Azion API Documentation

Release 0.1

Mauricio Antunes

Jun 13, 2018

Contents

azion-python is a library to interact with *Azion's ReST API*.

Note: This client only supports Python 3. I recommend you to upgrade your systems to use Python 3.

With this library you will be able to use the ReST API using a pythonic approach, handling Python objects (models) instead of raw JSON responses.

```
from azion import authorize, login

# Retrieve a new token
auth = authorize('myemail@mail.com', 'mysecretpassword')

# Login using the token
azion = login(auth.token)

# Retrieve a configuration
configuration = azion.get_configuration(1028910)

print(configuration.name)
```

You can checkout more examples throught the documentation.

CHAPTER 1

Configurations examples

Here are some examples so you don't have to read the source to understand how to use this client.

```
from azion import authorize, login

# Authorize and login
auth = authorize('myemail@mail.com', 'mysecretpassword')
azion = login(auth.token)

# Create a configuration
azion.create_configuration(
    'My cool configuration',
    'www.myorigin.com',
    'www.myhostheader.com'
)
```

CHAPTER 2

Purge examples

If you need to remove content from Azion cache before it expires, use the Purge API. You can use it to expire content based on your own business rules.

2.1 Purge by URL

```
from azion import authorize, login

# Authorize and login
auth = authorize('myemail@mail.com', 'mysecretpassword')
azion = login(auth.token)

# Purge by URL using CNAME and Domain name
# These two URLs will be purged from Azion cache
my_urls = [
    'www.maugzoide.com/foobar.jpg'
    '11111a.ha.azioncdn.net/test.js'
]

azion.purge_url(urls)
```

This endpoint answers with HTTP 207 (WebDAV). It is a multi-status code designed to represent an answer that was partially OK. In the result you can find that the dictionary keys are the status and the keys *urls* and *details* will exist for every status code. To filter for responses that succeed/failed, we provide two methods:

```
result = azion.purge_url(urls)

# URLs that were purged
result.succeed()

# URLs that were not purged
result.failed()
```

2.2 Purge by Cache Key

You can purge using a cache key. A common example is purging an image. Purging a URL like static.yourdomain.com/images/image.jpg@@ will include all variations found after *image.jpg*

```
my_urls = [
    'www.maugzoide.com/profile.jpg@@'
    '11111a.ha.azioncdn.net/@@cookie_name=foobar'
]

azion.purge_cache_key(urls)
```

2.3 Purge Wildcard

Use a wildcard purge to delete a list of objects from the cache. This function accepts one URL only.

```
url = 'www.maugzoide.com/static/img/*'

azion.purge_wildcard(url)
```

CHAPTER 3

Installation

```
$ pipenv install azion-python
```

CHAPTER 4

API

4.1 API Reference

4.1.1 Azion client

Azion is the main entrypoint to work with Azion's ReST API.

Azion

class azion.client.**Azion**(*token=None*, *session=None*)

Entrypoint to work with Azion API.

To start using this client, we need a valid token. For this we can use the *authorize* function:

```
from azion.api import authorize, login
auth = authorize(user, password)
azion = login(auth.token)
```

Now you can use all API resources.

authorize(*username*, *password*)

Obtain a fresh token to handle Azion's API protected calls.

Parameters

- **username** (*str*) – username
- **password** (*str*) – password

create_configuration(*name*, *origin_address*, *origin_host_header*, *cname=None*, *cname_access_only=False*, *delivery_protocol='http'*, *digital_certificate=None*, *origin_protocol_policy='preserve'*, *browser_cache_settings=False*, *browser_cache_settings_maximum_ttl=0*, *cdn_cache_settings='honor'*, *cdn_cache_settings_maximum_ttl=0*)

Create a configuration.

