MyGeotab Python SDK Documentation

Release 0.8.6

Aaron Toth

Mar 23, 2021

Contents

An Apache2 Licensed, unofficial Python client for the MyGeotab SDK.

Also bundled is the "myg" command line tool, which is a sandboxed console for quickly querying and operating on MyGeotab data.

CHAPTER 1

Features

- Automatic serializing and de-serializing of JSON results
- Clean, Pythonic API for querying data
- Cross-platform and compatible with Python 2.7.9+ and 3.4+

CHAPTER 2

Usage

It's very easy to get started once you've registered with MyGeotab:

```
import mygeotab

client = mygeotab.API(username='hello@example.com', password='mypass', database=
↪'MyDatabase')
client.authenticate()

devices = client.get('Device', name='%Test Dev%')

print(devices)

# [{'maxSecondsBetweenLogs': 200.0,
#   'activeTo': '2050-01-01',
#   'minAccidentSpeed': 3.0,
#   'ignoreDownloadsUntil': '1986-01-01',
#   'name': 'Test Device',
#   'idleMinutes': 3.0,
#   ......
```

If you're using Python 3.5 and higher, you can also make calls asynchronously via asyncio:

```
import asyncio
import mygeotab

client = mygeotab.API(username='hello@example.com', password='mypass', database=
↪'MyDatabase')
client.authenticate()

async def get_device():
  return await client.get_async('Device', name='%Test Dev%')

devices = loop.run_until_complete(get_device())
print(devices)
```

(continues on next page)

(continued from previous page)

```
# [{'maxSecondsBetweenLogs': 200.0,
#   'activeTo': '2050-01-01',
#   'minAccidentSpeed': 3.0,
#   'ignoreDownloadsUntil': '1986-01-01',
#   'name': 'Test Device',
#   'idleMinutes': 3.0,
#   ......
```

CHAPTER 3

Installation

To install the MyGeotab library and command line tool:

```
$ pip install mygeotab
```

or for the bleeding-edge version:

```
$ pip install git+https://github.com/geotab/mygeotab-python
```

CHAPTER 4

Documentation

Read the docs at http://mygeotab-python.readthedocs.org

CHAPTER 5

Changes

5.1 0.8.6 (2021-03-15)

Improvements

- Added new experimental API class (in the *mygeotab.ext* module) for more intuitive ways of using results from *client.get()* like *get_dataframe()* to easily get a Pandas DataFrame from the result set.
- Adds support for using with proxies #327.
- Calling *authenticate()* on an API instance more than once no longer results in an error #328.

Bug Fixes

- Handle *ConnectionError* as errors in feed extension rather than crashing #130.
- Remove extraneous "search" parameter from *API.get()*.

5.2 0.8.5 (2019-10-07)

Bug Fixes

- Fixes issue with warnings from Arrow about date parsing #306.

5.3 0.8.4 (2019-08-22)

Bug Fixes

- Fixes issue with serialization of datetime.date objects #284.

5.4 0.8.3 (2019-08-19)

Improvements

- Use the high-performance python-rapidjson library to serialize and deserialize JSON parameters and responses in Python 3.5+ #268.

Bug Fixes

- Silence warnings from arrow parsing when the library is used interactively or in a Jupyter notebook.

Housecleaning

- Added serialization benchmarking in CircleCI tests.
- Remove PyPy test config.

5.5 0.8.2 (2019-06-10)

Bug Fixes

- Remove asyncio-specific default arguments preventing from importing this pacakge in a Python 3.5+ thread #236.

5.6 0.8.1 (2019-06-03)

Improvements

- Return content with a non-JSON content type as text

Bug Fixes

- Handle serialization of very old dates #121.

5.7 0.8.0 (2018-06-18)

Improvements

- Python 3.7 support.
- Raises an exception when request was not successful.
- Documentation improvements.

Bug Fixes

- Since all MyGeotab servers enforce the use of TLS1.2, Python 2.7.9 or greater is required.
- Fix issue where the password was not provided when retrying authentication. Should better handle #92.

5.8 0.6.2 (2017-07-04)

Bug Fixes

- Revert the change to stop compilation in setup.cfg.

5.9 0.6.1 (2017-07-03)

Bug Fixes

- Don't compile to prevent issues when installing via setup.py on Python 2.7.

