest form
```

Note that you don't have to call login methods as the library does it internally itself.

8.2 Starting the Stream

Once you have a stream client, you MUST subscribe to streams before you start the main stream loop. Note that you can alter your subscriptions from other coroutines easily even after starting the main stream loop. See subscriptions methods below this section to know how to subscribe to streams.

AFTER you have called your initial subscription methods, you have two ways to start the main stream loop.

8.2.1 Without using the built-in reconnect functionality

In this case you'd need to have your own while loop, like so:

```
# assuming we create the client and sub to stream here already.
while 1:
    await stream_client.handle_messages()
```

and that's basically it. handle_message would take care of receiving messages and calling appropriate handlers (see below section for info on that aspect). You may want to implement your own reconnect mechanism here.

If that's your use case, you can basically ignore the below section completely.

8.2.2 Using the built-in reconnect functionality

here you don't need any outer while loop of your own. The lib has inner while loops and mechanisms to trap disconnection errors and will attempt to reconnect.

Note that this function is basic and not perfect yet and will continue to improve as we move ahead. If you figure out a way to implement reconnection, feel free to share that in discussions or on the wiki.

simple use example

```
# assuming we already have a client subscribed to streams
await stream_client.handle_messages(reconnect=True)
```

That's it. This should be enough for most users. For those who need more control over the behavior here; this is how the method definition looks like:

async AsyncStreamClient.**handle_messages**(*reconnect: bool = False*, *max_reconnection_attempts=5*, *reconnection_delay=5*)

The primary method to start the stream. Connects & Logs in by itself. Allows Reconnecting by simply altering a parameter (subscriptions are persisted across reconnected streams)

Parameters

- **reconnect** – If this is `False` (default), it simply awaits the next message and calls the appropriate handler. Uses the *_default_process_message()* if no handler was specified. You should use the statement inside a while loop in that case. Setting it to True creates an inner loop which traps disconnection errors except login failed due to invalid Key, and reconnects to the stream with the same subscriptions it had earlier before getting disconnected.
- **max_reconnection_attempts** – Determines how many times should the program attempt to reconnect in case of failed attempts. The Counter is reset as soon as a successful connection is re-established. Setting it to False disables the limit which is NOT recommended unless you know you got a situation. This value is ignored if `reconnect` is False (The default). Defaults to 5.
- **reconnection_delay** – Number of seconds to wait before attempting to reconnect after a failed reconnection attempt or a disconnection. This value is ignored if `reconnect` is False (the default). Defaults to 5.

Returns

None

8.3 Subscribing/Unsubscribing to Streams

All subscription methods have names in pattern `subscribe_service_name` and `unsubscribe_service_name`.

Symbols names must be specified as a list of symbols: `['AMD', 'NVDA', 'LOL']` is the correct way to specify symbols. Not specifying a list of symbols results in the action being applied to `ALL` tickers in that service. Note that either of `[]`, `None`, `['*']` or `'all'` as value of symbols would also results in ALL tickers.

The library allows specifying a string for symbol argument (that string is sent exactly as it is without processing), but only do that if you have the absolute need to. Most people should just specify a list. Note that a list of single ticker is accepted.

Options and Crypto stream endpoints expect prefixes ``O:, X:`` respectively in front of every ticker. The library handles this for you so you can pass symbols with or without those prefixes.

The Second argument on all unsubscribe methods is the `handler_function` which represents the handler function you'd like the library to call when a message from that service is received. You can have one handler for multiple services. Not supplying a handler results in the library using the default message handler.

All methods are async coroutines which need to be awaited.

```
await stream_client.subscribe_stock_trades(['AMD', 'NVDA'], handler_function=my_handler_function)
```

By default, the library will also enforce upper case for all symbols being passed. To disable this enforcement, just pass in `force_uppercase_symbols=False` when subscribing in the methods below.

8.4 Handling Messages

your handler functions should accept one argument which indicates the message.

```
async def sample_handler(msg):
    print(f'Look at me! I am the handler now. {msg}')
```

Note that you can also use a sync function as handler

```
def sample_handler(msg):
    print(f'I am also a handler. But sync.. {msg}')
```

In async streaming, **the library does the json decoding for you internally, and you will always receive a list/dict python object** (a list 99.99% of the time except the initial status messages). **You don't have to do** `json decoding` **yourself**. Internally it is already done using `json.loads(msg)`

Once you have the message in your callback handler function, you can process it the way you want. print it out, write it to a file, push it to a redis queue, write to a database, offload to a multi-threaded queue. Just whatever.

The default handler for the messages is `_default_process_message`.

8.5 Changing message handler functions while stream is running

Library allows you to change your handlers after your main stream loop has started running.

The function you'd need is:

async AsyncStreamClient.**change_handler**(*service_prefix*, *handler_function*)

Change your handler function for a service. Can be used to update handlers dynamically while stream is running.

Parameters

- **service_prefix** – The Prefix of the service you want to change handler for. see *polygon.enums.StreamServicePrefix* for choices.
- **handler_function** – The new handler function to assign for this service

Returns

None

Note that you should never need to change handler for `status` (which handles `ev` messages) unless you know you got a situation. Service prefixes just indicate which service (eg stock trades? options aggregates?) you want to change the handler.

8.6 Closing the Stream

To turn off the streamer and shut down the websockets connection gracefully, it is advised to `await stream_client.close_stream()` when closing the application. Not an absolute necessity but a good software practice.

Streams

8.7 Stock Streams

8.7.1 Stock Trades

`async AsyncStreamClient.subscribe_stock_trades(symbols: Optional[list] = None, handler_function=None, force_uppercase_symbols: bool = True)`

Get Real time trades for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL tickers.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`async AsyncStreamClient.unsubscribe_stock_trades(symbols: Optional[list] = None)`

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

8.7.2 Stock Quotes

`async AsyncStreamClient.subscribe_stock_quotes(symbols: Optional[list] = None, handler_function=None, force_uppercase_symbols: bool = True)`

Get Real time quotes for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL tickers.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_stock_quotes**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

8.7.3 Stock Minute Aggregates (OCHLV)

async AsyncStreamClient.**subscribe_stock_minute_aggregates**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Minute Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_stock_minute_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

8.7.4 Stock Second Aggregates (OCHLV)

async AsyncStreamClient.**subscribe_stock_second_aggregates**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Seconds Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_stock_second_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

8.7.5 Stock Limit Up Limit Down (LULD)

async AsyncStreamClient.**subscribe_stock_limit_up_limit_down**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time LULD Events for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_stock_limit_up_limit_down**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

8.7.6 Stock Imbalances

async AsyncStreamClient.**subscribe_stock_imbalances**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Imbalance Events for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_stock_imbalances**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

8.8 Options Streams

8.8.1 Options Trades

async AsyncStreamClient.**subscribe_option_trades**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options trades for provided ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_option_trades**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

8.8.2 Options Quotes

async AsyncStreamClient.**subscribe_option_quotes**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options quotes for provided ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`

- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`async AsyncStreamClient.unsubscribe_option_quotes(symbols: Optional[list] = None)`

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

8.8.3 Options Minute Aggregates (OCHLV)

`async AsyncStreamClient.subscribe_option_minute_aggregates(symbols: Optional[list] = None, handler_function=None, force_uppercase_symbols: bool = True)`

Get Real time options minute aggregates for given ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`async AsyncStreamClient.unsubscribe_option_minute_aggregates(symbols: Optional[list] = None)`

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

8.8.4 Options Second Aggregates (OCHLV)

async AsyncStreamClient.**subscribe_option_second_aggregates**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options second aggregates for given ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_option_second_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

8.9 Forex Streams

8.9.1 Forex Quotes

async AsyncStreamClient.**subscribe_forex_quotes**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Forex Quotes for provided symbol(s)

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_forex_quotes**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied forex symbols.

Parameters
symbols – A list of forex tickers. Default is * which unsubscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

Returns
None

8.9.2 Forex Minute Aggregates (OCHLV)

`async AsyncStreamClient.subscribe_forex_minute_aggregates(symbols: Optional[list] = None, handler_function=None, force_uppercase_symbols: bool = True)`

Get Real time Forex Minute Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`async AsyncStreamClient.unsubscribe_forex_minute_aggregates(symbols: Optional[list] = None)`

Unsubscribe from the stream for the supplied forex symbols.

Parameters
symbols – A list of forex tickers. Default is * which unsubscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

Returns
None

8.10 Crypto Streams

8.10.1 Crypto Trades

`async AsyncStreamClient.subscribe_crypto_trades(symbols: Optional[list] = None, handler_function=None, force_uppercase_symbols: bool = True)`

Get Real time Crypto Trades for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.

- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_crypto_trades**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

8.10.2 Crypto Quotes

async AsyncStreamClient.**subscribe_crypto_quotes**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Quotes for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_crypto_quotes**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

8.10.3 Crypto Minute Aggregates (OCHLV)

async AsyncStreamClient.**subscribe_crypto_minute_aggregates**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Minute Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_crypto_minute_aggregates**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

8.10.4 Crypto Level 2 Book

async AsyncStreamClient.**subscribe_crypto_level2_book**(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Level 2 Book Data for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async AsyncStreamClient.**unsubscribe_crypto_level2_book**(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters

symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns

None

CHAPTER

NINE

WHAT THE HELL ARE ENUMS ANYWAYS

Sooooo… you've had enough of these `enums` and finally decided to know what the hell they actually are and why you should care about them.

Well read this page to get your answers.

You should have seen them on many methods' documentation as argument choices.

First up, does everyone need them? that depends on their use case. enums in this library are only used on some endpoints, especially the ones in reference APIs and some basic uses in stream clients. So if someone only needs to ochlv chart data, they probably won't need to use enums.

If you notice any value which is supported by the API but not included in the enums, Let me Know using discussions

9.1 What are they

Simplest non technical terms definition
: They are a way to define pseudo constants (read constants) in python (python doesn't have anything as constants. That's why enums are precious :D). They have many use cases other than constants but for this library you only need to know this far.

For example
: consider the enum *`polygon.enums.AssetClass`* which has 4 values inside of it. The values are just class attribute and you can access them just like you'd access any other class attribute. `print(polygon.enums.AssetClass.STOCKS)` would print the string `stocks`. so in another words this enum class has 4 member enums which can be used to specify the value wherever needed. Like this `some_function(arg1, asset=AssetClass.STOCKS)`.

when you pass in an enum to a function or a method, it is equal to passing in the value of that enum.

so instead of `some_function(arg1, asset=AssetClass.STOCKS)` i could have said `some_function(arg1, asset='stocks')` and both mean the same thing.

Here are All the enums of this library in one place

9.2 Then why not just pass in raw values? Why do we need enums?

I mean you could do that. In fact many people would still do that despite the notes here (I'll be watching you all :/).

but think about it this way, can you have enums for a parameter which expects a person's name? Of course not. Because there isn't any constant value (or a fixed set of values) to choose from.

but can i have enums for TickerTypes? Yes. Because it has a set of fixed values and the API would not return the correct data if the value passed in is different than the ones which are in the fixed set.

Using enums

- Avoids passing in incorrect values.
- Avoids typing mistakes while passing in parameter values (I'm looking at you `TRAILING_TWELVE_MONTHS_ANNUALIZED`)
- gives you a fixed set of values to choose from and you don't have to hit and trial to know supported values.
- And finally, IDE autocomplete would make your life even easier while writing code that makes use of enums

Finally, it's not an absolute necessity to use enums but they are very much recommended.

9.3 Okay how do I use them

To start off, like any other name, you'd need to import the names. Now there are many ways to do that and it's up to your coding preferences. Make use of your IDE auto-completions to make it easier to fill in enums.

Some common ways are

9.3.1 Approach 1 - importing all enums at once

```
import polygon  # which you already do for using other clients so nothing new to import␣
↪here

# now you can use enums as

client.some_function(other_args, arg=polygon.enums.TickerType.ADRC)

# OR
import polygon.enums as enums

client.some_function(other_args, arg=enums.TickerType.ETF)
```

as you see this allows you to access all enums without having to import each one individually. But this also mean you'd be typing longer names (not big of an issue considering IDE completions).

Note that importing all enums doesn't have any resource overhead so don't worry about enums eating your RAM.

9.3.2 Approach 2 - importing just the enums you need

This approach is nicer for cases when you only specifically need a few enums.

```
from polygon.enums import TickerType

# using it as
client.some_function(other_args, arg=TickerType.CS)

# OR
from polygon.enums import (TickerType, AssetClass)

client.some_function(other_args, arg=TickerType.CS)

client.some_other_function(other_args, arg=TickerType.CS, other_arg=AssetClass.STOCKS)
```

9.3.3 Other Approaches

You could use any other import syntax if you like. such as `from polygon.enums import *` but I wouldn't recommend wild card imports.

CHAPTER

TEN

GETTING HELP

Generally, feel free to join our Discord Server for help/discussions.

If you're stuck at something. don't worry, everyone does. Need a hand? Here is how you can get help.

- See if you can find the relevant info in *FAQs* or *Community Wikis*
- See if there is an Open Issue or a Pull Request related to your concern already.
- See if your issue has been discussed already in one of the *Discussions*
- If you believe the issue could be on polygon.io end, get in touch with their support team. They're quite helpful. There is a button in bottom right corner of every documentation page

Once you have gone through these and haven't found your answer, you can

- Join our Discord Server and ask your question/discuss or chat with people.
- Start a Discussion. You can ask your questions in general channel or create a QnA discussion from left.

If your question is more of a bug report, you can raise a new issue or feature request with adequate information.

Remember that Issues is not a good place to ask for general help.

Always make sure to provide enough information when asking for help. This includes but not limited to

- Your Operating system (Ubuntu? Arch? Windows?)
- Your execution environment (Pycharm? VSC? A usual terminal? a cloud instance? a rasp pi?)
- Your python version and `polygon` version. always ensure you are on the latest version of the library. You can update if you're not using command `pip install --upgrade polygon`
- The full stack traceback and error message if any. **Do not attempt to describe error messages in your own languages. Sometimes error messages don't mean what they say**
- The source code which causes the error. **If your code is supposed to be secret, write a sample script which can reproduce the issue. Always make sure to remove sensitive info from logs/code**

CHAPTER

ELEVEN

BUGS, DISCUSSIONS, WIKIS, FAQS

This section provides info on Issues tracker, Discussions functionality, community wikis and FAQs.

11.1 Bug Reports or Feature Requests

Got a bug/report to report or a feature request? You're in the right place.

Before submitting, make sure you have enough information to provide. It is advised to follow the provided template but feel free to use your own. Just ensure you provide the following info:

- Your Operating system (Linux? Windows?)
- Your execution environment (Pycharm? VSC? A usual terminal? a cloud instance? a rasp pi?)
- Your python version and `polygon` version. always ensure you are on the latest version of the library. You can update if you're not using command `pip install --upgrade polygon`
- The full stack traceback and error message if any. Do not attempt to describe error messages in your own languages. Sometimes messages don't mean what they say
- The code which causes the error. If your code is supposed to be secret, write a sample script which can reproduce the issue. Always make sure to remove sensitive info from logs/code

In case of feature requests, describe what functionality would you like to be added to the library.

Open issues/feature requests here

11.2 Discussions

Discussions are meant to be a place for discussing general stuff which is not worth having an open issue for.

there are two discussion channels by default, one meant for everyone and other meant for contributors/developers

while it is possible to create your own discussions, it is preferred to keep it to those two channels unless needed.

11.3 Community Wikis

The community wiki is a place for everything which the community finds useful for others but isn't in the documentation. every article is just a title and the description text. written in good old markdown. You can write plain text too if you're unsure of what markdown is.

Figured out how to achieve a specific task? Found something interesting? share it with the community by creating a wiki page. Every contribution is significant so don't hesitate.

Read the wiki articles, you may find your answers there.

11.4 FAQs

This is a handpicked collection of common questions and answers about the lib and endpoints in general. A must read if you're looking for answers.

FAQs are added here as soon I have any solid conclusions about a useful question.

Feel free to join our Discord Server if you suggestions for questions to add. You don't necessarily need to know the answer :D

CHAPTER

TWELVE

CONTRIBUTING AND LICENSE

12.1 Contributing to the library

A bug you can fix? Improving documentation? Just wanna structure the code better? Every improvement matters.

Read this small guide to know how you can start contributing.

If this is your first time contributing to an open source project, Welcome. You'd probably want to contribute to something you are confident about

Want to discuss anything related to the lib? head over to Developer Discussions. You may also use discussions to ask anything related to contributions or library in general.

12.1.1 Picking up what to work on

If you already know what you're going to work on, Great! If you don't or just wanna explore the options; below are the places to look at:

1. Take a look at open issues and see which ones you can work on.
2. Anything which could be improved in the documentation or readme ?
3. Any new endpoints introduced by polygon.io which are not in the library?
4. Any changes to endpoints which are already in the lib but not adjusted according to the new changes?

Once you know what to work on, you can proceed with setting up your environment.

12.1.2 Setting Up the Development Environment

May not be needed for documentation improvements.

Dependencies are listed in requirements.txt. The list has `sphinx` and `sphinx_rtd_theme` which are only meant to build documentation.

It is highly recommended to install the dependencies in a virtual environment to avoid messing with your global interpreter.

