Pynationstates Documentation

Release 3.0.1.6

Joshua W

Dec 28, 2020

Contents

1 Content Pages 1

CHAPTER 1

Content Pages

1.1 Getting Started

The Nationstates package includes a simple interface for developers.

For example, for the following code:

```
>>> import nationstates
>>> api = nationstates.Nationstates(UniqueAndDescriptiveUserAgent)
>>> nation = api.nation("testlandia")
```

The *.Nationstates* object centralizes API use. The *.nation* method creates a *Nation* object for the *'testlandia'*.

1.2 Interacting with the API

There are multiple ways to interact with the api

Examples: Using the *flag* and *happenings* shard

This module supports direct attribute access for most of the shards. (For this example, you can check your shard by viewing *nation.auto_shards*), along side other methods.

```
>>> nation.flag #Returns Flag directly
>>> nation.get_flag() # see happenings for better usage
>>> nation.get_shards("flag")
```

Happenings:

```
>>> world = api.world()
>>> world.happenings # You are unable to pass any additional arguments with attribute␣
↪request
>>> world.get_happenings(filter=["law", "change", "rmb"], limit=100)
```

1.3 Interacting with the response

Outside of shard requests with *.get_shards*, all methods/attributes that referance a shard will only return that shard.

```
>>> resp = world.numnations
"183000"
>>> resp = world.get_numnations()
"183000"
>>> resp = world.get_shards('numnations')
{"numnations":'183000'}
```

get_shards returns a specialized dict, so code like this can be used

```
>>> resp.numnations
"183000"
```

This package includes multiple options for how to request/interact with the data, the default is through a specialize dictionary

```
>>> import nationstates
>>> api = nationstates.Nationstates(UniqueAndDescriptiveUserAgent)
>>> nation = api.nation("testlandia")
>>> result = nation.get_shards("region", "fullname") #supports multiple shards
{"region":"Example", fullname:"Example"}
```

You can interact with it in a multiple ways

```
>>> result.fullname
"Example"
>>> result["fullname"]
"Example"
```

Additionally, all api's have support for Auto Shards, and get_shardname() methods (which returns slightly more data)

```
>>> nation.fullname
"The Hive Mind of Testlandia"
>>> nation.get_fullname()
{'id': 'testlandia', 'fullname': 'The Hive Mind of Testlandia'}
```

You can view what each endpoint currently supports in it's magic methods by viewing

```
>>> nation.auto_shards
('admirable', 'animal', 'animaltrait', 'banner', 'banners', 'capital', 'category',
↪'census', 'crime', 'currency', 'customleader', 'customcapital', 'customreligion',
↪'dbid', 'deaths', 'demonym', 'demonym2', 'demonym2plural', 'dispatches',
↪'dispatchlist', 'endorsements', 'factbooks', 'factbooklist', 'firstlogin', 'flag',
↪'founded', 'foundedtime', 'freedom', 'fullname', 'gavote', 'gdp', 'govt', 'govtdesc
↪', 'govtpriority', 'happenings', 'income', 'industrydesc', 'influence',
↪'lastactivity', 'lastlogin', 'leader', 'legislation', 'majorindustry', 'motto',
↪'name', 'notable', 'policies', 'poorest', 'population', 'publicsector', 'rcensus',
↪'region', 'religion', 'richest', 'scvote', 'sectors', 'sensibilities', 'tax',
↪'tgcanrecruit', 'tgcancampaign', 'type', 'wa', 'wabadges', 'wcensus')
>>> nationstates.get_shard
{'get_customreligion', 'get_rcensus', 'get_banners', 'get_govtpriority', 'get_banner',
↪ 'get_census', 'get_gavote', 'get_wcensus', 'get_firstlogin', 'get_notable', 'get_
↪admirable', 'get_foundedtime', 'get_category', 'get_customleader', 'get_flag', 'get_
↪currency', 'get_endorsements', 'get_lastlogin', 'get_region', 'get_religion', 'get_
↪capital', 'get_name', 'get_type', 'get_happenings', 'get_crime', 'get_govtdesc',
↪'get_majorindustry', 'get_influence', 'get_customcapital', 'get_tax', 'get_
↪tgcanrecruit', 'get_demonym2', 'get_legislation', 'get_poorest', 'get_wa', 'get_
↪sectors', 'get_deaths', 'get_dbid', 'get_policies', 'get_scvote', 'get_lastactivity
↪', 'get_demonym', 'get_freedom', 'get_animal', 'get_factbooklist', 'get_industrydesc
↪', 'get_income', 'get_population', 'get_founded', 'get_richest', 'get_demonym2plural
↪', 'get_gdp', 'get_dispatches', 'get_publicsector', 'get_fullname', 'get_motto',
↪'get_tgcancampaign', 'get_govt', 'get_sensibilities', 'get_dispatchlist', 'get_
```