5.10 0.6.0 (2017-06-29)

Improvements

- Configurable timeouts when making calls.
- Removed *verify* parameter from API objects as SSL is required when calling a MyGeotab server.
- Removed *run* command from the CLI.
- Removed deprecated *API.search* and *API.search_async* methods.
- Refactored setup.py for async API. The async/awaitable methods are now automatically a part of the *API* object if using Python 3.5 or higher
- Code linting and cleanup

5.11 0.5.4 (2017-06-05)

Bug Fixes

- Ensure all dates are timezone aware and are always UTC-localized.

5.12 0.5.3 (2017-05-30)

Bug Fixes

- Fixed intermittent timeout errors due to upstream changes in the 'requests' module

5.13 0.5.2 (2017-02-02)

Bug Fixes

- Switched back to using abstract dependencies in setup.py (recommended by this guide)

5.14 0.5.1 (2017-01-04)

Bug Fixes

- Fix for search parameter not being properly handled in 'get()' call

5.15 0.5 (2017-01-02)

Enhancements

- Deprecated the 'search()' and 'search_async()' functions. Replaced by folding the previous functionality into 'run()'.
- Removed 'tzlocal' dependency. Always deal with dates in UTC by default.
- Prefer functions instead of making static methods in classes.
- Added helper to run async calls and collect their results
- Add ability to quickly run simple python scripts from the 'myg' console with no need for any authentication handling. Similar to 'console', but for running scripts rather than creating an interactive console.

5.16 0.4.4 (2016-07-10)

Enhancements

- Added the ability to make unauthenticated calls (like "GetVersion") with the static "API.server_call" method
- Added asyncio-based API query methods (Python 3.5+ only) into the "ext" package
- Moved the datafeed to the "ext" package, as well

Bug Fixes

- MyGeotab never returns 3 digits of milliseconds, so follow that format as well to allow the use of "dates.format_iso_datetime" to create MyGeotab URLs

5.17 0.4.2 (2016-03-17)

Bug Fixes

- Use a custom User-Agent when making requests

5.18 0.4 (2016-02-25)

Enhancements

- Extension for facilitating use of the MyGeotab Data Feed
- Allow Pythonic underscore-separated parameters mapped to camelcase ones
- Force the use of TLS 1.2 for upcoming strict security requirements in MyGeotab (Note that TLS 1.2 is only supported in Python 2.7.9+ and 3.4+)

Bug Fixes

- Fixed issue with CLI console startup
- Use the system's default user location for config files

CHAPTER 6

User Guide

How to use this library, and more information about bundled tools and examples.

6.1 Usage

6.1.1 Getting Started

For a quick introduction to the MyGeotab SDK and initial setup of a database, please refer to the Getting Started guide.

For an overview of some basic concepts, the Concepts guide is a good resource to find out how things work under the hood.

6.1.2 Authentication

The first step is to authenticate with a MyGeotab database. Import the library and create an `API` object:

```
import mygeotab
api = mygeotab.API(username='hello@example.com', password='mypass', database='DemoDB')
api.authenticate()
```

Note: If the intended purpose for this SDK application is for an end-user service where they can log in, consider catching the `AuthenticationException` and handle it (ex. displaying an error message to the user):

```
try:
    api.authenticate()
except mygeotab.AuthenticationException as ex:
    # Handle authentication issues here
    print(ex) # Cannot authenticate 'hello@example.com @ my.geotab.com/DemoDb'
```

To handle the saving of credentials for later use (a Geotab best practices recommendation), the `authenticate()` method returns an instance of the `Credentials` object. From this, store the *server*, *database*, *username*, and *session_id* properties so they can be used later:

```
credentials = api.authenticate()
my_save_credentials(username=credentials.username, database=credentials.database,␣
↪server=credentials.server, session_id=credentials.session_id)

# Continue with api object until your app finishes

local_credentials = my_read_credentials() # Next load of the app
new_api = mygeotab.api(username=local_credentials.user, database=local_credentials.
↪database, server=local_credentials.server, session_id=saved_session_id)
```

Note: The best practices of saving credentials only applies to some service-based SDK apps. The recommendation is that if the app runs on a schedule (for example, a operating system-scheduled task running every minute), store the credentials locally.