```
pip install virtualenv
virtualenv venv
. venv/bin/activate
```

The last instruction above is for *nix machines. For windows `.\venv\Scripts\activate.bat` (or similar) is used

Install the requirements using

```
pip install -r requirements.txt
```

Now you can make your changes

12.1.3 Testing your changes

Currently the project uses the actual endpoints to perform tests (Suggestions/PRs for better testing mechanism are welcome)

All test files are under directory `tests`. You'd need a valid polygon API key to perform the tests as they are right now. If you don't have a subscription, just make the changes, test them the way you like and raise the PR. I'll test the changes before merging.

However if you made changes to the documentation, run the below commands to build locally and test the documentation

```
cd docs
make html
```

The built docs would be placed under `docs/_build/_html`. Open `index.html` here in a browser and see your changes. When you're happy with them, raise the PR.

Remember to document your changes like this library does already.

12.2 License

Don't kid yourself. You don't care what license does the project use, do you? Anyways the project is licensed under MIT License. See License for more details.

CHAPTER

THIRTEEN

LIBRARY INTERFACE DOCUMENTATION

Here is the Entire Library Interface reference.

13.1 Base Clients

13.1.1 Base Client

class polygon.base_client.**Base**

split_date_range(*start*, *end*, *timespan: str*, *high_volatility: bool = False*, *reverse: bool = True*) → list

Internal helper function to split a BIGGER date range into smaller chunks to be able to easily fetch aggregate bars data. The chunks duration is supposed to be different for time spans. For 1 minute bars, multiplier would be 1, timespan would be 'minute'

Parameters

- **start** – start of the time frame. accepts date, datetime objects or a string `YYYY-MM-DD`
- **end** – end of the time frame. accepts date, datetime objects or a string `YYYY-MM-DD`
- **timespan** – The frequency type. like day or minute. see *polygon.enums.Timespan* for choices
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile. If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **reverse** – If True (the default), will reverse the order of chunks (chronologically)

Returns

a list of tuples. each tuple is in format `(start, end)` and represents one chunk of time frame

static normalize_datetime(*dt*, *output_type: str = 'ts'*, *_dir: str = 'start'*, *_format: str = '%Y-%m-%d'*, *unit: str = 'ms'*)

a core method to perform some specific datetime operations before/after interaction with the API

Parameters

- **dt** – The datetime input
- **output_type** – what to return. defaults to timestamp (utc if unaware obj)
- **_dir** – whether the input is meant for start of a range or end of it
- **_format** – The format string to use IFF expected to return as string
- **unit** – the timestamp units to work with. defaults to ms (milliseconds)

Returns
The output timestamp or formatted string

`static _change_enum(val: ~typing.Union[str, ~enum.Enum, float, int], allowed_type=<class 'str'>)`

13.1.2 Base Sync Client

`class polygon.base_client.BaseClient(api_key: str, connect_timeout: int = 10, read_timeout: int = 10)`

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This is the **base client class** for all other REST clients which inherit from this class and implement their own endpoints on top of it.

`__init__(api_key: str, connect_timeout: int = 10, read_timeout: int = 10)`

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.

`close()`

Closes the `requests.Session` and frees up resources. It is recommended to call this method in your exit handlers

`_get_response(path: str, params: Optional[dict] = None, raw_response: bool = True) → Union[Response, dict]`

Get response on a path. Meant to be used internally but can be used if you know what you're doing

Parameters

- **path** – RESTful path for the endpoint. Available on the docs for the endpoint right above its name.
- **params** – Query Parameters to be supplied with the request. These are mapped 1:1 with the endpoint.
- **raw_response** – Whether to return the `Response` Object. Useful for when you need to check the status code or inspect the headers. Defaults to True which returns the `Response` object.

Returns
A Response object by default. Make `raw_response=False` to get JSON decoded Dictionary

`get_page_by_url(url: str, raw_response: bool = False) → Union[Response, dict]`

Get the next page of a response. The URl is returned within `next_url` attribute on endpoints which support pagination (eg the tickers endpoint). If the response doesn't contain this attribute, either all pages were received or the endpoint doesn't have pagination. Meant for internal use primarily.

Parameters

- **url** – The next URL. As contained in `next_url` of the response.

- **raw_response** – Whether to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
Either a Dictionary or a Response object depending on value of raw_response. Defaults to Dict.

get_next_page(*old_response: Union[Response, dict]*, *raw_response: bool = False*) → Union[Response, dict, bool]

Get the next page using the most recent old response. This function simply parses the next_url attribute from the existing response and uses it to get the next page. Returns False if there is no next page remaining (which implies that you have reached the end of all pages or the endpoint doesn't support pagination).

Parameters

- **old_response** – The most recent existing response. Can be either `Response` Object or Dictionaries
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_previous_page(*old_response: Union[Response, dict]*, *raw_response: bool = False*) → Union[Response, dict, bool]

Get the previous page using the most recent old response. This function simply parses the previous_url attribute from the existing response and uses it to get the previous page. Returns False if there is no previous page remaining (which implies that you have reached the start of all pages or the endpoint doesn't support pagination).

Parameters

- **old_response** – The most recent existing response. Can be either `Response` Object or Dictionaries
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_all_pages(*old_response*, *max_pages: Optional[int] = None*, *direction: str = 'next'*, *verbose: bool = False*, *raw_responses: bool = False*)

A helper function for endpoints which implement pagination using `next_url` and `previous_url` attributes. Can be used externally too to get all responses in a list.

Parameters

- **old_response** – The last response you had. In most cases, this would be simply the very first response.
- **max_pages** – If you want to limit the number of pages to retrieve. Defaults to None which fetches ALL available pages

- **direction** – The direction to paginate in. Defaults to next which grabs all next_pages. see *polygon.enums.PaginationDirection* for choices
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_responses** – If set to True, the elements in container list, you will get underlying Response object instead of the json formatted dict/list. Only use if you need to check status codes or headers. Defaults to False, which makes it return decoded data in list.

Returns

A list of responses. By default, responses are actual json decoded dict/list. Depending on value of `raw_response`

_paginate(*_res*, *merge_all_pages: bool = True*, *max_pages: Optional[int] = None*, *verbose: bool = False*, *raw_page_responses: bool = False*)

Internal function to call the core pagination methods to build the response object to be parsed by individual methods.

Parameters

- **merge_all_pages** – whether to merge all the pages into one response. defaults to True
- **max_pages** – number of pages to fetch. defaults to all available pages.
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – whether to keep raw response objects or decode them. Only considered if merge_all_pages is set to False. Defaults to False.

Returns

get_full_range_aggregates(*fn*, *symbol: str*, *time_chunks: list*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*) → list

Internal helper function to fetch aggregate bars for BIGGER time ranges. Should only be used internally. Users should prefer the relevant aggregate function with additional parameters.

Parameters

- **fn** – The method to call in each chunked timeframe
- **symbol** – The ticker symbol to get data for
- **time_chunks** – The list of time chunks as returned by method `split_datetime_range`
- **run_parallel** – If true (the default), it will use an internal `ThreadPool` to get the responses in parallel. **Note That** since python has the GIL restrictions, it would mean that if you have a ThreadPool of your own, only one ThreadPool will be running at a time and the other pool will wait. set to False to get all responses in sequence (will take time)
- **warnings** – Defaults to True which prints warnings. Set to False to disable warnings.
- **max_concurrent_workers** – This is only used if run_parallel is set to true. Controls how many worker threads are spawned in the internal thread pool. Defaults to `your cpu core count * 5`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.

- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. asc default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to day

Returns
A single merged list of ALL candles/bars

13.1.3 Base Async Client

class polygon.base_client.**BaseAsyncClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This is the **base async client class** for all other REST clients which inherit from this class and implement their own endpoints on top of it.

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a ConnectTimeout if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be received. Raises a ReadTimeout if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a WriteTimeout if unable to connect within the specified time limit.

async static aw_task(*aw*, *semaphore*)

async close()

Closes the httpx.AsyncClient and frees up resources. It is recommended to call this method in your exit handlers. This method should be awaited as this is a coroutine.

async _get_response(*path: str*, *params: Optional[dict] = None*, *raw_response: bool = True*) → Union[Response, dict]

Get response on a path - meant to be used internally but can be used if you know what you're doing

Parameters

- **path** – RESTful path for the endpoint. Available on the docs for the endpoint right above its name.
- **params** – Query Parameters to be supplied with the request. These are mapped 1:1 with the endpoint.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to check the status code or inspect the headers. Defaults to True which returns the Response object.

Returns

A Response object by default. Make raw_response=False to get JSON decoded Dictionary

async get_page_by_url(*url: str*, *raw_response: bool = False*) → Union[Response, dict]

Get the next page of a response. The URl is returned within next_url attribute on endpoints which support pagination (eg the tickers' endpoint). If the response doesn't contain this attribute, either all pages were received or the endpoint doesn't have pagination. Meant for internal use primarily.

Parameters

- **url** – The next URL. As contained in next_url of the response.
- **raw_response** – Whether to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of raw_response. Defaults to Dict.

async get_next_page(*old_response: Union[Response, dict]*, *raw_response: bool = False*) → Union[Response, dict, bool]

Get the next page using the most recent old response. This function simply parses the next_url attribute from the existing response and uses it to get the next page. Returns False if there is no next page remaining (which implies that you have reached the end of all pages or the endpoint doesn't support pagination) - Async method

Parameters

- **old_response** – The most recent existing response. Can be either Response Object or Dictionaries
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

async get_previous_page(*old_response: Union[Response, dict]*, *raw_response: bool = False*) → Union[Response, dict, bool]

Get the previous page using the most recent old response. This function simply parses the previous_url attribute from the existing response and uses it to get the previous page. Returns False if there is no previous page remaining (which implies that you have reached the start of all pages or the endpoint doesn't support pagination) - Async method

Parameters

- **old_response** – The most recent existing response. Can be either `Response` Object or Dictionaries
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_all_pages(*old_response*, *max_pages: Optional[int] = None*, *direction: str = 'next'*, *verbose: bool = False*, *raw_responses: bool = False*)

A helper function for endpoints which implement pagination using `next_url` and `previous_url` attributes. Can be used externally too to get all responses in a list.

Parameters

- **old_response** – The last response you had. In most cases, this would be simply the very first response.
- **max_pages** – If you want to limit the number of pages to retrieve. Defaults to None which fetches ALL available pages
- **direction** – The direction to paginate in. Defaults to next which grabs all next_pages. see *polygon.enums.PaginationDirection* for choices
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_responses** – If set to True, the elements in container list, you will get underlying Response object instead of the json formatted dict/list. Only use if you need to check status codes or headers. Defaults to False, which makes it return decoded data in list.

Returns

A list of responses. By default, responses are actual json decoded dict/list. Depending on value of `raw_response`

async _paginate(*_res*, *merge_all_pages: bool = True*, *max_pages: Optional[int] = None*, *verbose: bool = False*, *raw_page_responses: bool = False*)

Internal function to call the core pagination methods to build the response object to be parsed by individual methods.

Parameters

- **merge_all_pages** – whether to merge all the pages into one response. defaults to True
- **max_pages** – number of pages to fetch. defaults to all available pages.
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.

- **raw_page_responses** – whether to keep raw response objects or decode them. Only considered if merge_all_pages is set to False. Defaults to False.

Returns

async get_full_range_aggregates(*fn*, *symbol: str*, *time_chunks: list*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*) → list

Internal helper function to fetch aggregate bars for BIGGER time ranges. Should only be used internally. Users should prefer the relevant aggregate function with additional parameters.

Parameters

- **fn** – The method to call in each chunked timeframe
- **symbol** – The ticker symbol to get data for
- **time_chunks** – The list of time chunks as returned by method `split_datetime_range`
- **run_parallel** – If true (the default), it will use an internal `ThreadPool` to get the responses in parallel. **Note That** since python has the GIL restrictions, it would mean that if you have a ThreadPool of your own, only one ThreadPool will be running at a time and the other pool will wait. set to False to get all responses in sequence (will take time)
- **warnings** – Defaults to True which prints warnings. Set to False to disable warnings.
- **max_concurrent_workers** – This is only used if run_parallel is set to true. Controls how many worker coroutines are spawned internally. Defaults to `your cpu core count * 5`. An `asyncio.Semaphore()` is used behind the scenes.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. `asc` default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to `day`

Returns

A single merged list of ALL candles/bars

13.2 Stocks Clients

13.2.1 Stocks Sync Client

class polygon.stocks.stocks.**SyncStocksClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the Stocks REST endpoints. Note that you should always import names from top level. eg: `from polygon import StocksClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.

get_trades(*symbol: str*, *date*, *timestamp: Optional[int] = None*, *timestamp_limit: Optional[int] = None*, *reverse: bool = True*, *limit: int = 5000*, *raw_response: bool = False*)

Get trades for a given ticker symbol on a specified date. The response from polygon seems to have a `map` attribute which gives a mapping of attribute names to readable values. Official Docs

Parameters

- **symbol** – The ticker symbol we want trades for.
- **date** – The date/day of the trades to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **timestamp** – The timestamp offset, used for pagination. Timestamp is the offset at which to start the results. Using the `timestamp` of the last result as the offset will give you the next page of results. Default: None. I'm trying to think of a good way to implement pagination support for this type of pagination.
- **timestamp_limit** – The maximum timestamp allowed in the results. Default: None
- **reverse** – Reverse the order of the results. Default True: oldest first. Make it False for Newest first
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_trades_v3(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for a ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want trades for.
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp

- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.StocksTradesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

get_quotes(*symbol: str*, *date*, *timestamp: Optional[int] = None*, *timestamp_limit: Optional[int] = None*, *reverse: bool = True*, *limit: int = 5000*, *raw_response: bool = False*)

Get Quotes for a given ticker symbol on a specified date. The response from polygon seems to have a `map` attribute which gives a mapping of attribute names to readable values. Official Docs

Parameters

- **symbol** – The ticker symbol we want quotes for.
- **date** – The date/day of the quotes to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **timestamp** – The timestamp offset, used for pagination. Timestamp is the offset at which to start the results. Using the `timestamp` of the last result as the offset will give you the

next page of results. Default: None. Thinking of a good way to implement this pagination here.

- **timestamp_limit** – The maximum timestamp allowed in the results. Default: None
- **reverse** – Reverse the order of the results. Default True: oldest first. Make it False for Newest first
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_quotes_v3(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get NBBO Quotes for a ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want quotes for.
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.StocksQuotesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True

- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_last_trade`(*symbol: str*, *raw_response: bool = False*)

Get the most recent trade for a given stock. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_last_quote`(*symbol: str*, *raw_response: bool = False*)

Get the most recent NBBO (Quote) tick for a given stock. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_daily_open_close`(*symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the OCHLV and after-hours prices of a stock symbol on a certain date. Official Docs

Parameters

- **symbol** – The ticker symbol we want daily-OCHLV for.
- **date** – The date/day of the daily-OCHLV to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_aggregate_bars`(*symbol: str*, *from_date*, *to_date*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a stock over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **from_date** – The start of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. `asc` default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to `day`
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

`get_grouped_daily_bars`(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily OCHLV for the entire stocks/equities markets. Official docs

Parameters

- **date** – The date to get the data for. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_previous_close`(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's OCHLV for the specified stock ticker. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_snapshot`(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded stock ticker. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_current_price`(*symbol: str*) → float

