Common Ground API Client

Release 0.15.0

Maykin Media, VNG-Realisatie

Mar 16, 2021

CONTENTS:

The Common Ground API Client is a generic client for Common Ground API's built with OpenAPI 3.0 specifications.

CHAPTER

ONE

FEATURES

- Driven by OAS 3.0 specification
- (Pluggable) caching of api specifications
- Create/mutate resources according to the api specifications
- Support for multiple authentication schemes
  - ZGW auth (JWT based)
  - API-key via HTTP headers
  - or none, for open APIs
- Generic approach, but very well suited for the "API's voor zaakgericht werken" standard
- Built on top of battle-proven requests library.

1.1 Getting started

1.1.1 Installation

Install with pip

```
pip install gemma-zds-client
```

1.1.2 Initialization

Initialise (static configuration)

De client moet geinitialiseerd worden met de locatie van de componenten. Dit kan eenmalig of just-in-time wanneer je de client nodig hebt:

```
from zds_client import Client

Client.load_config('/pad/naar/config.yml')
```

De makkelijkste manier is configuratie via een `yaml` bestand, in het formaat:

```
---

zrc:
  scheme: http
```

(continues on next page)

(continued from previous page)

```
  host: localhost
  port: 8000
  auth:
    client_id: my-zrc-client-id
    secret: my-zrc-client-secret

drc:
  scheme: http
  host: localhost
  port: 8001

ztc:
  scheme: http
  host: localhost
  port: 8002

orc:
  scheme: http
  host: localhost
  port: 8003
```

De key is de naam van de component.

Je kan echter ook de configuratie zonder yaml bestand doen, en volledig gebruik maken van Python dictonaries, bijvoorbeeld:

```
from zds_client import Client

ZRC = {
    'scheme': 'http',
    'host': 'localhost',
    'port': 8000,
}

DRC = {
    'scheme': 'http',
    'host': 'localhost',
    'port': 8001,
}

Client.load_config(**{
    'zrc': ZRC,
    'drc': DRC,
    ...
})
```

Initialise (ad-hoc configuration)

Je kan ook een client instance verkrijgen op basis van een specifieke resource URL.

```
from zds_client import Client

client = Client.from_url('https://api.nl/v1/resource/123')
```

Indien autorisatie hierop nodig is, kan je deze zelf assignen:

```
from zds_client import ClientAuth

client.auth = ClientAuth(
    client_id='my-client-id',
    secret='my-client-secret',
)
```

1.1.3 Using the client methods

Per component kan je vervolgens een client resources laten opvragen of manipuleren:

```
zrc_client = Client('zrc')  # gebruik alias uit configuratie

# oplijsten
zaken = zrc_client.list('zaak')

# opvragen
zaak = zrc_client.retrieve('zaak', uuid='<uuid>')

# opvragen met URL
zaak = zrc_client.retrieve('zaak', url='<zaak_url>')

# aanmaken
zaak = zrc_client.create('zaak', {
    'bronorganisatie': '000000000',
    'zaaktype': 'http://localhost:8002/api/v1/zaaktypen/<uuid>'
})
```

Operation suffixes

De `operation_id` van de OAS-operations staan centraal - op basis hiervan wordt de URL + HTTP method opgehaald die nodig is voor de call. Je kan deze suffixes overriden in client subclasses:

```
class MyClient(Client):
    operation_suffix_mapping = {
        "list": "List",
        "retrieve": "Retrieve",
        "create": "Create",
        "update": "Update",
        "partial_update": "PartialUpdate",
        "delete": "Delete",
    }
```

1.2 Usage with NLX

When you're using NLX outways, the URLs of resources change because of this. Services expoxed via NLX inways don't understand local outway URLs, so these need to get rewritten.

In Django projects, you can use zgw-consumers, which has built-in support for NLX and the required URL rewrites. This library is a dependency of `zgw-consumers`.

1.3 Python API reference

1.3.1 Client

1.3.2 Client authentication

1.3.3 Schema fetching

1.4 Changelog

1.4.1 0.15.0 (2021-03-15)

Preparations towards a 1.0 release

Breaking changes

`zds_client.schema.Schema` was removed. The initial reason to add it was to serve as a tool for NLX url rewriting middleware, which became obsolete with zgw-consumers' built-in support. If you need to parse OpenAPI 3.0 schema's, openapi-parser looks viable.

Deprecations

- `zds_client.auth.ClientAuth.claims` - claims namespaced under `zds` in the JWT payload are deprecated and scheduled for removal in 1.0.

  These claims became obsolete after the shift to store the application authorizations in the Autorisaties API.

  1.0 will support extra claims, but they will be added to the token payload without the `zds` namespace.

- `zds_client.Client.list`: `query_params` arg is deprecated in favour of `params`. This matches the underlying `requests` interface.

- `zds_client.tests.mocks` is deprecated. The mock client shim is overly complex and requires Django. Use requests-mock or responses instead to mock the underlying `requests` calls.

- `zds_client.nlx` module is deprecated. zgw-consumers is a better solution for Django-based projects. This module was Django-only already because of the dependency `nlx-url-rewriter`.

New stuff

- Added public API documentation, hosted on readthedocs.io
- Added docs build to CI workflow

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search