Swarm Intelligence

Release 0.1

Jan 17, 2017

CONTENTS

CHAPTER ONE

OVERVIEW

The aim of this project is to develop an open-source platform that fits well to the Holacracy® constitution (http://www.holacracy.org/constitution).

The stack of choice is Flask, a flexible Python micro framework to develop robust web applications. Different libraries, mentionable Flask-RESTful and Flask-SQLAlchemy, add further functionality to the stack. Data persistence is handled by a MySQL database.

To foster a fast, flexible and collaborative development process the project is hosted and maintained on GitHub (https://github.com/dcentralize/swarm-intelligence). Travis CI, a continuous integration service, makes it easy to test and to coordinate the commits and increases the quality of the product.

CHAPTER

TWO

INSTALLATION

Getting Started using Ubuntu

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

Prerequisites

First you need to checkout the GitHub Repository using:

```
git clone https://github.com/dcentralize/swarm-intelligence.git
```

It is highly recommended to run everything in an virtualenv. The environment can be set up using:

```
mkvirtualenv --python python3.4 -a . si
```

To create a local database, install Mariadb:

```
apt-get install mariadb-server
```

In order to run or deploy the project, it is necessary to download the dependencies:

```
pip3 install -r requirements.txt
```

Installation

A step by step series of examples that tell you how to get a development env running.

Starting mariadb:

```
service mariadb start
```

Setting up the database:

```
mysql -u root -e 'CREATE DATABASE swarm_intelligence'
```

Adding the directory ‘swarm-intelligence’ to your PYTHONPATH:

```
export PYTHONPATH=$PYTHONPATH:/path/of/swarm-intelligence
```

You can now navigate to the app.py and run it using:

```
cd swarm-intelligence
python3 swarm_intelligence_app/app.py
```

You can now access the API at localhost:5000. Please not that accessing the API via 127.0.0.1:5000 will not work.

Running the tests

Normally our tests are run using Travis-CI. In order to run the tests locally, navigate to the /tests directory and run:

```
py.test
```

Coding style tests

Our coding style is conform to flake8, except for some minor exceptions which can be found in the tox.ini.

Built With

- [PyCharm](https://www.jetbrains.com/pycharm/)
- [Travis-CI](https://travis-ci.org/)
- [Mariadb](https://mariadb.org/)
- [Flask](http://flask.pocoo.org/docs/0.11/)
- [Flask-Cors](https://github.com/corydolphin/flask-cors)
- [Flask-HTTPAuth](https://flask-httpauth.readthedocs.io/en/latest/)
- [Flask-RESTful](https://flask-restful-cn.readthedocs.io/en/0.3.5/)
- [Flask-SQLAlchemy](http://flask-sqlalchemy.pocoo.org/2.1/)
- [Jinja2](http://jinja.pocoo.org/)
- [PyJWT](http://github.com/jpadilla/pyjwt)
- [PyMySQL](https://media.readthedocs.org/pdf/pymysql/latest/pymysql.pdf)
- [SQLAlchemy](http://www.sqlalchemy.org)
- [SQLAlchemy-Utils](https://github.com/kvesteri/sqlalchemy-utils)
- [Py](https://pypi.python.org/pypi)
- [Pytest](http://doc.pytest.org/en/latest/)
- [Pytest-Flask](https://pytest-flask.readthedocs.io/en/latest/)
- [requests](http://python-requests.org)
- [Tox](https://tox.readthedocs.io/en/latest/)

CHAPTER THREE

USING THE API

HTTP Methods

The API is implemented as RESTful web service and uses HTTP to access and manipulate resources. The following table shows which HTTP methods are supported:

Method	Description
GET	Used for retrieving resources.
POST	Used for creating resources.
PUT	Used for updating resources.
DELETE	Used for deleting resources.

HTTP Status Codes

There are three different HTTP status codes for successful requests and five HTTP status codes to indicate client errors. The status codes are used as follows:

On success

Code	Description
200	The request has succeeded.
201	The request has succeeded and resulted in a new resource.
204	The request has succeeded without content being returned.