Too many authentication attempts within a period of time will cause the server to reject any further requests for a short time.

However, constantly running sessions may not need to store the credentials in the file system as they can retain the `API` instance in memory.

Subsequent calls to the `authenticate()` method with an already authenticted `API` object can extend the lifetime of the session. However, this still counts towards the count of authentication attempts and calling this may still result in ‘OverLimitException‘s and may reject requests for a short time afterward to prevent API abuse.

6.1.3 Making Calls

At the core of every interaction with the MyGeotab API is the `call()` method, which executes a secure HTTPS call to the MyGeotab server.

The most basic call is to get the version of MyGeotab that the server is running, which doesn't take any parameters:

```
api.call('GetVersion')
# '5.7.1610.229'
```

To demonstrate a (slightly) more complex call with 1 parameter, the following is a query for all the vehicles in a database.

Assume for this example there is one vehicle in the system, with a partial JSON representation:

```
{
    "id": "b0a46",
    "name": "007 - Aston Martin",
    "serialNumber": "GTA9000003EA",
    "deviceType": "GO6",
    "vehicleIdentificationNumber": "1002",
    ...
}
```

Get a list of all the vehicles by using:

```
api.call('Get', typeName='Device')
```

To filter this down to a specific vehicle, a 'search' parameter is added on the serial number of the GO device:

```
api.call('Get', typeName='Device', search={'serialNumber': 'GTA9000003EA'})
```

Note: In this Python library, a lot of effort was made to make this a much easier experience. Please read the below section to see how the above call was made to be more Pythonic and easier to use.

For more information on calls available, visit the "Methods" section of the MyGeotab API Reference.