get current market price for the ticker symbol specified.

Uses *`get_last_trade()`* under the hood Official Docs

Parameters

symbol – The ticker symbol of the stock/equity.

Returns

The current price. A `KeyError` indicates the request wasn't successful.

get_snapshot_all(*symbols: Optional[list] = None*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded stock symbols. Official Docs

Parameters

- **symbols** – A comma separated list of tickers to get snapshots for. Defaults to ALL tickers
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_gainers_and_losers(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in stocks/equities markets. Official Docs

Parameters

- **direction** – The direction of results. Defaults to gainers. See *polygon.enums.SnapshotDirection* for choices
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.2.2 Stocks Async Client

class polygon.stocks.stocks.**AsyncStocksClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the Stocks REST endpoints. Note that you should always import names from top level. eg: `from polygon import StocksClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.

- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

async get_trades(*symbol: str*, *date*, *timestamp: Optional[int] = None*, *timestamp_limit: Optional[int] = None*, *reverse: bool = True*, *limit: int = 5000*, *raw_response: bool = False*)

Get trades for a given ticker symbol on a specified date. The response from polygon seems to have a `map` attribute which gives a mapping of attribute names to readable values - Async method Official Docs

Parameters

- **symbol** – The ticker symbol we want trades for.
- **date** – The date/day of the trades to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **timestamp** – The timestamp offset, used for pagination. Timestamp is the offset at which to start the results. Using the `timestamp` of the last result as the offset will give you the next page of results. Default: None. I'm trying to think of a good way to implement pagination support for this type of pagination.
- **timestamp_limit** – The maximum timestamp allowed in the results. Default: None
- **reverse** – Reverse the order of the results. Default True: oldest first. Make it False for Newest first
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_trades_v3(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for a ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want trades for.

- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.StocksTradesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_quotes(*symbol: str*, *date*, *timestamp: Optional[int] = None*, *timestamp_limit: Optional[int] = None*, *reverse: bool = True*, *limit: int = 5000*, *raw_response: bool = False*)

Get Quotes for a given ticker symbol on a specified date. The response from polygon seems to have a `map` attribute which gives a mapping of attribute names to readable values - Async method Official Docs

Parameters

- **symbol** – The ticker symbol we want quotes for.
- **date** – The date/day of the quotes to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`

- **timestamp** – The timestamp offset, used for pagination. Timestamp is the offset at which to start the results. Using the `timestamp` of the last result as the offset will give you the next page of results. Default: None. Thinking of a good way to implement this pagination here.
- **timestamp_limit** – The maximum timestamp allowed in the results. Default: None
- **reverse** – Reverse the order of the results. Default True: oldest first. Make it False for Newest first
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_quotes_v3(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get NBBO Quotes for a ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want quotes for.
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.StocksQuotesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True

- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_last_trade(*symbol: str*, *raw_response: bool = False*)

Get the most recent trade for a given stock - Async method Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_last_quote(*symbol: str*, *raw_response: bool = False*)

Get the most recent NBBO (Quote) tick for a given stock - Async method Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_daily_open_close(*symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the OCHLV and after-hours prices of a stock symbol on a certain date - Async method Official Docs

Parameters

- **symbol** – The ticker symbol we want daily-OCHLV for.
- **date** – The date/day of the daily-OCHLV to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_aggregate_bars`(*symbol: str*, *from_date*, *to_date*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a stock over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **from_date** – The start of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. `asc` default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to `day`
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

`async get_grouped_daily_bars`(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily OCHLV for the entire stocks/equities markets - Async method Official docs

Parameters

- **date** – The date to get the data for. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_previous_close`(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's OCHLV for the specified stock ticker - Async method Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_snapshot`(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded stock ticker - Async method Official Docs

Parameters

- **symbol** – The ticker symbol of the stock/equity.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_current_price`(*symbol: str*) → float

get current market price for the ticker symbol specified - Async method

Uses *get_last_trade()* under the hood Official Docs

Parameters
symbol – The ticker symbol of the stock/equity.

Returns
The current price. A `KeyError` indicates the request wasn't successful.

`async get_snapshot_all`(*symbols: Optional[list] = None*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded stock symbols - Async method Official Docs

Parameters

- **symbols** – A comma separated list of tickers to get snapshots for. Defaults to ALL tickers
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_gainers_and_losers`(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in stocks/equities markets - Async method Official Docs

Parameters

- **direction** – The direction of results. Defaults to gainers. See *polygon.enums.SnapshotDirection* for choices
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.3 Options Clients

13.3.1 Option Symbol Helper Functions & Objects

`polygon.options.options.`**build_option_symbol**(*underlying_symbol: str*, *expiry*, *call_or_put*, *strike_price*, *_format='polygon'*, *prefix_o: bool = False*) → str

Generic function to build option symbols for ALL supported formats: *polygon.enums.OptionSymbolFormat*. Default format is `polygon`.

Parameters

- **underlying_symbol** – The underlying stock ticker symbol.
- **expiry** – The expiry date for the option. You can pass this argument as `datetime.datetime` or `datetime.date` object. Or a string in format: `YYMMDD`. Using datetime objects is recommended.
- **call_or_put** – The option type. You can specify: `c` or `call` or `p` or `put`. Capital letters are also supported.
- **strike_price** – The strike price for the option. ALWAYS pass this as one number. `145`, `240.5`, `15.003`, `56`, `129.02` are all valid values. Try to keep up to 3 digits after the decimal point

- **_format** – The format to use when building symbol. Defaults to `polygon`. Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*
- **prefix_o** – Whether to prefix the symbol with `O:`. It is needed by polygon endpoints. However, all the library functions will automatically add this prefix if you pass in symbols without this prefix. This parameter is **ignored** if format is not `polygon`

Returns
The option symbols string in the format specified

polygon.options.options.**parse_option_symbol**(*option_symbol: str*, *_format='polygon'*, *output_format='object'*)

Generic function to build option symbols for ALL supported formats: *polygon.enums.OptionSymbolFormat*. Default format is `polygon`.

Parameters

- **option_symbol** – the option symbol you want to parse
- **_format** – What format the symbol is in. If you don't know the format you can use the `detect_option_symbol_format` function to detect the format (best effort detection). Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*. Default: `polygon`
- **output_format** – Output format of the result. defaults to object. Set it to `dict` or `list` as needed.

Returns
The parsed info from symbol either as an object, list or a dict as indicated by `output_format`.

polygon.options.options.**build_polygon_option_symbol**(*underlying_symbol: str*, *expiry*, *call_or_put*, *strike_price*, *prefix_o: bool = False*) → str

Build the option symbol from the details provided, in standard polygon format

Parameters

- **underlying_symbol** – The underlying stock ticker symbol.
- **expiry** – The expiry date for the option. You can pass this argument as `datetime.datetime` or `datetime.date` object. Or a string in format: `YYMMDD`. Using datetime objects is recommended.
- **call_or_put** – The option type. You can specify: `c` or `call` or `p` or `put`. Capital letters are also supported.
- **strike_price** – The strike price for the option. ALWAYS pass this as one number. `145`, `240.5`, `15.003`, `56`, `129.02` are all valid values. It shouldn't have more than three numbers after decimal point.
- **prefix_o** – Whether to prefix the symbol with `O:`. It is needed by polygon endpoints. However, all the library functions will automatically add this prefix if you pass in symbols without this prefix.

Returns
The option symbol in the format specified by polygon

polygon.options.options.**parse_polygon_option_symbol**(*option_symbol: str*, *output_format='object'*)

Function to parse an option symbol in standard polygon format

Parameters

- **option_symbol** – the symbol you want to parse. Both `TSLA211015P125000` and `O:TSLA211015P125000` are valid
- **output_format** – Output format of the result. defaults to object. Set it to `dict` or `list` as needed.

Returns
The parsed values either as an object, list or a dict as indicated by `output_format`.

`polygon.options.options.`**convert_option_symbol_formats**(*option_symbol: str*, *from_format: str*, *to_format: str*) → str

Convert an option symbol from one format to another within supported formats: *polygon.enums.OptionSymbolFormat*

Parameters

- **option_symbol** – The option symbol you want to convert
- **from_format** – The format in which the option symbol is currently in. If you don't know the format you can use the `detect_option_symbol_format` function to detect the format (best effort detection). Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*
- **to_format** – The format to which you want to convert the option symbol. Supported formats are `polygon, tda, tos, ibkr, tradier, trade_station`. If you prefer to use convenient enums, see *polygon.enums.OptionSymbolFormat*

Returns
The converted option symbol as a string

`polygon.options.options.`**detect_option_symbol_format**(*option_symbol: str*) → Union[str, bool, list]

Detect what format a symbol is formed in. Supported formats are *polygon.enums.OptionSymbolFormat*. This function does basic detection according to some simple rules. Test well before using in production.

Parameters
option_symbol – The option symbol to check the format of

Returns
Format's shorthand string or list of strings if able to recognize the format. `False` otherwise. Possible shorthand strings are `polygon, tda, tos, ibkr, tradier, trade_station`

`polygon.options.options.`**ensure_prefix**(*symbol: str*)

Ensure that the option symbol has the prefix `O:` as needed by polygon endpoints. If it does, make no changes. If it doesn't, add the prefix and return the new value.

Parameters
symbol – the option symbol to check

class `polygon.options.options.`**OptionSymbol**(*option_symbol: str*, *symbol_format='polygon'*)

The custom object for parsed details from option symbols.

__init__(*option_symbol: str*, *symbol_format='polygon'*)

Parses the details from symbol and creates attributes for the object.

Parameters

- **option_symbol** – the symbol you want to parse. Both `TSLA211015P125000` and `O:TSLA211015P125000` are valid
- **symbol_format** – Which formatting spec to use. Defaults to polygon. also supports `tda` which is the format supported by TD Ameritrade

__repr__()

Return repr(self).

13.3.2 Options Sync Client

class polygon.options.options.**SyncOptionsClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the Options REST endpoints. Note that you should always import names from top level. eg: `from polygon import OptionsClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.

get_trades(*option_symbol: str*, *timestamp=None*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *sort='timestamp'*, *limit: int = 5000*, *order='asc'*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for an options ticker symbol in a given time range. Note that you need to have an option symbol in correct format for this endpoint. You can use `ReferenceClient.get_option_contracts` to query option contracts using many filter parameters such as underlying symbol etc. Official Docs

Parameters

- **option_symbol** – The options ticker symbol to get trades for. for eg `O:TSLA210903C00700000`. you can pass the symbol with or without the prefix `O:`
- **timestamp** – Query by trade timestamp. You can supply a `date`, `datetime` object or a `nanosecond UNIX timestamp` or a string in format: `YYYY-MM-DD`.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp

- **sort** – Sort field used for ordering. Defaults to timestamp. See *polygon.enums.OptionTradesSort* for available choices.
- **limit** – Limit the number of results returned. Defaults to 5000. max is 50000.
- **order** – order of the results. Defaults to `asc`. See *polygon.enums.SortOrder* for info and available choices.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_quotes`(*option_symbol: str*, *timestamp=None*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *sort='timestamp'*, *limit: int = 5000*, *order='asc'*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get quotes for an options ticker symbol in a given time range. Note that you need to have an option symbol in correct format for this endpoint. You can use `ReferenceClient.get_option_contracts` to query option contracts using many filter parameters such as underlying symbol etc. Official Docs

Parameters

- **option_symbol** – The options ticker symbol to get quotes for. for eg `O:TSLA210903C00700000`. you can pass the symbol with or without the prefix `O:`
- **timestamp** – Query by quote timestamp. You can supply a `date`, `datetime` object or a `nanosecond UNIX timestamp` or a string in format: `YYYY-MM-DD`.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp

- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **sort** – Sort field used for ordering. Defaults to timestamp. See *polygon.enums.OptionQuotesSort* for available choices.
- **limit** – Limit the number of results returned. Defaults to 5000. max is 50000.
- **order** – order of the results. Defaults to `asc`. See *polygon.enums.SortOrder* for info and available choices.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_last_trade`(*ticker: str*, *raw_response: bool = False*)

Get the most recent trade for a given options contract. Official Docs

Parameters

- **ticker** – The ticker symbol of the options contract. Eg: `O:TSLA210903C00700000`
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of `raw_response`. Defaults to Dict.

`get_daily_open_close`(*symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the OCHLV and after-hours prices of a contract on a certain date. Official Docs

Parameters

- **symbol** – The option symbol we want daily-OCHLV for. eg `O:FB210903C00700000`. You can pass it with or without the prefix `O:`

- **date** – The date/day of the daily-OCHLV to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_aggregate_bars`(*symbol: str*, *from_date*, *to_date*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for an option contract over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the contract. eg `O:FB210903C00700000`. You can pass in with or without the prefix `O:`
- **from_date** – The start of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. `asc` default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000. see this article for more info.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number. defaults to 1.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to `day`
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.

- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

`get_snapshot`(*underlying_symbol: str*, *option_symbol: str*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get the snapshot of an option contract for a stock equity. Official Docs

Parameters

- **underlying_symbol** – The underlying ticker symbol of the option contract. eg `AMD`
- **option_symbol** – the option symbol. You can use use the *Working with Option Symbols* section to make it easy to work with option symbols in polygon or tda formats.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_previous_close`(*ticker: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified option contract. Official Docs

Parameters

- **ticker** – The ticker symbol of the options contract. Eg: `O:TSLA210903C00700000`

- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of `raw_response`. Defaults to Dict.

13.3.3 Options Async Client

class polygon.options.options.**AsyncOptionsClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the Options REST endpoints for async uses. Note that you should always import names from top level. eg: `from polygon import OptionsClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

async get_trades(*option_symbol: str*, *timestamp=None*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *sort='timestamp'*, *limit: int = 5000*, *order='asc'*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for an options ticker symbol in a given time range. Note that you need to have an option symbol in correct format for this endpoint. You can use `ReferenceClient.get_option_contracts` to query option contracts using many filter parameters such as underlying symbol etc. Official Docs

Parameters

- **option_symbol** – The options ticker symbol to get trades for. for eg `O:TSLA210903C00700000`. you can pass the symbol with or without the prefix `O:`
- **timestamp** – Query by trade timestamp. You can supply a `date`, `datetime` object or a `nanosecond UNIX timestamp` or a string in format: `YYYY-MM-DD`.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **sort** – Sort field used for ordering. Defaults to timestamp. See *polygon.enums.OptionTradesSort* for available choices.
- **limit** – Limit the number of results returned. Defaults to 100. max is 50000.
- **order** – order of the results. Defaults to `asc`. See *polygon.enums.SortOrder* for info and available choices.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`async get_quotes`(*option_symbol: str*, *timestamp=None*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *sort='timestamp'*, *limit: int = 5000*, *order='asc'*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get quotes for an options ticker symbol in a given time range. Note that you need to have an option symbol in correct format for this endpoint. You can use `ReferenceClient.get_option_contracts` to query option contracts using many filter parameters such as underlying symbol etc. Official Docs

Parameters

- **option_symbol** – The options ticker symbol to get quotes for. for eg `O:TSLA210903C00700000`. you can pass the symbol with or without the prefix `O:`
- **timestamp** – Query by quote timestamp. You can supply a `date`, `datetime` object or a `nanosecond UNIX timestamp` or a string in format: `YYYY-MM-DD`.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **sort** – Sort field used for ordering. Defaults to timestamp. See *polygon.enums.OptionQuotesSort* for available choices.