On client error

Code	Description
400	The request failed due to malformed syntax.
401	The request failed due to missing or invalid token.
403	The request failed due to missing permissions.
404	The requested resource was not found.
409	The request failed due to a conflict with the resource.

Authentication

Authentication is implemented by using JSON Web Tokens (JWT). To authenticate through the Swarm Intelligence Platform API sent an Authorization header with each request like this:

```
Authorization: Bearer <JSON Web Token>
```

JSON Encoded Data

All reponses contain JSON encoded data. A single resource is represented by a JSON object; A collection of resources is represented by a JSON array.

Single resource

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'key1', 'value1',
    'key2', 'value2'
}
```

Collection of resources

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'key1': 'value1',
        'key2': 'value2'
    },
    {
        'key1': 'value1',
        'key2': 'value2'
    }
]
```

Cross Origin Resource Sharing

The API supports Cross Origin Resource Sharing (CORS) for AJAX requests from any origin. You can find further information in the CORS W3C Recommendation.

CHAPTER

FOUR

EXTENDING THE API

Project Structure

The main building blocks of the Swarm Intelligence App are resources and models. The following project structure shows the separation of different modules. Any resources are located in the *resources/* folder; any models in the *models/* folder. Helpers used accross the application are located in the *common/* folder. The app is configured in *config.py* and initialized in *app.py*, which is the main entry point of the application.

```
swarm_intelligence_app/         # application root directory
    common/                     # any helpers and utils
        __init.py__
        authentication.py
    docs/                       # any documentation source files
    models/                     # any models
        __init__.py
        accountability.py
        circle.py
        domain.py
        invitation.py
        organization.py
        partner.py
        policy.py
        role.py
        role_member.py
        user.py
    resources/                  # any resources
        __init__.py
        accountability.py
        circle.py
        domain.py
        invitation.py
        organization.py
        partner.py
        policy.py
        role.py
        user.py
    tests/                      # any tests
    __init__.py
    app.py                      # application entry point
    config.py                   # application configuration
```

Adding a Resource

Resources are implemented with Flask-RESTful, an extension for Flask that adds support for building RESTful APIs. A basic CRUD resource can be defined in *resources/myresource.py* and looks like this:

```
from flask_restful import Resource

class MyResource(Resource):
    def post(self):                # create a new resource
        ...                        # insert data
        return 201, {}             # return status 201 and JSON data

    def get(self, id):             # read a resource
        ...                        # query data
        return 200, {}             # return status 200 and JSON data

    def put(self, id):             # update a resource
        ...                        # update data
        return 200, {}             # return status 200 and JSON data

    def delete(self, id):          # delete a resource
        ...                        # delete data
        return 204, None           # return status 204
```

In *app.py* import your resource class

```
from swarm_intelligence_app.resources.myresource import MyResource
```

and add it to the API object

```
def create_app():
    ...
    api.add_resource(MyResource, '/myresource')
    ...
```

Adding a Model

The Swarm Intelligence App uses Flask-SQLAlchemy, an extension that provides support for SQLAlchemy. SQLAlchemy is an SQL toolkit and Object Relational Mapper for Python. A simple model can be defined in *models/mymodel.py* and looks like this:

```
from swarm_intelligence_app.models import db

class MyModel(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    firstname = db.Column(db.String(100), nullable=False)
    lastname = db.Column(db.String(100), nullable=False)

    def __init__(self, firstname):
        self.firstname = firstname
        self.lastname = lastname

    def __repr__(self):
        return '<MyModel %r>' % self.id
    @property
    def serialize(self):
        return {
            'firstname': self.firstname,
            'lastname': self.lastname
        }
```

Import the SQLAlchemy object and your model class and use your model as follows:

```
from swarm_intelligence_app.models import db
from swarm_intelligence_app.models.mymodel import MyModel

try:
    # insert data
    mymodel = MyModel('John', 'Doe')
    db.session.add(mymodel)
    db.session.flush()

    # query data
    mymodel = MyModel.query.get(mymodel.id)

    # update data
    mymodel.lastname = 'Smith'

    # delete data
    db.session.delete(mymodel)

    # persist data
    db.session.commit()
except:
    db.session.rollback()
```

CHAPTER

FIVE

API REFERENCE

Accountability

Represents an accountability.