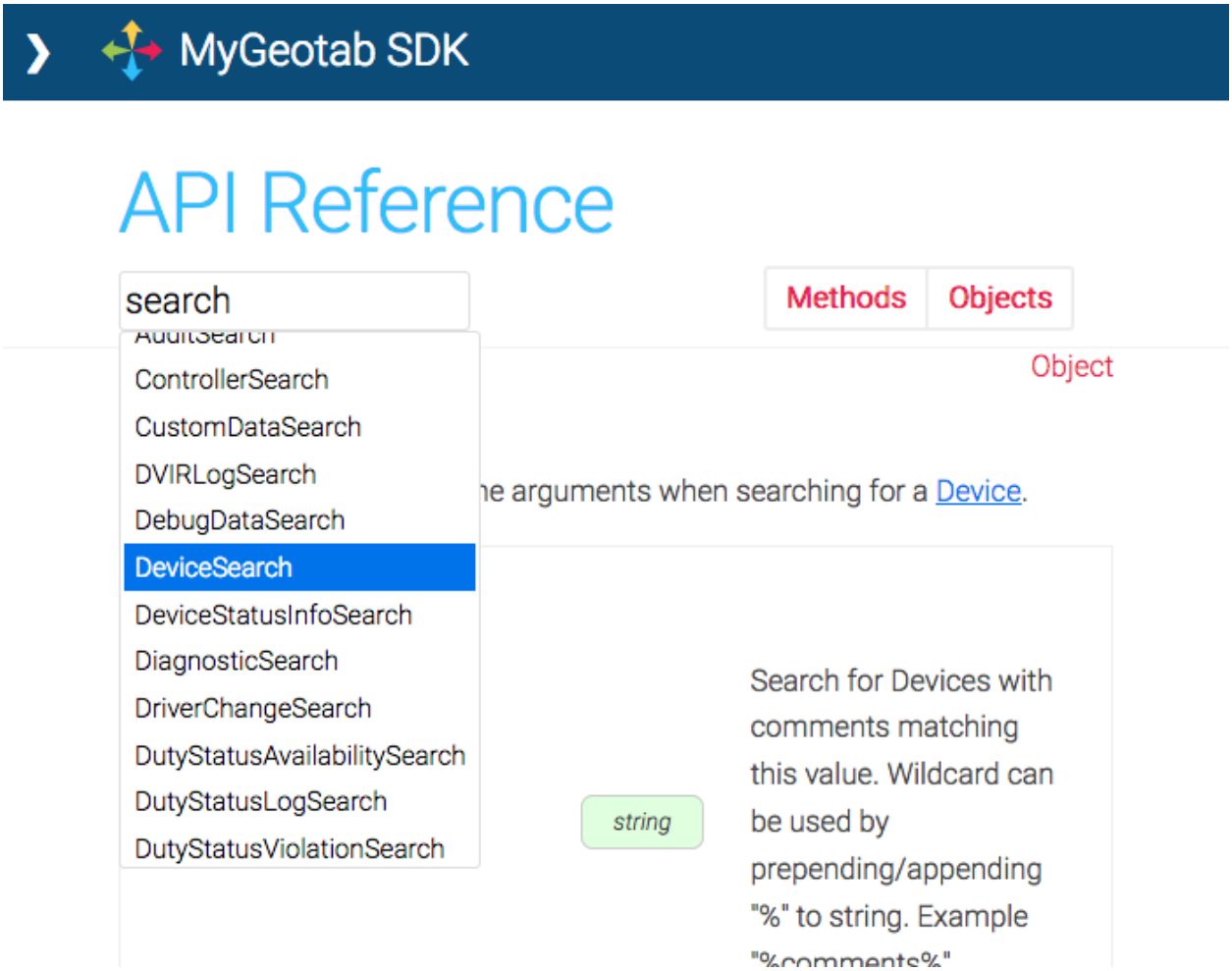
6.1.4 Entities

From the MyGeotab API Concepts documentation:

> All objects in the MyGeotab system are called entities. Entities have an ID property that is used to uniquely identify that object in the database.

To see all available entities, refer to the API _MyGeotab API Reference.

Note: To see which objects are entities in the SDK, type in "search" into the search box of the API reference page.



For example, the "Device" object has a corresponding "DeviceSearch", and the "User" object has a corresponding "UserSearch" object.

There are several helper methods added in this SDK library that do some wrapping around the `call()` method to make it more Pythonic and easier to work with.

Getting

To re-use the above example vehicle of getting all vehicles, the `get()` method is much more concise:

```
api.get('Device')
```

This also simplifies the filtering down to the specific vehicle:

```
api.get('Device', serialNumber='GTA9000003EA')
```

Note: Because the "search" parameter is common in a call, the library brings all parameters that can be passed into a search to the top level parameters for the `get()` method.

Adding

To add an entity, use the `add()` method:

```
api.add('Device', {
    'serialNumber': 'GTA9000003EA',
    'name': 'My Vehicle'
})
```

Setting

To modify an entity, first get the full entity:

```
devices = api.get('Device', serialNumber='GTA9000003EA', resultsLimit=1)
device = devices[0]
```

Note: The the `get()` method always returns a list of entities, even when querying on a specific serial number or VIN, etc.

Then modify a property:

```
device['name'] = 'My New Vehicle'
```

And then call `set()`:

```
api.set('Device', device)
```

Removing

To remove the entity, once again get the full entity, as above in *Setting*, and then call the `remove()` method:

```
api.remove('Device', device)
```

6.2 Command Line Tools

The *myg* command line script is installed alongside this package, which currently makes some administration and querying simple, as it handles the credential storage and token expiry automatically.

Currently, the script launches an interactive console to quickly query data from a database. More functionality will be added in the future.

The tools never store passwords. The username, session token, and database are persisted and managed in the local user's data directory.

6.2.1 Usage

The most common usage of the *myg* script is to launch an interactive console.

For example, to launch a console for a database called *my_database*:

```
$ myg console my_database
Username: my_user
Password: ******
```

Note: The *myg* script automatically handles storing credentials for various databases and remembers the last logged in database. It also handles session expiry: it will prompt for a new password if the session has expired.

For example, once the database has been authenticated against, the script won't prompt for passwords until the sesison expires:

```
$ myg console my_database
MyGeotab Console 0.5.1 [Python 3.5.2 |Anaconda custom (x86_64)| (default, Jul  2␣
↪2016, 17:52:12) [GCC 4.2.1 Compatible Apple LLVM 4.2 (clang-425.0.28)]]
Logged in as:  my_user @ my1.geotab.com/my_database
In [1]:
```

If *my_database* was the last logged in database, the following also works:

```
$ myg console
```

To view current sessions:

```
$ myg sessions
my_database
my_other_database
```

And to remove a session:

```
$ myg sessions remove my_database
my_other_database
```

6.2.2 Additional Help

Run *–help* after any command to get available options and descriptions.

```
$ myg --help
$ myg console --help
```

6.3 Extras

Extras for easier querying of the MyGeotab API. Note that these are currently experimental, and likely to change at any time without notice.