- **limit** – Limit the number of results returned. Defaults to 5000. max is 50000.
- **order** – order of the results. Defaults to `asc`. See *polygon.enums.SortOrder* for info and available choices.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_last_trade(*ticker: str*, *raw_response: bool = False*)

Get the most recent trade for a given options contract - Async Official Docs

Parameters

- **ticker** – The ticker symbol of the options contract. Eg: `O:TSLA210903C00700000`
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of `raw_response`. Defaults to Dict.

async get_daily_open_close(*symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the OCHLV and after-hours prices of a contract on a certain date. Official Docs

Parameters

- **symbol** – The option symbol we want daily-OCHLV for. eg `O:FB210903C00700000`. You can pass it with or without the prefix `O:`
- **date** – The date/day of the daily-OCHLV to retrieve. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_aggregate_bars(*symbol: str*, *from_date*, *to_date*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *multiplier: int = 1*, *timespan='day'*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for an option contract over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the contract. eg `O:FB210903C00700000`. You can pass in with or without the prefix `O:`
- **from_date** – The start of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`

- **to_date** – The end of the aggregate time window. Could be `datetime` or `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. See *polygon.enums.SortOrder* for choices. `asc` default.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000. see this article for more info.
- **multiplier** – The size of the timespan multiplier. Must be a positive whole number. defaults to 1.
- **timespan** – The size of the time window. See *polygon.enums.Timespan* for choices. defaults to `day`
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

async get_snapshot(*underlying_symbol: str*, *option_symbol: str*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get the snapshot of an option contract for a stock equity. Official Docs

Parameters

- **underlying_symbol** – The underlying ticker symbol of the option contract. eg `AMD`
- **option_symbol** – the option symbol. You can use use the *Working with Option Symbols* section to make it easy to work with option symbols in polygon or tda formats.
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.

- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_previous_close(*ticker: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified option contract - Async Official Docs

Parameters

- **ticker** – The ticker symbol of the options contract. Eg: `O:TSLA210903C00700000`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to false to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

Either a Dictionary or a Response object depending on value of `raw_response`. Defaults to Dict.

13.4 References Clients

13.4.1 Reference Sync Client

class polygon.reference_apis.reference_api.**SyncReferenceClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the References REST endpoints. Note that you should always import names from top level. eg: `from polygon import ReferenceClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.

get_tickers(*symbol: str = ''*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *symbol_type=''*, *market=''*, *exchange: str = ''*, *cusip: Optional[str] = None*, *cik: str = ''*, *date=None*, *search: Optional[str] = None*, *active: bool = True*, *sort='ticker'*, *order='asc'*, *limit: int = 1000*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Query all ticker symbols which are supported by Polygon.io. This API currently includes Stocks/Equities, Crypto, and Forex. Official Docs

Parameters

- **symbol** – Specify a ticker symbol. Defaults to empty string which queries all tickers.
- **ticker_lt** – Return results where this field is less than the value given
- **ticker_lte** – Return results where this field is less than or equal to the value given
- **ticker_gt** – Return results where this field is greater than the value given
- **ticker_gte** – Return results where this field is greater than or equal to the value given
- **symbol_type** – Specify the type of the tickers. See *polygon.enums.TickerType* for common choices. Find all supported types via the Ticker Types API Defaults to empty string which queries all types.
- **market** – Filter by market type. By default all markets are included. See *polygon.enums.TickerMarketType* for available choices.
- **exchange** – Specify the primary exchange of the asset in the ISO code format. Find more information about the ISO codes at the ISO org website. Defaults to empty string which queries all exchanges.
- **cusip** – Specify the `CUSIP` code of the asset you want to search for. Find more information about CUSIP codes on their website Defaults to empty string which queries all CUSIPs
- **cik** – Specify the `CIK` of the asset you want to search for. Find more information about CIK codes at their website Defaults to empty string which queries all CIKs.
- **date** – Specify a point in time to retrieve tickers available on that date. Defaults to the most recent available date. Could be `datetime`, `date` or a string `YYYY-MM-DD`
- **search** – Search for terms within the ticker and/or company name. for eg `MS` will match matching symbols
- **active** – Specify if the tickers returned should be actively traded on the queried date. Default is True

- **sort** – The field to sort the results on. Default is ticker. If the search query parameter is present, sort is ignored and results are ordered by relevance. See *polygon.enums.TickerSortType* for available choices.
- **order** – The order to sort the results on. Default is asc. See *polygon.enums.SortOrder* for available choices.
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_ticker_types`(*asset_class=None*, *locale=None*, *raw_response: bool = False*)

Get a mapping of ticker types to their descriptive names. Official Docs

Parameters

- **asset_class** – Filter by asset class. see *polygon.enums.AssetClass* for choices
- **locale** – Filter by locale. See *polygon.enums.Locale* for choices
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_ticker_details`(*symbol: str*, *date=None*, *raw_response: bool = False*)

Get a single ticker supported by Polygon.io. This response will have detailed information about the ticker and the company behind it. Official Docs

Parameters

- **symbol** – The ticker symbol of the asset.
- **date** – Specify a point in time to get information about the ticker available on that date. When retrieving information from SEC filings, we compare this date with the period of report date on the SEC filing. Defaults to the most recent available date.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_option_contract`(*ticker: str*, *as_of_date=None*, *raw_response: bool = False*)

get Info about an option contract Official Docs

Parameters

- **ticker** – An option ticker in standard format. The lib provides easy functions to build and work with option symbols
- **as_of_date** – Specify a point in time for the contract. You can pass a `datetime` or `date` object or a string in format `YYYY-MM-DD`. Defaults to today's date
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_option_contracts`(*underlying_ticker: Optional[str] = None*, *ticker: Optional[str] = None*, *contract_type=None*, *expiration_date=None*, *expiration_date_lt=None*, *expiration_date_lte=None*, *expiration_date_gt=None*, *expiration_date_gte=None*, *order='asc'*, *sort='expiration_date'*, *limit=1000*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

List currently active options contracts Official Docs

Parameters

- **underlying_ticker** – Query for contracts relating to an underlying stock ticker.
- **ticker** – Query for a contract by option ticker.
- **contract_type** – Query by the type of contract. see *polygon.enums.OptionsContractType* for choices
- **expiration_date** – Query by contract expiration date. either `datetime`, `date` or string `YYYY-MM-DD`
- **expiration_date_lt** – expiration date less than given value
- **expiration_date_lte** – expiration date less than equal to given value
- **expiration_date_gt** – expiration_date greater than given value
- **expiration_date_gte** – expiration_date greater than equal to given value
- **order** – Order of results. See *polygon.enums.SortOrder* for choices.

- **sort** – Sort field for ordering. See *polygon.enums.OptionsContractsSortType* for choices. defaults to expiration_date
- **limit** – Limit the size of the response, default is 1000. Pagination is supported by the pagination function below
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if all_pages is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter raw_page_responses. This argument is Only considered if all_pages is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if merge_all_pages is set to False. Default: False
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

get_ticker_news(*symbol: Optional[str] = None*, *limit: int = 1000*, *order='desc'*, *sort='published_utc'*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *published_utc=None*, *published_utc_lt=None*, *published_utc_lte=None*, *published_utc_gt=None*, *published_utc_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get the most recent news articles relating to a stock ticker symbol, including a summary of the article and a link to the original source. Official Docs

Parameters

- **symbol** – To get news mentioning the name given. Defaults to empty string which doesn't filter tickers
- **limit** – Limit the size of the response, default is 1000 which is also the max. Pagination is supported by the pagination function below
- **order** – Order the results. See *polygon.enums.SortOrder* for choices.
- **sort** – The field key to sort. See *polygon.enums.TickerNewsSort* for choices.
- **ticker_lt** – Return results where this field is less than the value.
- **ticker_lte** – Return results where this field is less than or equal to the value.
- **ticker_gt** – Return results where this field is greater than the value
- **ticker_gte** – Return results where this field is greater than or equal to the value.

- **published_utc** – A date string `YYYY-MM-DD` or `datetime` for published date time filters.
- **published_utc_lt** – Return results where this field is less than the value given
- **published_utc_lte** – Return results where this field is less than or equal to the value given
- **published_utc_gt** – Return results where this field is greater than the value given
- **published_utc_gte** – Return results where this field is greater than or equal to the value given
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

get_stock_dividends(*ticker: Optional[str] = None*, *ex_dividend_date=None*, *record_date=None*, *declaration_date=None*, *pay_date=None*, *frequency: Optional[int] = None*, *limit: int = 1000*, *cash_amount=None*, *dividend_type=None*, *sort: str = 'pay_date'*, *order: str = 'asc'*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *ex_dividend_date_lt=None*, *ex_dividend_date_lte=None*, *ex_dividend_date_gt=None*, *ex_dividend_date_gte=None*, *record_date_lt=None*, *record_date_lte=None*, *record_date_gt=None*, *record_date_gte=None*, *declaration_date_lt=None*, *declaration_date_lte=None*, *declaration_date_gt=None*, *declaration_date_gte=None*, *pay_date_lt=None*, *pay_date_lte=None*, *pay_date_gt=None*, *pay_date_gte=None*, *cash_amount_lt=None*, *cash_amount_lte=None*, *cash_amount_gt=None*, *cash_amount_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get a list of historical cash dividends, including the ticker symbol, declaration date, ex-dividend date, record date, pay date, frequency, and amount. Official Docs

Parameters

- **ticker** – Return the dividends that contain this ticker.

- **ex_dividend_date** – Query by ex-dividend date. could be a date, datetime object or a string YYYY-MM-DD
- **record_date** – Query by record date. could be a date, datetime object or a string YYYY-MM-DD
- **declaration_date** – Query by declaration date. could be a date, datetime object or a string YYYY-MM-DD
- **pay_date** – Query by pay date. could be a date, datetime object or a string YYYY-MM-DD
- **frequency** – Query by the number of times per year the dividend is paid out. No default value applied. see *polygon.enums.PayoutFrequency* for choices
- **limit** – Limit the size of the response, default is 1000 which is also the max. Pagination is supported by the pagination function below
- **cash_amount** – Query by the cash amount of the dividend.
- **dividend_type** – Query by the type of dividend. See *polygon.enums.DividendType* for choices
- **sort** – sort key used for ordering. See *polygon.enums.DividendSort* for choices.
- **order** – orders of results. defaults to asc. see *polygon.enums.SortOrder* for choices
- **ticker_lt** – filter where ticker is less than given value (alphabetically)
- **ticker_lte** – filter where ticker is less than or equal to given value (alphabetically)
- **ticker_gt** – filter where ticker is greater than given value (alphabetically)
- **ticker_gte** – filter where ticker is greater than or equal to given value (alphabetically)
- **ex_dividend_date_lt** – filter where ex-div date is less than given date
- **ex_dividend_date_lte** – filter where ex-div date is less than or equal to given date
- **ex_dividend_date_gt** – filter where ex-div date is greater than given date
- **ex_dividend_date_gte** – filter where ex-div date is greater than or equal to given date
- **record_date_lt** – filter where record date is less than given date
- **record_date_lte** – filter where record date is less than or equal to given date
- **record_date_gt** – filter where record date is greater than given date
- **record_date_gte** – filter where record date is greater than or equal to given date
- **declaration_date_lt** – filter where declaration date is less than given date
- **declaration_date_lte** – filter where declaration date is less than or equal to given date
- **declaration_date_gt** – filter where declaration date is greater than given date
- **declaration_date_gte** – filter where declaration date is greater than or equal to given date
- **pay_date_lt** – filter where pay date is less than given date
- **pay_date_lte** – filter where pay date is less than or equal to given date
- **pay_date_gt** – filter where pay date is greater than given date
- **pay_date_gte** – filter where pay date is greater than or equal to given date
- **cash_amount_lt** – filter where cash amt is less than given value

- **cash_amount_lte** – filter where cash amt is less than or equal to given value
- **cash_amount_gt** – filter where cash amt is greater than given value
- **cash_amount_gte** – filter where cash amt is greater than or equal to given value
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

get_stock_financials_vx(*ticker: Optional[str] = None*, *cik: Optional[str] = None*, *company_name: Optional[str] = None*, *company_name_search: Optional[str] = None*, *sic: Optional[str] = None*, *filing_date=None*, *filing_date_lt=None*, *filing_date_lte=None*, *filing_date_gt=None*, *filing_date_gte=None*, *period_of_report_date=None*, *period_of_report_date_lt=None*, *period_of_report_date_lte=None*, *period_of_report_date_gt=None*, *period_of_report_date_gte=None*, *time_frame=None*, *include_sources: bool = False*, *order='asc'*, *limit: int = 50*, *sort='filing_date'*, *raw_response: bool = False*)

Get historical financial data for a stock ticker. The financials data is extracted from XBRL from company SEC filings using this methodology Official Docs

This API is experimental and will replace `get_stock_financials()` in future.

Parameters

- **ticker** – Filter query by company ticker.
- **cik** – filter the Query by `central index key (CIK)` Number
- **company_name** – filter the query by company name
- **company_name_search** – partial match text search for company names
- **sic** – Query by `standard industrial classification (SIC)`
- **filing_date** – Query by the date when the filing with financials data was filed. `datetime/date` or string `YYYY-MM-DD`

- **filing_date_lt** – filter for filing date less than given value
- **filing_date_lte** – filter for filing date less than equal to given value
- **filing_date_gt** – filter for filing date greater than given value
- **filing_date_gte** – filter for filing date greater than equal to given value
- **period_of_report_date** – query by The period of report for the filing with financials data. `datetime/date` or string in format: `YYY-MM-DD`.
- **period_of_report_date_lt** – filter for period of report date less than given value
- **period_of_report_date_lte** – filter for period of report date less than equal to given value
- **period_of_report_date_gt** – filter for period of report date greater than given value
- **period_of_report_date_gte** – filter for period of report date greater than equal to given value
- **time_frame** – Query by timeframe. Annual financials originate from 10-K filings, and quarterly financials originate from 10-Q filings. Note: Most companies do not file quarterly reports for Q4 and instead include those financials in their annual report, so some companies my not return quarterly financials for Q4. See *polygon.enums.StockFinancialsTimeframe* for choices.
- **include_sources** – Whether or not to include the xpath and formula attributes for each financial data point. See the xpath and formula response attributes for more info. `False` by default
- **order** – Order results based on the sort field. 'asc' by default. See *polygon.enums.SortOrder* for choices.
- **limit** – number of max results to obtain. defaults to 50.
- **sort** – Sort field key used for ordering. 'filing_date' default. see *polygon.enums.StockFinancialsSortKey* for choices.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_stock_splits(*ticker: Optional[str] = None*, *execution_date=None*, *reverse_split: Optional[bool] = None*, *order: str = 'asc'*, *sort: str = 'execution_date'*, *limit: int = 1000*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *execution_date_lt=None*, *execution_date_lte=None*, *execution_date_gt=None*, *execution_date_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get a list of historical stock splits, including the ticker symbol, the execution date, and the factors of the split ratio. Official Docs

Parameters

- **ticker** – Return the stock splits that contain this ticker. defaults to no ticker filter returning all.

- **execution_date** – query by execution date. could be a date, datetime object or a string YYYY-MM-DD
- **reverse_split** – Query for reverse stock splits. A split ratio where split_from is greater than split_to represents a reverse split. By default this filter is not used.
- **order** – Order results based on the sort field. defaults to ascending. See *polygon.enums.SortOrder* for choices
- **sort** – Sort field used for ordering. Defaults to 'execution_date'. See *polygon.enums.SplitsSortKey* for choices.
- **limit** – Limit the size of the response, default is 1000 which is also the max. Pagination is supported by the pagination function below
- **ticker_lt** – filter where ticker name is less than given value (alphabetically)
- **ticker_lte** – filter where ticker name is less than or equal to given value (alphabetically)
- **ticker_gt** – filter where ticker name is greater than given value (alphabetically)
- **ticker_gte** – filter where ticker name is greater than or equal to given value (alphabetically)
- **execution_date_lt** – filter where execution date is less than given value
- **execution_date_lte** – filter where execution date is less than or equal to given value
- **execution_date_gt** – filter where execution date is greater than given value
- **execution_date_gte** – filter where execution date is greater than or equal to given value
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if all_pages is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter raw_page_responses. This argument is Only considered if all_pages is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if merge_all_pages is set to False. Default: False
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

get_market_holidays(*raw_response: bool = False*)