Parameters

- **name** (*str*) – human-readable name for the configuration.
- **origin_address** (*str*) – origin address that can be an IP or a hostname (FQDN)
- **origin_host_header** (*str*) – host header will be sent to the origin.
- **cname** (*list*) – a list os strings containing all cnames. Default empty string.
- **cname_access_only** (*bool*) – defines whether the content delivery should be done only through cnames. Default to False.
- **delivery_protocol** (*str*) – defines the HTTP protocol used to deliver content. Default to http.
- **digital_certificate** (*int*) – Digital Certificate ID. Check Digital Certificates for more info.
- **origin_protocol_policy** (*str*) – Protocol policy used to connect to the origin.
- **browser_cache_settings** (*bool*) – whether the user browser should respect the cache headers sent from the origin. Default to False.
- **browser_cache_settings_maximum_ttl** (*int*) – used within *browser_cache_settings*, defines how many seconds browser cache object will live. Default to 0.

create_origin(*configuration_id*, *name*, *origin_type*, *method*, *host_header*, *origin_protocol_policy*, *addresses*, *connection_timeout*, *timeout_between_bytes*)
Create an origin.

delete_configuration(*configuration_id*)
Delete a configuration.

Parameters **configuration_id** (*int*) – Configuration ID.

get_configuration(*configuration_id*)
Retrieve a configuration.

Parameters **configuration_id** (*int*) – configuration id

list_configurations()
List configurations.

list_origins(*configuration_id*)
List origins of the given configuration.

Parameters **configuration_id** (*int*) – Configuration ID

login(*token*)
Log the user into Azion's API.

Parameters **token** (*str*) – Authorization token. It can be obtained from `token_auth()`

partial_update_configuration(*configuration_id*, *name=None*, *cname=None*, *cname_access_only=None*, *delivery_protocol=None*, *digital_certificate=None*, *rawlogs=None*, *active=None*)
Partially updates a configuration.

One or more fields can be updated, without changing the current values of the other fields of this configuration.

Parameters

- **name** (*str*) – human-readable name for the configuration.

- **cname** (*list*) – a list os strings containing all cnames. Default empty string.
- **cname_access_only** (*bool*) – defines whether the content delivery should be done only through cnames. Default to False.
- **delivery_protocol** (*str*) – defines the HTTP protocol used to deliver content. Default to http.
- **digital_certificate** (*int*) – Digital Certificate ID. Check Digital Certificates for more info.
- **rawlogs** (*boolean*) – Whether this configuration will store logs in the Cloud Storage.
- **active** (*boolean*) – Whether this configuration is active.

purge_cache_key(*urls*, *method='delete'*)

Purge content of the given URLs inside the *urls* list. With this purge endpoint you can pass *cache keys*.

Parameters

- **urls** (*list*) – List of URLs to be purged.
- **method** (*str*) – How the content will be purged. Default to 'delete'.

purge_url(*urls*, *method='delete'*)

Purge content of the given URLs inside the *urls* list.

Parameters

- **urls** (*list*) – List of URLs to be purged.
- **method** (*str*) – How the content will be purged. Default to 'delete'.

purge_wildcard(*url*, *method='delete'*)

Purge content of the given URL. With this purge endpoint you can use a wildcard (*) to remove all objects matching the URL.

Parameters

- **url** (*str*) – Wildcard URL to be purged.
- **method** (*str*) – How the content will be purged. Default to 'delete'.

replace_configuration(*configuration_id*, *name=None*, *cname=None*, *cname_access_only=None*, *delivery_protocol=None*, *digital_certificate=None*, *rawlogs=None*, *active=None*)

Replace a configuration.

One or more fields can be updated. Fields that were not specificed in the request will be replaced for default values. Consider using *partial_update_configuration()*

Parameters

- **configuration_id** (*int*) – Configuration ID
- **name** (*str*) – human-readable name for the configuration.
- **cname** (*list*) – a list os strings containing all cnames. Default empty string.
- **cname_access_only** (*bool*) – defines whether the content delivery should be done only through cnames. Default to False.
- **delivery_protocol** (*str*) – defines the HTTP protocol used to deliver content. Default to http.
- **digital_certificate** (*int*) – Digital Certificate ID. Check Digital Certificates for more info.

- **rawlogs** (*boolean*) – Whether this configuration will store logs in the Cloud Storage.
- **active** (*boolean*) – Whether this configuration is active.

4.1.2 Client errors

Documentation of client errors generated from Azion's ReST API.

Exceptions used to describe errors when handling Azion API requests/responses.

Documentation of possible errors to be returned from the API can be found here: https://www.azion.com.br/developers/api-v1/#status-codes

exception azion.exceptions.**AzionError**(*response*)
This class represents all errors related to response.