6.3.1 Data Feed

Handler to get data as efficiently as possible.

A simple example package can be found in the */examples* directory.

See the API reference for the `DataFeed` and `DataFeedListener` classes for more information.

CHAPTER 7

API Reference

The full API reference for all public classes and functions.

7.1 API

The full API reference for all public classes and functions.

7.1.1 Querying Data

class mygeotab.api.**API**(*username*, *password=None*, *database=None*, *session_id=None*, *server='my.geotab.com'*, *timeout=300*, *proxies=None*)

A simple and Pythonic wrapper for the MyGeotab API.

add(*type_name*, *entity*)

Adds an entity using the API. Shortcut for using call() with the 'Add' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **entity** (*dict*) – The entity to add.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

Returns The id of the object added.

Return type str

authenticate()

Authenticates against the API server.

Raises

- **AuthenticationException** – Raises if there was an issue with authenticating or logging in.
- **MyGeotabException** – Raises when an exception occurs on the MyGeotab server.
- **TimeoutException** – Raises when the request does not respond after some time.

Returns A Credentials object with a session ID created by the server.

Return type *Credentials*

call(*method*, ***parameters*)
Makes a call to the API.

Parameters

- **method** (*str*) – The method name.
- **parameters** – Additional parameters to send (for example, search=dict(id='b123')).

Raises

- **MyGeotabException** – Raises when an exception occurs on the MyGeotab server.
- **TimeoutException** – Raises when the request does not respond after some time.

Returns The results from the server.

Return type dict or list

static from_credentials(*credentials*)
Returns a new API object from an existing Credentials object.

Parameters **credentials** (Credentials) – The existing saved credentials.

Returns A new API object populated with MyGeotab credentials.

Return type *API*

get(*type_name*, ***parameters*)
Gets entities using the API. Shortcut for using call() with the 'Get' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **parameters** – Additional parameters to send.

Raises

- **MyGeotabException** – Raises when an exception occurs on the MyGeotab server.
- **TimeoutException** – Raises when the request does not respond after some time.

Returns The results from the server.

Return type list

multi_call(*calls*)
Performs a multi-call to the API.

Parameters **calls** (*list((str, dict))*) – A list of call 2-tuples with method name and params (for example, ('Get', dict(typeName='Trip'))).

Raises

- **MyGeotabException** – Raises when an exception occurs on the MyGeotab server.
- **TimeoutException** – Raises when the request does not respond after some time.

Returns The results from the server.

Return type list

remove(*type_name*, *entity*)
Removes an entity using the API. Shortcut for using call() with the 'Remove' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **entity** (*dict*) – The entity to remove.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

set(*type_name*, *entity*)
Sets an entity using the API. Shortcut for using call() with the 'Set' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **entity** (*dict*) – The entity to set.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

class mygeotab.api.**MyGeotabException**(*full_error*, **args*, ***kwargs*)
There was an exception while handling your call.

class mygeotab.api.**TimeoutException**(*server*, **args*, ***kwargs*)
The request timed out while handling your request.

7.1.2 Credentials & Authentication

class mygeotab.api.**Credentials**(*username*, *session_id*, *database*, *server*, *password=None*)
The MyGeotab Credentials object.

get_param()
A simple representation of the credentials object for passing into the API.authenticate() server call.

Returns The simple credentials object for use by API.authenticate().

Return type dict

class mygeotab.api.**AuthenticationException**(*username*, *database*, *server*, **args*, ***kwargs*)
Unsuccessful authentication with the server.

7.1.3 Date Helpers

mygeotab.dates

Date helper objects for timezone shifting and date formatting for the MyGeotab API.

mygeotab.dates.**format_iso_datetime**(*datetime_obj*)
Formats the given datetime as a UTC-zoned ISO 8601 date string.

Parameters **datetime_obj** (*datetime*) – The datetime or date object.

Returns The datetime object in 8601 string form.