Get upcoming market holidays and their open/close times. Official Docs

Parameters
raw_response – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

get_market_status(*raw_response: bool = False*)

Get the current trading status of the exchanges and overall financial markets. Official Docs

Parameters
raw_response – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

get_conditions(*asset_class=None*, *data_type=None*, *condition_id=None*, *sip=None*, *order=None*, *limit: int = 50*, *sort='name'*, *raw_response: bool = False*)

List all conditions that Polygon.io uses. Official Docs

Parameters

- **asset_class** – Filter for conditions within a given asset class. See *polygon.enums.AssetClass* for choices. Defaults to all assets.
- **data_type** – Filter by data type. See *polygon.enums.ConditionsDataType* for choices. defaults to all.
- **condition_id** – Filter for conditions with a given ID
- **sip** – Filter by SIP. If the condition contains a mapping for that SIP, the condition will be returned.
- **order** – Order results. See *polygon.enums.SortOrder* for choices.
- **limit** – limit the number of results. defaults to 50.
- **sort** – Sort field used for ordering. Defaults to 'name'. See *polygon.enums.ConditionsSortKey* for choices.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

get_exchanges(*asset_class=None*, *locale=None*, *raw_response: bool = False*)

List all exchanges that Polygon.io knows about. Official Docs

Parameters

- **asset_class** – filter by asset class. See *polygon.enums.AssetClass* for choices.
- **locale** – Filter by locale name. See *polygon.enums.Locale*

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.4.2 Reference Async Client

class polygon.reference_apis.reference_api.**AsyncReferenceClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the References REST endpoints. Note that you should always import names from top level. eg: `from polygon import ReferenceClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

async get_tickers(*symbol: str = ''*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *symbol_type=''*, *market=''*, *exchange: str = ''*, *cusip: Optional[str] = None*, *cik: str = ''*, *date=None*, *search: Optional[str] = None*, *active: bool = True*, *sort='ticker'*, *order: str = 'asc'*, *limit: int = 1000*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Query all ticker symbols which are supported by Polygon.io. This API currently includes Stocks/Equities, Crypto, and Forex. Official Docs

Parameters

- **symbol** – Specify a ticker symbol. Defaults to empty string which queries all tickers.
- **ticker_lt** – Return results where this field is less than the value given
- **ticker_lte** – Return results where this field is less than or equal to the value given
- **ticker_gt** – Return results where this field is greater than the value given
- **ticker_gte** – Return results where this field is greater than or equal to the value given
- **symbol_type** – Specify the type of the tickers. See *polygon.enums.TickerType* for common choices. Find all supported types via the Ticker Types API Defaults to empty string which queries all types.
- **market** – Filter by market type. By default all markets are included. See *polygon.enums.TickerMarketType* for available choices.
- **exchange** – Specify the primary exchange of the asset in the ISO code format. Find more information about the ISO codes at the ISO org website. Defaults to empty string which queries all exchanges.
- **cusip** – Specify the `CUSIP` code of the asset you want to search for. Find more information about CUSIP codes on their website Defaults to empty string which queries all CUSIPs
- **cik** – Specify the `CIK` of the asset you want to search for. Find more information about CIK codes at their website Defaults to empty string which queries all CIKs.
- **date** – Specify a point in time to retrieve tickers available on that date. Defaults to the most recent available date. Could be `datetime`, `date` or a string `YYYY-MM-DD`
- **search** – Search for terms within the ticker and/or company name. for eg `MS` will match matching symbols
- **active** – Specify if the tickers returned should be actively traded on the queried date. Default is True
- **sort** – The field to sort the results on. Default is ticker. If the search query parameter is present, sort is ignored and results are ordered by relevance. See *polygon.enums.TickerSortType* for available choices.
- **order** – The order to sort the results on. Default is asc. See *polygon.enums.SortOrder* for available choices.
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.

- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_ticker_types(*asset_class=None*, *locale=None*, *raw_response: bool = False*)

Get a mapping of ticker types to their descriptive names - Async method Official Docs

Parameters

- **asset_class** – Filter by asset class. see *polygon.enums.AssetClass* for choices
- **locale** – Filter by locale. See *polygon.enums.Locale* for choices
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_ticker_details(*symbol: str*, *date=None*, *raw_response: bool = False*)

Get a single ticker supported by Polygon.io. This response will have detailed information about the ticker and the company behind it. Official Docs

Parameters

- **symbol** – The ticker symbol of the asset.
- **date** – Specify a point in time to get information about the ticker available on that date. When retrieving information from SEC filings, we compare this date with the period of report date on the SEC filing. Defaults to the most recent available date.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_option_contract(*ticker: str*, *as_of_date=None*, *raw_response: bool = False*)

get Info about an option contract Official Docs

Parameters

- **ticker** – An option ticker in standard format. The lib provides easy functions to build and work with option symbols
- **as_of_date** – Specify a point in time for the contract. You can pass a `datetime` or `date` object or a string in format `YYYY-MM-DD`. Defaults to today's date
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_option_contracts(*underlying_ticker: Optional[str] = None*, *ticker: Optional[str] = None*, *contract_type=None*, *expiration_date=None*, *expiration_date_lt=None*, *expiration_date_lte=None*, *expiration_date_gt=None*, *expiration_date_gte=None*, *order='asc'*, *sort='expiration_date'*, *limit=1000*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

List currently active options contracts Official Docs

Parameters

- **underlying_ticker** – Query for contracts relating to an underlying stock ticker.
- **ticker** – Query for a contract by option ticker.
- **contract_type** – Query by the type of contract. see *polygon.enums.OptionsContractType* for choices
- **expiration_date** – Query by contract expiration date. either `datetime`, `date` or string `YYYY-MM-DD`
- **expiration_date_lt** – expiration date less than given value
- **expiration_date_lte** – expiration date less than equal to given value
- **expiration_date_gt** – expiration_date greater than given value
- **expiration_date_gte** – expiration_date greater than equal to given value
- **order** – Order of results. See *polygon.enums.SortOrder* for choices.
- **sort** – Sort field for ordering. See *polygon.enums.OptionsContractsSortType* for choices. defaults to expiration_date
- **limit** – Limit the size of the response, default is 1000. `Pagination` is supported by the pagination function below
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True

- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_ticker_news(*symbol: Optional[str] = None*, *limit: int = 1000*, *order='desc'*, *sort='published_utc'*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *published_utc=None*, *published_utc_lt=None*, *published_utc_lte=None*, *published_utc_gt=None*, *published_utc_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get the most recent news articles relating to a stock ticker symbol, including a summary of the article and a link to the original source - Async method Official Docs

Parameters

- **symbol** – To get news mentioning the name given. Defaults to empty string which doesn't filter tickers
- **limit** – Limit the size of the response, default is 1000 which is also the max. `Pagination` is supported by the pagination function below
- **order** – Order the results. See *polygon.enums.SortOrder* for choices.
- **sort** – The field key to sort. See *polygon.enums.TickerNewsSort* for choices.
- **ticker_lt** – Return results where this field is less than the value.
- **ticker_lte** – Return results where this field is less than or equal to the value.
- **ticker_gt** – Return results where this field is greater than the value
- **ticker_gte** – Return results where this field is greater than or equal to the value.
- **published_utc** – A date string `YYYY-MM-DD` or `datetime` for published date time filters.
- **published_utc_lt** – Return results where this field is less than the value given
- **published_utc_lte** – Return results where this field is less than or equal to the value given
- **published_utc_gt** – Return results where this field is greater than the value given
- **published_utc_gte** – Return results where this field is greater than or equal to the value given

- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_stock_dividends(*ticker: Optional[str] = None*, *ex_dividend_date=None*, *record_date=None*, *declaration_date=None*, *pay_date=None*, *frequency: Optional[int] = None*, *limit: int = 1000*, *cash_amount=None*, *dividend_type=None*, *sort: str = 'pay_date'*, *order: str = 'asc'*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *ex_dividend_date_lt=None*, *ex_dividend_date_lte=None*, *ex_dividend_date_gt=None*, *ex_dividend_date_gte=None*, *record_date_lt=None*, *record_date_lte=None*, *record_date_gt=None*, *record_date_gte=None*, *declaration_date_lt=None*, *declaration_date_lte=None*, *declaration_date_gt=None*, *declaration_date_gte=None*, *pay_date_lt=None*, *pay_date_lte=None*, *pay_date_gt=None*, *pay_date_gte=None*, *cash_amount_lt=None*, *cash_amount_lte=None*, *cash_amount_gt=None*, *cash_amount_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get a list of historical cash dividends, including the ticker symbol, declaration date, ex-dividend date, record date, pay date, frequency, and amount. Official Docs

Parameters

- **ticker** – Return the dividends that contain this ticker.
- **ex_dividend_date** – Query by ex-dividend date. could be a date, datetime object or a string `YYYY-MM-DD`
- **record_date** – Query by record date. could be a date, datetime object or a string `YYYY-MM-DD`
- **declaration_date** – Query by declaration date. could be a date, datetime object or a string `YYYY-MM-DD`

- **pay_date** – Query by pay date. could be a date, datetime object or a string YYYY-MM-DD
- **frequency** – Query by the number of times per year the dividend is paid out. No default value applied. see *polygon.enums.PayoutFrequency* for choices
- **limit** – Limit the size of the response, default is 1000 which is also the max. Pagination is supported by the pagination function below
- **cash_amount** – Query by the cash amount of the dividend.
- **dividend_type** – Query by the type of dividend. See *polygon.enums.DividendType* for choices
- **sort** – sort key used for ordering. See *polygon.enums.DividendSort* for choices.
- **order** – orders of results. defaults to asc. see *polygon.enums.SortOrder* for choices
- **ticker_lt** – filter where ticker is less than given value (alphabetically)
- **ticker_lte** – filter where ticker is less than or equal to given value (alphabetically)
- **ticker_gt** – filter where ticker is greater than given value (alphabetically)
- **ticker_gte** – filter where ticker is greater than or equal to given value (alphabetically)
- **ex_dividend_date_lt** – filter where ex-div date is less than given date
- **ex_dividend_date_lte** – filter where ex-div date is less than or equal to given date
- **ex_dividend_date_gt** – filter where ex-div date is greater than given date
- **ex_dividend_date_gte** – filter where ex-div date is greater than or equal to given date
- **record_date_lt** – filter where record date is less than given date
- **record_date_lte** – filter where record date is less than or equal to given date
- **record_date_gt** – filter where record date is greater than given date
- **record_date_gte** – filter where record date is greater than or equal to given date
- **declaration_date_lt** – filter where declaration date is less than given date
- **declaration_date_lte** – filter where declaration date is less than or equal to given date
- **declaration_date_gt** – filter where declaration date is greater than given date
- **declaration_date_gte** – filter where declaration date is greater than or equal to given date
- **pay_date_lt** – filter where pay date is less than given date
- **pay_date_lte** – filter where pay date is less than or equal to given date
- **pay_date_gt** – filter where pay date is greater than given date
- **pay_date_gte** – filter where pay date is greater than or equal to given date
- **cash_amount_lt** – filter where cash amt is less than given value
- **cash_amount_lte** – filter where cash amt is less than or equal to given value
- **cash_amount_gt** – filter where cash amt is greater than given value
- **cash_amount_gte** – filter where cash amt is greater than or equal to given value
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.

- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_stock_financials_vx(*ticker: Optional[str] = None*, *cik: Optional[str] = None*, *company_name: Optional[str] = None*, *company_name_search: Optional[str] = None*, *sic: Optional[str] = None*, *filing_date=None*, *filing_date_lt=None*, *filing_date_lte=None*, *filing_date_gt=None*, *filing_date_gte=None*, *period_of_report_date=None*, *period_of_report_date_lt=None*, *period_of_report_date_lte=None*, *period_of_report_date_gt=None*, *period_of_report_date_gte=None*, *time_frame=None*, *include_sources: bool = False*, *order='asc'*, *limit: int = 50*, *sort='filing_date'*, *raw_response: bool = False*)

Get historical financial data for a stock ticker. The financials data is extracted from XBRL from company SEC filings using this methodology - Async method Official Docs

This API is experimental and will replace `get_stock_financials()` in future.

Parameters

- **ticker** – Filter query by company ticker.
- **cik** – filter the Query by `central index key (CIK)` Number
- **company_name** – filter the query by company name
- **company_name_search** – partial match text search for company names
- **sic** – Query by `standard industrial classification (SIC)`
- **filing_date** – Query by the date when the filing with financials data was filed. `datetime/date` or string `YYYY-MM-DD`
- **filing_date_lt** – filter for filing date less than given value
- **filing_date_lte** – filter for filing date less than equal to given value
- **filing_date_gt** – filter for filing date greater than given value
- **filing_date_gte** – filter for filing date greater than equal to given value

- **period_of_report_date** – query by The period of report for the filing with financials data. `datetime/date` or string in format: `YYY-MM-DD`.
- **period_of_report_date_lt** – filter for period of report date less than given value
- **period_of_report_date_lte** – filter for period of report date less than equal to given value
- **period_of_report_date_gt** – filter for period of report date greater than given value
- **period_of_report_date_gte** – filter for period of report date greater than equal to given value
- **time_frame** – Query by timeframe. Annual financials originate from 10-K filings, and quarterly financials originate from 10-Q filings. Note: Most companies do not file quarterly reports for Q4 and instead include those financials in their annual report, so some companies my not return quarterly financials for Q4. See *polygon.enums.StockFinancialsTimeframe* for choices.
- **include_sources** – Whether or not to include the xpath and formula attributes for each financial data point. See the xpath and formula response attributes for more info. `False` by default
- **order** – Order results based on the sort field. 'asc' by default. See *polygon.enums.SortOrder* for choices.
- **limit** – number of max results to obtain. defaults to 50.
- **sort** – Sort field key used for ordering. 'filing_date' default. see *polygon.enums.StockFinancialsSortKey* for choices.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_stock_splits(*ticker: Optional[str] = None*, *execution_date=None*, *reverse_split: Optional[bool] = None*, *order: str = 'asc'*, *sort: str = 'execution_date'*, *limit: int = 1000*, *ticker_lt=None*, *ticker_lte=None*, *ticker_gt=None*, *ticker_gte=None*, *execution_date_lt=None*, *execution_date_lte=None*, *execution_date_gt=None*, *execution_date_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get a list of historical stock splits, including the ticker symbol, the execution date, and the factors of the split ratio. Official Docs

Parameters

- **ticker** – Return the stock splits that contain this ticker. defaults to no ticker filter returning all.
- **execution_date** – query by execution date. could be a date, datetime object or a string `YYYY-MM-DD`
- **reverse_split** – Query for reverse stock splits. A split ratio where split_from is greater than split_to represents a reverse split. By default this filter is not used.
- **order** – Order results based on the sort field. defaults to ascending. See *polygon.enums.SortOrder* for choices

- **sort** – Sort field used for ordering. Defaults to 'execution_date'. See *polygon.enums.SplitsSortKey* for choices.
- **limit** – Limit the size of the response, default is 1000 which is also the max. Pagination is supported by the pagination function below
- **ticker_lt** – filter where ticker name is less than given value (alphabetically)
- **ticker_lte** – filter where ticker name is less than or equal to given value (alphabetically)
- **ticker_gt** – filter where ticker name is greater than given value (alphabetically)
- **ticker_gte** – filter where ticker name is greater than or equal to given value (alphabetically)
- **execution_date_lt** – filter where execution date is less than given value
- **execution_date_lte** – filter where execution date is less than or equal to given value
- **execution_date_gt** – filter where execution date is greater than given value
- **execution_date_gte** – filter where execution date is greater than or equal to given value
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if all_pages is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter raw_page_responses. This argument is Only considered if all_pages is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if merge_all_pages is set to False. Default: False
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_market_holidays(*raw_response: bool = False*)

Get upcoming market holidays and their open/close times - Async method Official Docs

Parameters

raw_response – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

async get_market_status(*raw_response: bool = False*)

Get the current trading status of the exchanges and overall financial markets - Async method Official Docs

Parameters

raw_response – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

async get_conditions(*asset_class=None*, *data_type=None*, *condition_id=None*, *sip=None*, *order=None*, *limit: int = 50*, *sort='name'*, *raw_response: bool = False*)

List all conditions that Polygon.io uses - Async method Official Docs

Parameters

- **asset_class** – Filter for conditions within a given asset class. See *polygon.enums.AssetClass* for choices. Defaults to all assets.
- **data_type** – Filter by data type. See *polygon.enums.ConditionsDataType* for choices. defaults to all.
- **condition_id** – Filter for conditions with a given ID
- **sip** – Filter by SIP. If the condition contains a mapping for that SIP, the condition will be returned.
- **order** – Order results. See *polygon.enums.SortOrder* for choices.
- **limit** – limit the number of results. defaults to 50.
- **sort** – Sort field used for ordering. Defaults to 'name'. See *polygon.enums.ConditionsSortKey* for choices.
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