Whenever Azion's API returns a error code, we catch them and provide a better and uniform way to handle/inspect them.

response
HTTP response that originated the error.

status_code
HTTP status code raised by the error.

errors
List of errors generated from the response.

exception azion.exceptions.**AzionException**
Base class exception. All exceptions related to requests/responses errors are inherited from it.

exception azion.exceptions.**BadRequest**(*response*)
Indicate that the server could not understand the request due to invalid syntax.

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400

exception azion.exceptions.**Conflict**(*response*)
Indicate a request conflict with current state of the server.

exception azion.exceptions.**Forbidden**(*response*)
Indicate that the server understood the request but refuses to authorize it.

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/403

exception azion.exceptions.**MethodNotAllowed**(*response*)
Indicate that the request method is known by the server but has been disabled and cannot be used.

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/405

exception azion.exceptions.**NotAcceptable**(*response*)
Indicate that a response matching the list of acceptable values defined in Accept-Charset and Accept-Language cannot be served.

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/406

exception azion.exceptions.**NotFound**(*response*)
Indicate that the server can't find the requested resource. Links which lead to a 404 page are often called broken or dead links.

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404

exception azion.exceptions.**TooManyRequests**(*response*)
Indicate he user has sent too many requests in a given amount of time ("rate limiting").

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/429

exception azion.exceptions.**Unauthorized**(*response*)
Indicate that the request has not been applied because it lacks valid authentication credentials for the target resource.

More info here: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401

azion.exceptions.**handle_error**(*response*)
Handle the request that failed to retrieve an appropriate response.

Parameters **response** (*object*) – requests Response object.

4.1.3 Configurations

Documentation of Configurations API.

class azion.models.**Configuration**(*data*)
Model representing the configuration retrieved from the API.

id
Configuration's unique ID.

name
Configuration name - a human representation to identify the configuration.

domain_name
Domain name is an unique represenation of the configuration in the entire CDN.

active
Wheter the configuration is currently deployed or not.

delivery_protocol
Delivery Protocol is the protocol used to deliver the content through the CDN.

digital_certificate
Digital's Certificate ID used to deliver the content using a SSL certificate.

rawlogs
Whether RawLogs is enabled for this configuration.

cnames
A list of domains used to represent your configuration, other than the domain_name.

CHAPTER 5

Contributing

5.1 Testing the client

Every feature of a codebase must be tested if we want to be more confident about it. Tests help to document what the code does and how it works. Need a new feature or need to change the behavior of a current function? Tests are here to cover you.

5.1.1 Integration tests

The purpose of an Integration test is to ensure that our code units, depending on other units, work as expected. In our case, we need to test how our code behave when interacting to the real API. How does it handle real JSON responses?

To acomplish this task, we use Betamax library. It records real requests made to the API, saves it as JSON and replays it next time, without hitting the production site again.

Here is an example on how to test the creation of a new configuration:

```
class TestConfiguration(object):

    def test_create_configuration(self):
        client = Azion(token)
        recorder = betamax.Betamax(client.session)

        with recorder.use_cassette('Configuration_create'):
            configuration = client.create_configuration(
                'Dummy configuration', 'www.example.com', 'ww2.example.com',
                cname=['www.example-cname.com'], delivery_protocol='http')
        assert isinstance(configuration, Configuration)
```

To create new tests, you need to export an environment variable named *AZ_TOKEN*. If you are using *Bash shell* you can export the variable and then run the tests:

```
AZ_TOKEN='mytoken'
make test
```

Using *Fish shell* like me? No problems:

```
set -x AZ_TOKEN 'mytoken'
make test
```

It is necessary because your new tests will try to hit the production API to fetch the response. After executing the test, a new *cassette* will be generated. Your patch must contain these files.

Cassettes

Cassettes are the files used to store/load requests and responses. A good convention is to name them with the resource capitalized and the action of the API function in lowercase, for example: `Configuration_create`

5.1.2 Unit tests

Fast feedback can be given by unit tests. These are tests used to cover units of code in isolation - they should not depend on other components. And to achieve this goal, we must not rely on third party results like JSON responses obtained over a unreliable network.

Mocking the response is the right way to go here, ensuring that we called our function with the right parameters.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index