Return type datetime

mygeotab.dates.**localize_datetime**(*datetime_obj*, *tz=<UTC>*)
Converts a naive or UTC-localized date into the provided timezone.

Parameters

- **datetime_obj** (*datetime*) – The datetime object.
- **tz** (*datetime.tzinfo*) – The timezone. If blank or None, UTC is used.

Returns The localized datetime object.

Return type datetime

7.1.4 Extras

EntityList

class mygeotab.ext.entitylist.**API**(*username*, *password=None*, *database=None*, *session_id=None*, *server='my.geotab.com'*, *timeout=300*, *proxies=None*)
An experimental wrapper around the base MyGeotab API class that adds some helper methods to results when retrieving results with *get()*.

add(*type_name*, *entity*)
Adds an entity using the API. Shortcut for using call() with the 'Add' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **entity** (*dict*) – The entity to add.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

Returns The id of the object added.

Return type str

authenticate()
Authenticates against the API server.

Raises

- ***AuthenticationException*** – Raises if there was an issue with authenticating or logging in.
- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

Returns A Credentials object with a session ID created by the server.

Return type *Credentials*

call(*method*, ***parameters*)
Makes a call to the API.

Parameters

- **method** (*str*) – The method name.
- **parameters** – Additional parameters to send (for example, search=dict(id='b123')).

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

Returns The results from the server.

Return type dict or list

static from_credentials(*credentials*)

Returns a new API object from an existing Credentials object.

Parameters **credentials** (Credentials) – The existing saved credentials.

Returns A new API object populated with MyGeotab credentials.

Return type *API*

get(*type_name*, ***parameters*)

Gets entities using the API. Shortcut for using call() with the 'Get' method. This returns an EntityList with added convience methods.

Parameters

- **type_name** (*str*) – The type of entity.
- **parameters** – Additional parameters to send.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

Returns The results from the server.

Return type *EntityList*

multi_call(*calls*)

Performs a multi-call to the API.

Parameters **calls** (*list((str, dict))*) – A list of call 2-tuples with method name and params (for example, ('Get', dict(typeName='Trip'))).

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

Returns The results from the server.

Return type list

remove(*type_name*, *entity*)

Removes an entity using the API. Shortcut for using call() with the 'Remove' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **entity** (*dict*) – The entity to remove.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

set(*type_name*, *entity*)
Sets an entity using the API. Shortcut for using call() with the 'Set' method.

Parameters

- **type_name** (*str*) – The type of entity.
- **entity** (*dict*) – The entity to set.

Raises

- ***MyGeotabException*** – Raises when an exception occurs on the MyGeotab server.
- ***TimeoutException*** – Raises when the request does not respond after some time.

class mygeotab.ext.entitylist.**EntityList**(*data*, *type_name*)
The customized result list

entity
Like *first*, but first asserts that there is only one entity in the results list.

Return type dict

first
Gets the first entity in the list, if it exists.

Return type dict

last
Gets the last entity in the list, if it exists.

Return type dict

sort_by(*key*, *reverse=False*)
Returns an EntityList, sorted by a provided key.

Parameters

- **key** (*str*) – The key to sort the data with.
- **reverse** (*bool*) – If true, reverse the sort direction.

Return type *EntityList*

to_dataframe(*normalize=False*)
Transforms the data into a pandas DataFrame

Parameters **normalize** (*bool*) – Whether or not to normalize any nested objects in the results into distinct columns.

Return type pandas.DataFrame

Data Feed

class mygeotab.ext.feed.**DataFeed**(*client_api*, *listener*, *type_name*, *interval*, *search=None*, *results_limit=None*)
A simple wrapper for the MyGeotab Data Feed. Create a listener that inherits from DataFeedListener to pass in.

start(*threaded=True*)
Start the data feed.

Parameters **threaded** – If True, run in a separate thread.

class mygeotab.ext.feed.**DataFeedListener**
The abstract DataFeedListener to override

on_data(*data*)
Called when rows of data are received.

Parameters **data** – A list of data objects.

on_error(*error*)
Called when server errors are encountered. Return False to close the stream.

Return type bool

Parameters **error** – The error object.

Returns If True, keep listening. If False, stop the data feed.

Python Module Index

m

Index