async get_exchanges(*asset_class=None*, *locale=None*, *raw_response: bool = False*)

List all exchanges that Polygon.io knows about - Async method Official Docs

Parameters

- **asset_class** – filter by asset class. See *polygon.enums.AssetClass* for choices.
- **locale** – Filter by locale name. See *polygon.enums.Locale*
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

13.5 Forex Clients

13.5.1 Forex Sync Client

class polygon.forex.forex_api.**SyncForexClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the Forex REST endpoints. Note that you should always import names from top level. eg: `from polygon import ForexClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.

get_historic_forex_ticks(*from_symbol: str*, *to_symbol: str*, *date*, *offset: Optional[Union[str, int]] = None*, *limit: int = 500*, *raw_response: bool = False*)

Get historic trade ticks for a forex currency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the forex currency pair.
- **to_symbol** – The "to" symbol of the forex currency pair.
- **date** – The date/day of the historic ticks to retrieve. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **offset** – The timestamp offset, used for pagination. This is the offset at which to start the results. Using the timestamp of the last result as the offset will give you the next page of results. I'm thinking about a good way to implement this type of pagination in the lib which doesn't have a `next_url` in the response attributes.
- **limit** – Limit the size of the response, max 10000. Default 500
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_quotes(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get NBBO Quotes for a forex ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want quotes for. eg: `C:EUR-USD`. you can pass with or without prefix C:
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.ForexQuotesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_last_quote`(*from_symbol: str*, *to_symbol: str*, *raw_response: bool = False*)

Get the last trade tick for a forex currency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the forex currency pair.
- **to_symbol** – The "to" symbol of the forex currency pair.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_aggregate_bars`(*symbol: str*, *from_date*, *to_date*, *multiplier: int = 1*, *timespan='day'*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a forex pair over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then `5-minute` bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the forex pair. eg: `C:EURUSD`. You can supply with or without prefix `C:`
- **from_date** – The start of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **multiplier** – The size of the timespan multiplier
- **timespan** – The size of the time window. Defaults to day candles. see *polygon.enums.Timespan* for choices
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. see *polygon.enums.SortOrder* for available choices. Defaults to `asc` which is oldest at the top.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration

(eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

`get_grouped_daily_bars`(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily open, high, low, and close (OHLC) for the entire forex markets. Official Docs

Parameters

- **date** – The date for the aggregate window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_previous_close`(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified forex pair. Official Docs

Parameters

- **symbol** – The ticker symbol of the forex pair.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_snapshot_all`(*symbols: list*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded forex symbols Official Docs

Parameters

- **symbols** – A list of tickers to get snapshots for.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_snapshot`(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded forex symbol. Official Docs

Parameters

- **symbol** – Symbol of the forex pair. eg: `C:EURUSD`. You can supply with or without prefix `C:`.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_gainers_and_losers`(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in forex markets. Official docs

Parameters

- **direction** – The direction of the snapshot results to return. See *polygon.enums.SnapshotDirection* for available choices. Defaults to Gainers.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`real_time_currency_conversion`(*from_symbol: str*, *to_symbol: str*, *amount: float*, *precision: int = 2*, *raw_response: bool = False*)

Get currency conversions using the latest market conversion rates. Note than you can convert in both directions. For example USD to CAD or CAD to USD. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **amount** – The amount to convert,
- **precision** – The decimal precision of the conversion. Defaults to 2 which is 2 decimal places accuracy.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.5.2 Forex Async Client

class polygon.forex.forex_api.**AsyncForexClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the Forex REST endpoints. Note that you should always import names from top level. eg: `from polygon import ForexClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

async get_historic_forex_ticks(*from_symbol: str*, *to_symbol: str*, *date*, *offset: Optional[Union[str, int]] = None*, *limit: int = 500*, *raw_response: bool = False*)

Get historic trade ticks for a forex currency pair - Async method. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the forex currency pair.
- **to_symbol** – The "to" symbol of the forex currency pair.
- **date** – The date/day of the historic ticks to retrieve. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **offset** – The timestamp offset, used for pagination. This is the offset at which to start the results. Using the timestamp of the last result as the offset will give you the next page of results. I'm thinking about a good way to implement this type of pagination in the lib which doesn't have a `next_url` in the response attributes.

- **limit** – Limit the size of the response, max 10000. Default 500
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make raw_response=True to get underlying response object

async **get_quotes**(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get NBBO Quotes for a forex ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want quotes for. eg: C:EUR-USD. you can pass with or without prefix C:
- **timestamp** – Query by trade timestamp. Could be datetime or date or string YYYY-MM-DD or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.ForexQuotesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if all_pages is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter raw_page_responses. This argument is Only considered if all_pages is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.

- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_last_quote(*from_symbol: str*, *to_symbol: str*, *raw_response: bool = False*)

Get the last trade tick for a forex currency pair - Async method Official Docs

Parameters

- **from_symbol** – The "from" symbol of the forex currency pair.
- **to_symbol** – The "to" symbol of the forex currency pair.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_aggregate_bars(*symbol: str*, *from_date*, *to_date*, *multiplier: int = 1*, *timespan='day'*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a forex pair over a given date range in custom time window sizes. For example, if `timespan = 'minute'` and `multiplier = '5'` then `5-minute` bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the forex pair. eg: `C:EURUSD`. You can supply with or without prefix `C:`
- **from_date** – The start of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **multiplier** – The size of the timespan multiplier
- **timespan** – The size of the time window. Defaults to day candles. see *polygon.enums.Timespan* for choices
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **sort** – Sort the results by timestamp. see *polygon.enums.SortOrder* for available choices. Defaults to `asc` which is oldest at the top.
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.

- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

async get_grouped_daily_bars(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily open, high, low, and close (OHLC) for the entire forex markets - Async method Official Docs

Parameters

- **date** – The date for the aggregate window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_previous_close(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified forex pair - Async method Official Docs

Parameters

- **symbol** – The ticker symbol of the forex pair.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_snapshot_all`(*symbols: list*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded forex symbols - Async method Official Docs

Parameters

- **symbols** – A list of tickers to get snapshots for.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_snapshot`(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded forex symbol - Async method Official Docs

Parameters

- **symbol** – Symbol of the forex pair. eg: `C:EURUSD`. You can supply with or without prefix `C:`.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_gainers_and_losers`(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in forex markets. Official docs

Parameters

- **direction** – The direction of the snapshot results to return. See *polygon.enums.SnapshotDirection* for available choices. Defaults to Gainers.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async real_time_currency_conversion`(*from_symbol: str*, *to_symbol: str*, *amount: float*, *precision: int = 2*, *raw_response: bool = False*)

Get currency conversions using the latest market conversion rates. Note than you can convert in both directions. For example USD to CAD or CAD to USD - Async method Official Docs

Parameters

- **from_symbol** – The "from" symbol of the pair.

- **to_symbol** – The "to" symbol of the pair.
- **amount** – The amount to convert,
- **precision** – The decimal precision of the conversion. Defaults to 2 which is 2 decimal places accuracy.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.6 Crypto Clients

13.6.1 Crypto Sync Client

class polygon.crypto.crypto_api.**SyncCryptoClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the crypto REST endpoints. Note that you should always import names from top level. eg: `from polygon import CryptoClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.
- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for date to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.

get_historic_trades(*from_symbol: str*, *to_symbol: str*, *date*, *offset: Optional[Union[str, int]] = None*, *limit: int = 500*, *raw_response: bool = False*)

Get historic trade ticks for a cryptocurrency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the crypto pair.
- **to_symbol** – The "to" symbol of the crypto pair.
- **date** – The date/day of the historic ticks to retrieve. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **offset** – The timestamp offset, used for pagination. This is the offset at which to start the results. Using the timestamp of the last result as the offset will give you the next page of

results. I'm trying to think of a good way to implement pagination in the library for these endpoints which do not return a `next_url` attribute.

- **limit** – Limit the size of the response, max 10000. Default 500
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_trades(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for a crypto ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want trades for. eg `X:BTC-USD`. you can pass with or without the prefix `C:`
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp
- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.CryptoTradesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.

- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

`get_last_trade`(*from_symbol: str*, *to_symbol: str*, *raw_response: bool = False*)

Get the last trade tick for a cryptocurrency pair. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_daily_open_close`(*from_symbol: str*, *to_symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the open, close prices of a cryptocurrency symbol on a certain day. Official Docs:

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **date** – The date of the requested open/close. Could be `datetime`, `date` or string `YYYY-MM-DD`.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_aggregate_bars`(*symbol: str*, *from_date*, *to_date*, *multiplier: int = 1*, *timespan='day'*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a cryptocurrency pair over a given date range in custom time window sizes. For example, if `timespan='minute'` and `multiplier='5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the currency pair. eg: `X:BTCUSD`. You can specify with or without prefix `X:`
- **from_date** – The start of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **multiplier** – The size of the timespan multiplier
- **timespan** – The size of the time window. Defaults to day candles. see *polygon.enums.Timespan* for choices
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **sort** – Order of sorting the results. See *polygon.enums.SortOrder* for available choices. Defaults to `asc` (oldest at the top)
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.
- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

get_grouped_daily_bars(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily open, high, low, and close (OHLC) for the entire cryptocurrency market. Official Docs

Parameters

- **date** – The date for the aggregate window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_previous_close`(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified cryptocurrency pair. Official Docs

Parameters

- **symbol** – The ticker symbol of the currency pair. eg: `X:BTCUSD`. You can specify with or without the prefix `X:`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_snapshot_all`(*symbols: list*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded cryptocurrency symbols Official Docs

Parameters

- **symbols** – A list of tickers to get snapshots for.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`get_snapshot`(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded cryptocurrency symbol. Official Docs

Parameters

- **symbol** – Symbol of the currency pair. eg: `X:BTCUSD`. you can specify with or without prefix `X:`
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_gainers_and_losers(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in cryptocurrency markets. Official docs

Parameters

- **direction** – The direction of the snapshot results to return. See *polygon.enums.SnapshotDirection* for available choices. Defaults to Gainers.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

get_level2_book(*symbol: str*, *raw_response: bool = False*)

Get the current level 2 book of a single ticker. This is the combined book from all of the exchanges. Official Docs

Parameters

- **symbol** – The cryptocurrency ticker. eg: `X:BTCUSD`. You can specify with or without the prefix `` `X: ``
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.6.2 Crypto Async Client

class polygon.crypto.crypto_api.**AsyncCryptoClient**(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

This class implements all the crypto REST endpoints. Note that you should always import names from top level. eg: `from polygon import CryptoClient` or `import polygon` (which allows you to access all names easily)

__init__(*api_key: str*, *connect_timeout: int = 10*, *read_timeout: int = 10*, *pool_timeout: int = 10*, *max_connections: Optional[int] = None*, *max_keepalive: Optional[int] = None*, *write_timeout: int = 10*)

Initiates a Client to be used to access all the endpoints.

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **connect_timeout** – The connection timeout in seconds. Defaults to 10. basically the number of seconds to wait for a connection to be established. Raises a `ConnectTimeout` if unable to connect within specified time limit.

- **read_timeout** – The read timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be received. Raises a `ReadTimeout` if unable to connect within the specified time limit.
- **pool_timeout** – The pool timeout in seconds. Defaults to 10. Basically the number of seconds to wait while trying to get a connection from connection pool. Do NOT change if you're unsure of what it implies
- **max_connections** – Max number of connections in the pool. Defaults to NO LIMITS. Do NOT change if you're unsure of application
- **max_keepalive** – max number of allowable keep alive connections in the pool. Defaults to no limit. Do NOT change if you're unsure of the applications.
- **write_timeout** – The write timeout in seconds. Defaults to 10. basically the number of seconds to wait for data to be written/posted. Raises a `WriteTimeout` if unable to connect within the specified time limit.

async get_historic_trades(*from_symbol: str*, *to_symbol: str*, *date*, *offset: Optional[Union[str, int]] = None*, *limit: int = 500*, *raw_response: bool = False*)

Get historic trade ticks for a cryptocurrency pair - Async method. Official Docs

Parameters

- **from_symbol** – The "from" symbol of the crypto pair.
- **to_symbol** – The "to" symbol of the crypto pair.
- **date** – The date/day of the historic ticks to retrieve. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **offset** – The timestamp offset, used for pagination. This is the offset at which to start the results. Using the timestamp of the last result as the offset will give you the next page of results. I'm trying to think of a good way to implement pagination in the library for these endpoints which do not return a `next_url` attribute.
- **limit** – Limit the size of the response, max 10000. Default 500
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_trades(*symbol: str*, *timestamp: Optional[int] = None*, *order=None*, *sort=None*, *limit: int = 5000*, *timestamp_lt=None*, *timestamp_lte=None*, *timestamp_gt=None*, *timestamp_gte=None*, *all_pages: bool = False*, *max_pages: Optional[int] = None*, *merge_all_pages: bool = True*, *verbose: bool = False*, *raw_page_responses: bool = False*, *raw_response: bool = False*)

Get trades for a crypto ticker symbol in a given time range. Official Docs

Parameters

- **symbol** – The ticker symbol you want trades for. eg `X:BTC-USD`. you can pass with or without the prefix `C:`
- **timestamp** – Query by trade timestamp. Could be `datetime` or `date` or string `YYYY-MM-DD` or a nanosecond timestamp

- **order** – sort order. see *polygon.enums.SortOrder* for available choices. defaults to None
- **sort** – field key to sort against. Defaults to None. see *polygon.enums.CryptoTradesSort* for choices
- **limit** – Limit the size of the response, max 50000 and default 5000.
- **timestamp_lt** – return results where timestamp is less than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_lte** – return results where timestamp is less than/equal to the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gt** – return results where timestamp is greater than the given value. Can be date or date string or nanosecond timestamp
- **timestamp_gte** – return results where timestamp is greater than/equal to the given value. Can be date or date string or nanosecond timestamp
- **all_pages** – Whether to paginate through next/previous pages internally. Defaults to False. If set to True, it will try to paginate through all pages and merge all pages internally for you.
- **max_pages** – how many pages to fetch. Defaults to None which fetches all available pages. Change to an integer to fetch at most that many pages. This param is only considered if `all_pages` is set to True
- **merge_all_pages** – If this is True, returns a single merged response having all the data. If False, returns a list of all pages received. The list can be either a list of response objects or decoded data itself, controlled by parameter `raw_page_responses`. This argument is Only considered if `all_pages` is set to True. Default: True
- **verbose** – Set to True to print status messages during the pagination process. Defaults to False.
- **raw_page_responses** – If this is true, the list of pages will be a list of corresponding Response objects. Else, it will be a list of actual data for pages. This parameter is only considered if `merge_all_pages` is set to False. Default: False