(continues on next page)

(continued from previous page)

1.4 Is pynationstates thread safe?

pynationstates does have locks in place that should make it safe to use across threads. However, the author recommends you have independent *nationstates.Nationstates* instances for earch thread. This guarntees everything will work.

1.5 Shards

The Nationstates package has multiple ways to represent shards, Strings and the `Shard` Object.

1.5.1 Using Strings

Strings can be used to represent simpler shards.

```
>>> import nationstates as ns
>>> api = ns.Nationstates(UniqueAndDescriptiveUserAgent)
>>> r = api.world().get_shards("numnations")
```

1.5.2 Shard Object

The `Shard` Object was built to allow more complicated shards. They allow you to attach parameters to the shard as well as just plainly representing a shard. It is recommended for more dynamic/advanced usage of the module.

```
>>> import nationstates as ns
>>> api = ns.Nationstates(UniqueAndDescriptiveUserAgent)
>>> happenings = ns.Shard("happenings", view="region.the_pacific", filter="founding")
>>> r = api.world().get_shards(happenings)
```

1.6 Using Private Requests

Private Shards and command, which are currently only supported the api for Nations

> **Warning:** It's not secure to have your password simply sitting in your source code, you may want use various config libraries or other means to get the password at runtime, rather than directly using it.

Using them is as simple as providing either a password or pin

```
>>> import nationstates
>>> api = nationstates.Nationstates(UniqueAndDescriptiveUserAgent)
>>> nation = api.nation("testlandia", password=yourpassword)
```

Additionally you may use a PIN

Warning: if you authenticate elsewhere (including the user facing website), this library will have no way to recover without a password. Using a password automatically handles the api's auth, while a pin will be reset if you log into nationstates.

Example

```
>>> import nationstates
>>> api = nationstates.Nationstates(UniqueAndDescriptiveUserAgent)
>>> nation = api.nation("testlandia", pin=yourpin)
```

Once you have this setup, the methods *.command()* will be functional, and you can send private shards through .'get_shards()'

1.7 Commands

Once a nation is authenticated, you will have access to commands. *.command()* will implement the basic structure of a command request.

pynationstates supplies some alias methods, like *.pickissue()*, *create_dispatch()*, *.edit_dispatch()*, and *.remove_dispatch()* that make it easier to support some of the more complicated commands.

1.8 Rate limiter

The Nationstates package constains an automatic ratelimiter. It's a simple implementation that works for scripts, but you may wish to disable it and use your own system if need be.

1.8.1 How to use the Rate Limiter

The Rate Limit system is completely automatic, and will simply `time.sleep()` your thread if it thinks your getting close to the api limit. Each thread keeps tracks of its own requests, so it can leave this period as soon as possible. Due to the possibility of multiple threads, this library slightly lowers the amount of requests per 30 second block to prevent accidental breaches of the ratelimit.

1.8.2 Disabling the Rate Limiter

If you are using your own rate limit code, it's recommend you disable this system as it may interfere and unexpectly sleep your thread.

```
>>> import nationstates
>>> api = nationstates.Nationstates(ratelimit_enabled=False)
```

Additionally you may wish to remove some other functionality like the retry system

```
>>> api = nationstates.Nationstates(ratelimit_enabled=False, do_retry=False)
```

Or you can keep using the ratelimit tracker, and just disable sleeping the thread. This means when the code would have sleep, it will raise a *RateLimitReached* exception

All of the library is centralized around the `Nationstates` module, so it's recommended you just have one. Do note though, that if you have mulitple *Nationstates* objects you'll need to pass the ratelimit argument for each. Additionally, some requests like commands will sometimes use multiple requests if required too by the api

1.8.3 Telegrams

Telegram rate limits have not been implemented in the ratelimiter, they act as the same as a request and affect the main tracker, but this library does not independently track telegram ratelimits.