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. This is ignored if pagination is set to True.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If pagination is set to True, will return a merged response of all pages for convenience.

async get_last_trade(*from_symbol: str*, *to_symbol: str*, *raw_response: bool = False*)

Get the last trade tick for a cryptocurrency pair - Async method Official Docs

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_daily_open_close(*from_symbol: str*, *to_symbol: str*, *date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the open, close prices of a cryptocurrency symbol on a certain day - Async method Official Docs:

Parameters

- **from_symbol** – The "from" symbol of the pair.
- **to_symbol** – The "to" symbol of the pair.
- **date** – The date of the requested open/close. Could be `datetime`, `date` or string `YYYY-MM-DD`.
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_aggregate_bars(*symbol: str*, *from_date*, *to_date*, *multiplier: int = 1*, *timespan='day'*, *adjusted: bool = True*, *sort='asc'*, *limit: int = 5000*, *full_range: bool = False*, *run_parallel: bool = True*, *max_concurrent_workers: int = 10*, *warnings: bool = True*, *high_volatility: bool = False*, *raw_response: bool = False*)

Get aggregate bars for a cryptocurrency pair over a given date range in custom time window sizes. For example, if `timespan='minute'` and `multiplier='5'` then 5-minute bars will be returned. Official Docs

Parameters

- **symbol** – The ticker symbol of the currency pair. eg: `X:BTCUSD`. You can specify with or without prefix `X:`
- **from_date** – The start of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **to_date** – The end of the aggregate time window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **multiplier** – The size of the timespan multiplier
- **timespan** – The size of the time window. Defaults to day candles. see *polygon.enums.Timespan* for choices
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **sort** – Order of sorting the results. See *polygon.enums.SortOrder* for available choices. Defaults to `asc` (oldest at the top)
- **limit** – Limits the number of base aggregates queried to create the aggregate results. Max 50000 and Default 5000.

- **full_range** – Default False. If set to True, it will get the ENTIRE range you specify and **merge** all the responses and return ONE single list with all data in it. You can control its behavior with the next few arguments.
- **run_parallel** – Only considered if `full_range=True`. If set to true (default True), it will run an internal ThreadPool to get the responses. This is fine to do if you are not running your own ThreadPool. If you have many tickers to get aggs for, it's better to either use the async version of it OR set this to False and spawn threads for each ticker yourself.
- **max_concurrent_workers** – Only considered if `run_parallel=True`. Defaults to `your cpu cores * 5`. controls how many worker threads to use in internal ThreadPool
- **warnings** – Set to False to disable printing warnings if any when fetching the aggs. Defaults to True.
- **high_volatility** – Specifies whether the symbol/security in question is highly volatile which just means having a very high number of trades or being traded for a high duration (eg SPY, Bitcoin) If set to True, the lib will use a smaller chunk of time to ensure we don't miss any data due to 50k candle limit. Defaults to False.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary. Will be ignored if `full_range=True`

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object. If `full_range=True`, will return a single list with all the candles in it.

`async get_grouped_daily_bars`(*date*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the daily open, high, low, and close (OHLC) for the entire cryptocurrency market - Async method Official Docs

Parameters

- **date** – The date for the aggregate window. Could be `datetime`, `date` or string `YYYY-MM-DD`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

`async get_previous_close`(*symbol: str*, *adjusted: bool = True*, *raw_response: bool = False*)

Get the previous day's open, high, low, and close (OHLC) for the specified cryptocurrency pair - Async method Official Docs

Parameters

- **symbol** – The ticker symbol of the currency pair. eg: `X:BTCUSD`. You can specify with or without the prefix `X:`
- **adjusted** – Whether or not the results are adjusted for splits. By default, results are adjusted. Set this to False to get results that are NOT adjusted for splits.

- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_snapshot_all(*symbols: list*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for all traded cryptocurrency symbols - Async method Official Docs

Parameters

- **symbols** – A list of tickers to get snapshots for.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_snapshot(*symbol: str*, *raw_response: bool = False*)

Get the current minute, day, and previous day's aggregate, as well as the last trade and quote for a single traded cryptocurrency symbol - Async method Official Docs

Parameters

- **symbol** – Symbol of the currency pair. eg: `X:BTCUSD`. you can specify with or without prefix `X:`
- **raw_response** – Whether or not to return the Response Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_gainers_and_losers(*direction='gainers'*, *raw_response: bool = False*)

Get the current top 20 gainers or losers of the day in cryptocurrency markets - Async method Official docs

Parameters

- **direction** – The direction of the snapshot results to return. See *polygon.enums.SnapshotDirection* for available choices. Defaults to Gainers.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns
A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

async get_level2_book(*symbol: str*, *raw_response: bool = False*)

Get the current level 2 book of a single ticker. combined book from all of the exchanges - Async method Official Docs

Parameters

- **symbol** – The cryptocurrency ticker. eg: `X:BTCUSD`. You can specify with or without the prefix `X:.
- **raw_response** – Whether or not to return the `Response` Object. Useful for when you need to say check the status code or inspect the headers. Defaults to False which returns the json decoded dictionary.

Returns

A JSON decoded Dictionary by default. Make `raw_response=True` to get underlying response object

13.7 Callback Streamer Client (Sync)

class polygon.streaming.streaming.**StreamClient**(*api_key: str*, *cluster*, *host='socket.polygon.io'*, *on_message=None*, *on_close=None*, *on_error=None*, *enable_connection_logs: bool = False*)

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

Note that this is callback based stream client which is suitable for threaded/multi-processed applications. If you need to stream using an `asyncio` based stream client, see *Async Streamer Client*.

This class implements all the websocket endpoints. Note that you should always import names from top level. eg: `from polygon import StreamClient` or `import polygon` (which allows you to access all names easily)

Creating the client is as simple as: `client = StreamClient('MY_API_KEY', 'other_options')`

Once you have the client, you can call its methods to subscribe/unsubscribe to streams, change handlers and process messages. All methods have sane default values and almost everything can be customized.

Type Hinting tells you what data type a parameter is supposed to be. You should always use `enums` for most parameters to avoid supplying error prone values.

Take a look at the Official documentation to get an idea of the stream, data formatting for messages and related useful stuff.

__init__(*api_key: str*, *cluster*, *host='socket.polygon.io'*, *on_message=None*, *on_close=None*, *on_error=None*, *enable_connection_logs: bool = False*)

Initializes the callback function based stream client Official Docs

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **cluster** – Which market/cluster to connect to. See *polygon.enums.StreamCluster* for choices. NEVER connect to the same cluster again if there is an existing stream connected to it. The existing connection would be dropped and new one will be established. You can have up to 4 concurrent streams connected to 4 different clusters.
- **host** – Host url to connect to. Default is real time. See *polygon.enums.StreamHost* for choices.
- **on_message** – The function to be called when data is received. This is primary function you'll write to process the data from the stream. The function should accept one and only one `arg` (message). Default handler is *_default_on_msg()*.
- **on_close** – The function to be called when stream is closed. Function should accept two args (close_status_code, close_message). Default handler is *_default_on_close()*

- **on_error** – Function to be called when an error is encountered. Function should accept one arg (exception object). Default handler is *_default_on_error()*
- **enable_connection_logs** – Whether or not to print debug info related to the stream connection. Helpful for debugging.

_start_stream(*ping_interval: int = 21*, *ping_timeout: int = 20*, *ping_payload: str = ''*, *skip_utf8_validation: bool = True*)

Starts the Stream Event Loop. The loop is infinite and will continue to run until the stream is terminated, either manually or due to an exception. This method is for internal use only. you should always use *start_stream_thread()* to start the stream.

Parameters

- **ping_interval** – client would send a `ping` every specified number of seconds to server to keep connection alive. Set to 0 to disable pinging. Defaults to 21 seconds
- **ping_timeout** – Timeout in seconds if a `pong` (response to ping from server) is not received. The Stream is terminated as it is considered to be dead if no pong is received within the specified timeout. default: 20 seconds
- **ping_payload** – The option message to be sent with the ping. Better to leave it empty string.
- **skip_utf8_validation** – Whether to skip utf validation of messages. Defaults to True. Setting it to False may result in performance downgrade

Returns
None

start_stream_thread(*ping_interval: int = 21*, *ping_timeout: int = 20*, *ping_payload: str = ''*, *skip_utf8_validation: bool = True*)

Starts the Stream. This will not block the main thread and it spawns the streamer in its own thread.

Parameters

- **ping_interval** – client would send a `ping` every specified number of seconds to server to keep connection alive. Set to 0 to disable pinging. Defaults to 21 seconds
- **ping_timeout** – Timeout in seconds if a `pong` (response to ping from server) is not received. The Stream is terminated as it is considered to be dead if no pong is received within the specified timeout. default: 20 seconds
- **ping_payload** – The option message to be sent with the ping. Better to leave it empty string.
- **skip_utf8_validation** – Whether to skip utf validation of messages. Defaults to True. Setting it to False may result in performance downgrade

Returns
None

close_stream(**args*, ***kwargs*)

Close the websocket connection. Wait for thread to finish if running.

_authenticate()

Authenticates the client with the server using API key. Internal function, not meant to be called directly by users.

Returns
None

_modify_sub(*symbols=None*, *action='subscribe'*, *_prefix='T.'*, *force_uppercase_symbols: bool = True*)

Internal Function to send subscribe or unsubscribe requests to websocket. You should prefer using the corresponding methods to subscribe or unsubscribe to stream.

Parameters

- **symbols** – The list of symbols to apply the actions to.
- **action** – Defaults to subscribe which subscribes to requested stream. Change to unsubscribe to remove an existing subscription.
- **_prefix** – prefix of the stream service. See *polygon.enums.StreamServicePrefix* for choices.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

subscribe_stock_trades(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time trades for given stock ticker symbol(s).

Parameters

- **symbols** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_stock_trades(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

subscribe_stock_quotes(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Quotes for given stock ticker symbol(s).

Parameters

- **symbols** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_stock_quotes(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

subscribe_stock_minute_aggregates(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time minute aggregates for given stock ticker symbol(s).

Parameters

- **symbols** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_stock_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

subscribe_stock_second_aggregates(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time second aggregates for given stock ticker symbol(s).

Parameters

- **symbols** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_stock_second_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

subscribe_stock_limit_up_limit_down(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time LULD events for given stock ticker symbol(s).

Parameters

- **symbols** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_stock_limit_up_limit_down(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

subscribe_stock_imbalances(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Imbalance Events for given stock ticker symbol(s).

Parameters

- **symbols** – A list of tickers. Default is * which subscribes to ALL tickers in the market
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_stock_imbalances(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

subscribe_option_trades(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Trades for given Options contract.

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_option_trades(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Trades for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

subscribe_option_quotes(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Quotes for given Options contract.

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_option_quotes(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Quotes for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

subscribe_option_minute_aggregates(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Minute Aggregates for given Options contract(s).

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL tickers in the market. you can pass **with or without** the prefix `O:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_option_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Minute aggregates for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

`subscribe_option_second_aggregates`(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Options Second Aggregates for given Options contract(s).

Parameters

- **symbols** – A list of symbols. Default is * which subscribes to ALL tickers in the market. you can pass **with or without** the prefix `O:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`unsubscribe_option_second_aggregates`(*symbols: Optional[list] = None*)

Unsubscribe real-time Options Second Aggregates for given Options contract.

Parameters
symbols – A list of symbols. Default is * which subscribes to ALL symbols in the market. you can pass **with or without** the prefix `O:`

Returns
None

`subscribe_forex_quotes`(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time forex quotes for given forex pair(s).

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

`unsubscribe_forex_quotes`(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

Parameters
symbols – A list of forex tickers. Default is * which unsubscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

`subscribe_forex_minute_aggregates`(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time forex Minute Aggregates for given forex pair(s).

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_forex_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream service for the symbols specified. Defaults to all symbols.

Parameters

symbols – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

subscribe_crypto_trades(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Trades for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns

None

unsubscribe_crypto_trades(*symbols: Optional[list] = None*)

Unsubscribe real-time trades for given cryptocurrency pair(s).

Parameters

symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns

None

subscribe_crypto_quotes(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Quotes for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns

None

unsubscribe_crypto_quotes(*symbols: Optional[list] = None*)

Unsubscribe real-time quotes for given cryptocurrency pair(s).

Parameters

symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns

None

subscribe_crypto_minute_aggregates(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time Minute Aggregates for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_crypto_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe real-time minute aggregates for given cryptocurrency pair(s).

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

subscribe_crypto_level2_book(*symbols: Optional[list] = None*, *force_uppercase_symbols: bool = True*)

Stream real-time level 2 book data for given cryptocurrency pair(s).

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

unsubscribe_crypto_level2_book(*symbols: Optional[list] = None*)

Unsubscribe real-time level 2 book data for given cryptocurrency pair(s).

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

static _default_on_msg(*_ws: WebSocketApp*, *msg*)

Default handler for message processing

Parameters
msg – The message as received from the server

Returns
None

static _default_on_close(*_ws: WebSocketApp*, *close_code*, *msg*)

THe default function to be called when stream is closed.

Parameters

- **close_code** – The close code as received from server
- **msg** – The close message as received from server

Returns

`static _default_on_error(_ws: WebSocketApp, error, *args)`

Default function to be called when an error is encountered.

Parameters
error – The exception object as supplied by the handler

Returns
None

`_default_on_open(_ws: WebSocketApp, *args)`

Default function to be called when stream client is initialized. Takes care of the authentication.

Parameters
args – Any args supplied by the handler

Returns
None

`static _change_enum(val: ~typing.Union[str, ~enum.Enum, float, int], allowed_type=<class 'str'>)`

13.8 Async Streamer Client

`class polygon.streaming.async_streaming.AsyncStreamClient(api_key: str, cluster, host='socket.polygon.io', ping_interval: Optional[int] = 20, ping_timeout: Optional[int] = 19, max_message_size: int = 1048576, max_memory_queue: Optional[int] = 32, read_limit: int = 65536, write_limit: int = 65536)`

These docs are not meant for general users. These are library API references. The actual docs will be available on the index page when they are prepared.

Note that this is asyncio based stream client which is suitable for async applications. If you need to stream using an `callback` based stream client, see *Callback Streamer Client (Sync)*.

This class implements all the websocket endpoints. Note that you should always import names from top level. eg: `from polygon import AsyncStreamClient` or `import polygon` (which allows you to access all names easily)

Creating the client is as simple as: `client = AsyncStreamClient('MY_API_KEY', 'other_options')`

Once you have the client, you can call its methods to subscribe/unsubscribe to streams, change handlers and process messages. All methods have sane default values and almost everything can be customized.

Type Hinting tells you what data type a parameter is supposed to be. You should always use `enums` for most parameters to avoid supplying error prone values.

Take a look at the Official documentation to get an idea of the stream, data formatting for messages and related useful stuff.

__init__(*api_key: str*, *cluster*, *host='socket.polygon.io'*, *ping_interval: Optional[int] = 20*, *ping_timeout: Optional[int] = 19*, *max_message_size: int = 1048576*, *max_memory_queue: Optional[int] = 32*, *read_limit: int = 65536*, *write_limit: int = 65536*)

Initializes the stream client for async streaming Official Docs

Parameters

- **api_key** – Your API Key. Visit your dashboard to get yours.
- **cluster** – Which market/cluster to connect to. See *polygon.enums.StreamCluster* for choices. NEVER connect to the same cluster again if there is an existing stream connected to it. The existing connection would be dropped and new one will be established. You can have up to 4 concurrent streams connected to 4 different clusters.
- **host** – Host url to connect to. Default is real time. See *polygon.enums.StreamHost* for choices
- **ping_interval** – Send a `ping` to server every specified number of seconds to keep the connection alive. Defaults to 20 seconds. Setting to 0 disables pinging.
- **ping_timeout** – The number of seconds to wait after sending a ping for the response (`pong`). If no response is received from the server in those many seconds, stream is considered dead and exits with code `1011`. Defaults to 19 seconds.
- **max_message_size** – The max_size parameter enforces the maximum size for incoming messages in bytes. The default value is `1 MiB` (not MB). None disables the limit. If a message larger than the maximum size is received, `recv()` will raise `ConnectionClosedError` and the connection will be closed with code `1009`
- **max_memory_queue** – sets the maximum length of the queue that holds incoming messages. The default value is `32`. None disables the limit. Messages are added to an in-memory queue when they're received; then `recv()` pops from that queue
- **read_limit** – sets the high-water limit of the buffer for incoming bytes. The low-water limit is half the high-water limit. The default value is `64 KiB`, half of asyncio's default. Don't change if you are unsure of what it implies.
- **write_limit** – The write_limit argument sets the high-water limit of the buffer for outgoing bytes. The low-water limit is a quarter of the high-water limit. The default value is `64 KiB`, equal to asyncio's default. Don't change if you're unsure what it implies.

async login(*key: Optional[str] = None*)

Creates Websocket Socket client using the configuration and Logs to the stream with credentials. Primarily meant for internal uses. You shouldn't need to call this method manually as the streamer does it automatically behind the scenes

Returns
None

async _send(*data: str*)

Internal function to send data to websocket server endpoint

Parameters
data – The formatted data string to be sent.

Returns
None

async _recv()

Internal function to receive messages from websocket server endpoint.

Returns
The JSON decoded message data dictionary.

async handle_messages(*reconnect: bool = False*, *max_reconnection_attempts=5*, *reconnection_delay=5*)

The primary method to start the stream. Connects & Logs in by itself. Allows Reconnecting by simply altering a parameter (subscriptions are persisted across reconnected streams)

Parameters

- **reconnect** – If this is `False` (default), it simply awaits the next message and calls the appropriate handler. Uses the *_default_process_message()* if no handler was specified. You should use the statement inside a while loop in that case. Setting it to True creates an inner loop which traps disconnection errors except login failed due to invalid Key, and reconnects to the stream with the same subscriptions it had earlier before getting disconnected.
- **max_reconnection_attempts** – Determines how many times should the program attempt to reconnect in case of failed attempts. The Counter is reset as soon as a successful connection is re-established. Setting it to False disables the limit which is NOT recommended unless you know you got a situation. This value is ignored if `reconnect` is False (The default). Defaults to 5.
- **reconnection_delay** – Number of seconds to wait before attempting to reconnect after a failed reconnection attempt or a disconnection. This value is ignored if `reconnect` is False (the default). Defaults to 5.

Returns
None

async reconnect() → tuple

Reconnects the stream. Existing subscriptions (ones before disconnections) are persisted and automatically re-subscribed when reconnection succeeds. All the handlers are also automatically restored. Returns a tuple based on success status. While this instance method is supposed to be used internally, it is possible to utilize this in your your custom attempts of reconnection implementation. Feel free to share your implementations with the community if you find success :)

Returns
`(True, message)` if reconnection succeeds else `(False, message)`

async _default_process_message(*update*)

The default Handler for Message Streams which were NOT initialized with a handler function

Parameters
update – The update message as received after decoding the message.

Returns
None

_default_handlers_and_apis()

Assign default handler value to all stream setups. ONLY meant for internal use

async _modify_sub(*symbols: Optional[Union[str, list]]*, *action: str = 'subscribe'*, *_prefix: str = 'T.'*, *force_uppercase_symbols: bool = True*)

Internal Function to send subscribe or unsubscribe requests to websocket. You should prefer using the corresponding methods to subscribe or unsubscribe to stream.

Parameters

- **symbols** – The list of symbols to apply the actions to.

- **action** – Defaults to subscribe which subscribes to requested stream. Change to unsubscribe to remove an existing subscription.
- **_prefix** – prefix of the stream service. See *polygon.enums.StreamServicePrefix* for choices.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async subscribe_stock_trades(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time trades for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL tickers.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_stock_trades(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

async subscribe_stock_quotes(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time quotes for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL tickers.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_stock_quotes(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

async subscribe_stock_minute_aggregates(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Minute Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_stock_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

async subscribe_stock_second_aggregates(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Seconds Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_stock_second_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

async subscribe_stock_limit_up_limit_down(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time LULD Events for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.

- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_stock_limit_up_limit_down(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

async subscribe_stock_imbalances(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Imbalance Events for provided symbol(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_stock_imbalances(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied ticker symbols.

Parameters
symbols – A list of tickers to unsubscribe from. Defaults to ALL tickers.

Returns
None

async subscribe_option_trades(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options trades for provided ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_option_trades(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

async subscribe_option_quotes(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options quotes for provided ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_option_quotes(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

async subscribe_option_minute_aggregates(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options minute aggregates for given ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_option_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

async subscribe_option_second_aggregates(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time options second aggregates for given ticker(s)

Parameters

- **symbols** – A list of tickers to subscribe to. Defaults to ALL ticker. You can specify with or without the prefix `O:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_option_second_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied option symbols.

Parameters
symbols – A list of symbols to unsubscribe from. Defaults to ALL tickers. You can specify with or without the prefix `O:`

Returns
None

async subscribe_forex_quotes(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Forex Quotes for provided symbol(s)

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_forex_quotes(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied forex symbols.

Parameters
symbols – A list of forex tickers. Default is * which unsubscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

Returns
None

async subscribe_forex_minute_aggregates(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Forex Minute Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of forex tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_forex_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied forex symbols.

Parameters
symbols – A list of forex tickers. Default is * which unsubscribes to ALL tickers in the market. each Ticker must be in format: `from/to`. For example: `USD/CNH`.

Returns
None

async subscribe_crypto_trades(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Trades for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_crypto_trades(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

async subscribe_crypto_quotes(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Quotes for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_crypto_quotes(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

async subscribe_crypto_minute_aggregates(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Minute Aggregates for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_crypto_minute_aggregates(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

async subscribe_crypto_level2_book(*symbols: Optional[list] = None*, *handler_function=None*, *force_uppercase_symbols: bool = True*)

Get Real time Crypto Level 2 Book Data for provided symbol(s)

Parameters

- **symbols** – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`
- **handler_function** – The function which you'd want to call to process messages received from this subscription. Defaults to None which uses the default process message function.
- **force_uppercase_symbols** – Set to `False` if you don't want the library to make all symbols upper case

Returns
None

async unsubscribe_crypto_level2_book(*symbols: Optional[list] = None*)

Unsubscribe from the stream for the supplied crypto symbols.

Parameters
symbols – A list of Crypto tickers. Default is * which subscribes to ALL tickers in the market. each Ticker must be in format: `from-to`. For example: `BTC-USD`. you can pass symbols with or without the prefix `X:`

Returns
None

async change_handler(*service_prefix*, *handler_function*)

Change your handler function for a service. Can be used to update handlers dynamically while stream is running.

Parameters

- **service_prefix** – The Prefix of the service you want to change handler for. see *polygon.enums.StreamServicePrefix* for choices.
- **handler_function** – The new handler function to assign for this service

Returns
None

13.9 Enums Interface

class polygon.enums.**TickerMarketType**(*value*)

Market Types for method: `ReferenceClient.get_tickers()`

STOCKS = 'stocks'

OPTIONS = 'options'

FOREX = 'fx'

CRYPTO = 'crypto'

class polygon.enums.**TickerType**(*value*)

Ticker types for method: `ReferenceClient.get_tickers()`

CS = 'CS'

COMMON_STOCKS = 'CS'

ADRC = 'ADRC'

ADRP = 'ADRP'

ADRR = 'ADRR'

UNIT = 'UNIT'

RIGHT = 'RIGHT'

PFD = 'PFD'

FUND = 'FUND'

SP = 'SP'

WARRANT = 'WARRANT'

INDEX = 'INDEX'

ETF = 'ETF'

ETN = 'ETN'

class polygon.enums.**TickerSortType**(*value*)

Sort key for method: `ReferenceClient.get_tickers()`

TICKER = 'ticker'

NAME = 'name'

MARKET = 'market'

LOCALE = 'locale'

PRIMARY_EXCHANGE = 'primary_exchange'

TYPE = 'type'

ACTIVE = 'active'

CURRENCY_SYMBOL = 'currency_symbol'

CURRENCY_NAME = 'currency_name'

BASE_CURRENCY_SYMBOL = 'base_currency_symbol'

BASE_CURRENCY_NAME = 'base_currency_name'

CIK = 'cik'

COMPOSITE_FIGI = 'composite_figi'

SHARE_CLASS_FIGI = 'share_class_figi'

LAST_UPDATED_UTC = 'last_updated_utc'

DELISTED_UTC = 'delisted_utc'

class polygon.enums.SortOrder(*value*)

Order of sort. Ascending usually means oldest at the top. Descending usually means newest at the top. It is recommended to ensure the behavior in the corresponding function's docs. This enum can be used by any method accepting Sort order values.

ASCENDING = 'asc'

ASC = 'asc'

DESCENDING = 'desc'

DESC = 'desc'

class polygon.enums.TickerTypeAssetClass(*value*)

Asset Class for method: ReferenceClient.get_ticker_types_v3()

STOCKS = 'stocks'

OPTIONS = 'options'

FOREX = 'fx'

CRYPTO = 'crypto'

class polygon.enums.TickerNewsSort(*value*)

Sort key for method: ReferenceClient.get_ticker_news()

PUBLISHED_UTC = 'published_utc'

ALL = None

class polygon.enums.StockReportType(*value*)

Type of report for method: ReferenceClient.get_stock_financials()

YEAR = 'Y'

Y = 'Y'

YA = 'YA'

YEAR_ANNUALIZED = 'YA'

Q = 'Q'

QUARTER = 'Q'

QA = 'QA'

QUARTER_ANNUALIZED = 'QA'

T = 'T'

TRAILING_TWELVE_MONTHS = 'T'

TA = 'TA'

TRAILING_TWELVE_MONTHS_ANNUALIZED = 'TA'

class polygon.enums.StockFinancialsSortType(*value*)

Direction to use for sorting report for method: ReferenceClient.get_stock_financials()

REPORT_PERIOD = 'reportPeriod'

REVERSE_REPORT_PERIOD = '-reportPeriod'

CALENDAR_DATE = 'calendarDate'

REVERSE_CALENDAR_DATE = '-calendarDate'

class polygon.enums.StockFinancialsTimeframe(*value*)

Query by timeframe. Annual financials originate from 10-K filings, and quarterly financials originate from 10-Q filings. Note: Most companies do not file quarterly reports for Q4 and instead include those financials in their annual report, so some companies my not return quarterly financials for Q4 for method: `ReferenceClient.get_stock_financials_vx()`

ANNUAL = 'annual'

QUARTERLY = 'quarterly'

class polygon.enums.StockFinancialsSortKey(*value*)

Sort field for method: `ReferenceClient.get_stock_financials_vx()`

FILLING_DATE = 'filling_date'

PERIOD_OF_REPORT_DATE = 'period_of_report_date'

class polygon.enums.ConditionMappingTickType(*value*)

Tick Type for method: `ReferenceClient.get_condition_mappings()`

TRADES = 'trades'

QUOTES = 'quotes'

class polygon.enums.ConditionsDataType(*value*)

Type of data for method: `ReferenceClient.get_conditions()`

TRADE = 'trade'

BBO = 'bbo'

NBBO = 'nbbo'

class polygon.enums.ConditionsSIP(*value*)

SIP for method: `ReferenceClient.get_conditions()`

CTA = 'CTA'

UTP = 'UTP'

OPRA = 'OPRA'

class polygon.enums.ConditionsSortKey(*value*)

Sort key for method: `ReferenceClient.get_conditions()`

ASSET_CLASS = 'asset_class'

ID = 'id'

TYPE = 'type'

NAME = 'name'

DATA_TYPES = 'data_types'

LEGACY = 'legacy'

class polygon.enums.AssetClass(*value*)

Asset Class for methods: ReferenceClient.get_exchanges_v3() and ReferenceClient.get_conditions() and wherever needed.

STOCKS = 'stocks'

OPTIONS = 'options'

FOREX = 'fx'

CRYPTO = 'crypto'

class polygon.enums.Locale(*value*)

Locale name``

US = 'us'

GLOBAL = 'global'

class polygon.enums.SnapshotDirection

Direction to be supplied to the SnapShot - Gainers and Losers APIs on Stocks, Forex and Crypto endpoints

GAINERS = 'gainers'

GAIN = 'gainers'

LOSERS = 'losers'

LOSE = 'losers'

class polygon.enums.PaginationDirection(*value*)

The direction to paginate in.

NEXT = 'next'

FORWARD = 'next'

PREV = 'previous'

PREVIOUS = 'previous'

BACKWARD = 'previous'

class polygon.enums.StreamCluster(*value*)

The cluster to connect to. To be used for both callback and async stream client. NEVER connect to the same cluster again if there is an existing stream connected to it. The existing connection would be dropped and new one will be established. You can have up to 4 concurrent streams connected to 4 different clusters.

STOCKS = 'stocks'

OPTIONS = 'options'

FOREX = 'forex'

CRYPTO = 'crypto'

class polygon.enums.**OptionsContractType**(*value*)

Contract Type for method: `ReferenceClient.get_options_contracts()`

CALL = 'call'

PUT = 'put'

OTHER = 'other'

class polygon.enums.**OptionsContractsSortType**(*value*)

Sort field used for ordering for method: `ReferenceClient.get_options_contracts()`

TICKER = 'ticker'

UNDERLYING_TICKER = 'underlying_ticker'

EXPIRATION_DATE = 'expiration_date'

STRIKE_PRICE = 'strike_price'

class polygon.enums.**OptionTradesSort**(*value*)

Sort field used for ordering option trades. Used for method: `OptionsClient.get_trades`

TIMESTAMP = 'timestamp'

class polygon.enums.**OptionQuotesSort**(*value*)

Sort field used for ordering option quotes. Used for method: `OptionsClient.get_quotes`

TIMESTAMP = 'timestamp'

class polygon.enums.**StocksTradesSort**(*value*)

Sort field used for ordering Stocks trades. Used for method: `StocksClient.get_trades`

TIMESTAMP = 'timestamp'

class polygon.enums.**StocksQuotesSort**(*value*)

Sort field used for ordering Stocks quotes. Used for method: `StocksClient.get_quotes`

TIMESTAMP = 'timestamp'

class polygon.enums.**SplitsSortKey**(*value*)

Sort field used for ordering stock splits. Used for method `ReferenceClient.get_stock_splits`

EXECUTION_DATE = 'execution_date'

TICKER = 'ticker'

class polygon.enums.**PayoutFrequency**(*value*)

the number of times per year the dividend is paid out. Possible values are 0 (one-time), 1 (annually), 2 (bi-annually), 4 (quarterly), and 12 (monthly). used by method `ReferenceClient.get_stock_dividends`

ONE_TIME = 0

ANNUALLY = 1

BI_ANNUALLY = 2

QUARTERLY = 4

MONTHLY = 12

class polygon.enums.**DividendType**(*value*)

the type of dividend. Dividends that have been paid and/or are expected to be paid on consistent schedules are denoted as CD. Special Cash dividends that have been paid that are infrequent or unusual, and/or can not be expected to occur in the future are denoted as SC. Used for method `ReferenceClient.get_stock_dividends`

CD = 'CD'

SC = 'SC'

LT = 'LT'

ST = 'ST'

class polygon.enums.**DividendSort**(*value*)

sort field used for ordering dividend results. used for method `ReferenceClient.get_stock_dividends`

EX_DIVIDEND_DATE = 'ex_dividend_date'

PAY_DATE = 'pay_date'

DECLARATION_DATE = 'declaration_date'

RECORD_DATE = 'record_date'

CASH_AMOUNT = 'cash_amount'

TICKER = 'ticker'

class polygon.enums.**ForexQuotesSort**(*value*)

Sort field used for ordering Forex quotes. Used for method: `ForexClient.get_quotes`

TIMESTAMP = 'timestamp'

class polygon.enums.**CryptoTradesSort**(*value*)

Sort field used for ordering crypto trades. Used for method: `CryptoClient.get_trades`

TIMESTAMP = 'timestamp'

class polygon.enums.**StreamHost**(*value*)

Host to be used for stream connections. WHY on earth would you use delayed if you're paying for polygon??

REAL_TIME = 'socket.polygon.io'

DELAYED = 'delayed.polygon.io'

class polygon.enums.**StreamServicePrefix**(*value*)

Service Prefix for Stream endpoints. To be used for method: `AsyncStreamClient.async change_handler()`

STOCK_TRADES = 'T'

STOCK_QUOTES = 'Q'

STOCK_MINUTE_AGGREGATES = 'AM'

STOCK_SECOND_AGGREGATES = 'A'

STOCK_LULD = 'LULD'

STOCK_IMBALANCES = 'NOI'

FOREX_QUOTES = 'C'

FOREX_MINUTE_AGGREGATES = 'CA'

CRYPTO_TRADES = 'XT'

CRYPTO_QUOTES = 'XQ'

CRYPTO_LEVEL2 = 'XL2'

CRYPTO_MINUTE_AGGREGATES = 'XA'

STATUS = 'status'

OPTION_TRADES = 'T'

OPTION_QUOTES = 'Q'

OPTION_MINUTE_AGGREGATES = 'AM'

OPTION_SECOND_AGGREGATES = 'A'

class polygon.enums.**Timespan**(*value*)

The timespan values. Usually meant for aggregates endpoints. It is best to consult the relevant docs before using any value on an endpoint.

MINUTE = 'minute'

MIN = 'minute'

HOUR = 'hour'

DAY = 'day'

WEEK = 'week'

MONTH = 'month'

QUARTER = 'quarter'

YEAR = 'year'

class polygon.enums.**OptionSymbolFormat**(*value*)

Option symbol formats supported by the library. To be used with functions to build or parse option symbols

POLYGON = 'polygon'

TDA = 'tda'

TD_AMERITRADE = 'tda'

TOS = 'tos'

THINK_OR_SWIM = 'tos'

TRADIER = 'tradier'

TRADE_STATION = 'trade_station'

```
IB = 'ibkr'

IBKR = 'ibkr'

INTERACTIVE_BROKERAGE = 'ibkr'
```

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

B

C

D

H

I

L

M

N

O

P

Q

R

S

T

U

W

Y