Resource	Operation	Description
Accountability	*PUT /accountabilities/(accountability_id)*	Update an accountability.
	DELETE /accountabilities/(accountability_id)	Delete an accountability.
	GET /accountabilities/(accountability_id)	Retrieve an accountability.

PUT /accountabilities/(*accountability_id*)

Update an accountability.

Example request:

```
PUT /accountabilities/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'title': 'Accountability's new title'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'title': 'Accountability's new title',
    'role_id': 1
}
```

Parameters

- **accountability_id** (*int*) – the accountability to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the accountability's title

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the accountability's unique id
- **title** (*string*) – the accountability's title
- **role_id** (*int*) – the role the accountability is related to

Status Codes

- 200 OK – Accountability is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Accountability is not found

DELETE /accountabilities/ (*accountability_id*)

Delete an accountability.

Example request:

```
DELETE /accountabilities/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **accountability_id** (*int*) – the accountability to delete

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Accountability is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Accountability is not found

GET /accountabilities/ (*accountability_id*)

Retrieve an accountability.

Example request:

```
GET /accountabilities/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'title': 'Accountability's title',
    'role_id': 1
}
```

Parameters

- **accountability_id** (*int*) – the accountability to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the accountability's unique id
- **title** (*string*) – the accountability's title
- **role_id** (*int*) – the role the accountability is related to

Status Codes

- 200 OK – Accountability is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Accountability is not found

Circle

Represents a circle. A circle is a *Role* that is broken down into sub roles. Every circle has core roles (e.g. facilitator, secretary, lead link, rep link, cross link) as well as custom roles. A *Partner* can be assigned to a circle as a core member.

Resource	Operation	Description
Circle	*PUT /circles/(circle_id)*	Update a circle.
	GET /circles/(circle_id)	Retrieve a circle.

PUT /circles/(*circle_id*)

Update a circle.

In order to update a circle, the authenticated user must be a partner of the organization that the circle is associated with.

Example request:

```
PUT /circles/6 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'name': 'My Circle's new name',
    'purpose': 'My Circle's new purpose',
    'strategy': 'My Circle's new strategy'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 6,
    'type': 'circle',
    'name': 'My Circle's new name',
    'purpose': 'My Circle's new purpose',
    'strategy': 'My Circle's new strategy',
    'parent_role_id': 1,
    'organization_id': 1
}
```

Parameters

- **circle_id** (*int*) – the circle to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the circle's name
- **purpose** (*string*) – the circle's purpose
- **strategy** (*string*) – the circle's strategy

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the circle's unique id

- **type** (*string*) – the circle's type
- **name** (*string*) – the circle's name
- **purpose** (*string*) – the circle's purpose
- **strategy** (*string*) – the circle's strategy
- **parent_role_id** (*int*) – the parent role the circle is related to
- **organization_id** (*int*) – the organization the circle is related to

Status Codes

- 200 OK – Circle is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Circle is not found

GET /circles/(*circle_id*)

Retrieve a circle.

In order to retrieve a circle, the authenticated user must be a partner of the organization that the circle is associated with.

Example request:

```
GET /circles/6 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 6,
    'type': 'circle',
    'name': 'Circle's name',
    'purpose': 'Circle's purpose',
    'strategy': 'Circle's strategy',
    'parent_role_id': 1,
    'organization_id': 1
}
```

Parameters

- **circle_id** (*int*) – the circle to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **`id`** (*`int`*) – the circle's unique id
- **`type`** (*`string`*) – the circle's type
- **`name`** (*`string`*) – the circle's name
- **`purpose`** (*`string`*) – the circle's purpose
- **`strategy`** (*`string`*) – the circle's optional strategy
- **`parent_role_id`** (*`int`*) – the parent role the circle is related to
- **`organization_id`** (*`int`*) – the organization the circle is related to

Status Codes

- 200 OK – Circle is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Circle is not found

Members

Represents the members of a circle. See *Partner* for a description of a single member.

Resource	Operation	Description
Circle Members	*GET /circles/(circle_id)/members*	List members of a circle.
	PUT /circles/(circle_id)/members/(partner_id)	Assign a partner to a circle.
	DELETE /circles/(circle_id)/members/(partner_id)	Unassign a partner from a circle.

`GET /circles/`(*circle_id*)**`/members`**

List members of a circle.

In order to list the members of a circle, the authenticated user must be a partner of the organization that the circle is associated with.

Example request:

```
GET /circles/1/members HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'type': 'admin',
        'firstname': 'John',
        'lastname': 'Doe',
        'email': 'john@example.org',
        'is_active': True,
        'user_id': 1,
        'organization_id': 1,
        'invitation_id': null
    }
]
```

Parameters

- **circle_id** (*int*) – the circle the members are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the partner's unique id
- **type** (*string*) – the partner's type
- **firstname** (*string*) – the partner's firstname
- **lastname** (*string*) – the partner's lastname
- **email** (*string*) – the partner's email address
- **is_active** (*boolean*) – the partner's status
- **user_id** (*int*) – the user account the partner is related to
- **organization_id** (*int*) – the organization the partner is related to
- **invitation_id** (*int*) – the invitation the partner is related to

Status Codes

- 200 OK – Members are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Circle is not found

PUT /circles/(*circle_id*)**/members/**
partner_id Assign a partner to a circle.

In order to assign a partner to a circle, the authenticated user must be an admin of the organization that the circle is associated with.

Example request:

```
PUT /circles/1/members/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **`circle_id`** (*int*) – the circle the partner is assigned to
- **`partner_id`** (*int*) – the partner who is assigned to the circle

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Partner is assigned to circle
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Circle is not found
- 404 Not Found – Partner is not found
- 409 Conflict – Circle is not associated with partner's organization

`DELETE /circles/`(*circle_id*)**`/members/`**
partner_id Unassign a partner from a circle.

In order to unassign a partner from a circle, the authenticated user must be an admin of the organization that the circle is associated with.

Example request:

```
DELETE /circles/1/members/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **`circle_id`** (*int*) – the circle the partner is unassigned from
- **`partner_id`** (*int*) – the partner who is unassigned from the circle

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Partner is unassigned from circle
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized

- 404 Not Found – Circle is not found
- 404 Not Found – Partner is not found

Roles

Represents the roles of a circle. See *Role* for a description of a single role.

Resource	Operation	Description
Circle Roles	*POST /circles/(circle_id)/roles*	Add a role to a circle.
	GET /circles/(circle_id)/roles	List roles of a circle.

POST /circles/(*circle_id*)**/roles**

Add a role to a circle.

Example request:

```
POST /circles/1/roles HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'name': 'Role's name',
    'purpose': 'Role's purpose'
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    'id': 5,
    'type': 'custom',
    'name': 'My Role's name',
    'purpose': 'My Role's purpose',
    'parent_role_id': 1,
    'organization_id': 1
}
```

Parameters

- **circle_id** (*int*) – the circle the role is added to

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the role's name
- **purpose** (*string*) – the role's purpose

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the role's unique id
- **type** (*string*) – the role's type
- **name** (*string*) – the role's name
- **purpose** (*string*) – the role's purpose
- **parent_role_id** (*int*) – the parent role the role is related to
- **organization_id** (*int*) – the organization the role is related to

Status Codes

- 201 Created – Role is added
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Circle is not found

GET /circles/(*circle_id*)**/roles**

List roles of a circle.

Example request:

```
GET /circles/1/roles HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 2,
        'type': 'lead_link',
        'name': 'Lead Link's name',
        'purpose': 'Lead Link's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 3,
        'type': 'secretary',
        'name': 'Secretary's name',
        'purpose': 'Secretary's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 4,
        'type': 'facilitator',
        'name': 'Facilitator's name',
        'purpose': 'Facilitator's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 5,
        'type': 'custom',
        'name': 'My Role's name',
        'purpose': 'My Role's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 6,
        'type': 'circle',
        'name': 'My Circle's name',
        'purpose': 'My Circle's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    }
]
```

Parameters

- **`circle_id`** (*`int`*) – the circle the roles are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **`id`** (*`int`*) – the role's unique id
- **`type`** (*`string`*) – the role's type
- **`name`** (*`string`*) – the role's name
- **`purpose`** (*`string`*) – the role's purpose
- **`parent_role_id`** (*`int`*) – the parent role the role is related to
- **`organization_id`** (*`int`*) – the organization the role is related to

Status Codes

- 200 OK – Roles are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Circle is not found

Domain

Represents a domain.

Resource	Operation	Description
Domain	*DELETE /domains/(domain_id)*	Delete a domain.
	GET /domains/(domain_id)	Retrieve a domain.
Role	*PUT /domains/(domain_id)*	Update a domain.

PUT /domains/(*domain_id*)

Update a domain.

Example request:

```
PUT /domains/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'title': 'Domain's new title'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'title': 'Domain's new title',
    'role_id': 1
}
```

Parameters

- **domain_id** (*int*) – the domain to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the domain's title

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the domain's unique id
- **title** (*string*) – the domain's title

- **role_id** (*int*) – the domain the role is related to

Status Codes

- 200 OK – Domain is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Domain is not found

DELETE /domains/(*domain_id*)

Delete a domain.

Example request:

```
DELETE /domain/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **domain_id** (*int*) – the domain to delete

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Domain is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Domain is not found

GET /domains/(*domain_id*)

Retrieve a domain.

Example request:

```
GET /domains/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'title': 'Domain's title',
    'role_id': 1
}
```

Parameters

- **domain_id** (*int*) – the domain to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the domain's unique id
- **title** (*string*) – the domain's title
- **role_id** (*int*) – the role the domain is related to

Status Codes

- 200 OK – Domain is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Domain is not found

Policies

Represents the policies of a domain. See *Policy* for a description of a single policy.

Resource	Operation	Description
Domain Policies	*POST /domains/(domain_id)/policies*	Add a policy to a domain.
	GET /domains/(domain_id)/policies	List policies of a domain.

POST /domains/(*domain_id*)**/policies**

Add a policy to a domain.

Example request:

```
POST /domains/1/policies HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'title': 'Policy's title'
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    'id': 1,
    'title': 'Policy's title',
    'domain_id': 1
}
```

Parameters

- **organization_id** (*int*) – the domain the policy is added to

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **title** (*string*) – the policy's title

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the policy's unique id
- **title** (*string*) – the policy's title
- **domain_id** (*int*) – the domain the policy is related to

Status Codes

- 201 Created – Policy is added to domain
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Domain is not found

GET /domains/(*domain_id*)**/policies**

List policies of a domain.

Example request:

```
GET /domains/1/policies HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'title': 'Policy's title',
        'domain_id': 1
    }
]
```

Parameters

- **`domain_id`** (*`int`*) – the domain the policies are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **`id`** (*`int`*) – the policy's unique id
- **`title`** (*`string`*) – the policy's title
- **`domain_id`** (*`int`*) – the domain the policy is related to

Status Codes

- 200 OK – Policies are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Domain is not found

Invitation

Represents an invitation.

Resource	Operation	Description
Invitation	*GET /invitations/(invitation_id)*	Retrieve an invitation.
	GET /invitations/(code)/accept	Accept an invitation.
	PUT /invitations/(invitation_id)/cancel	Cancel an invitation.

GET /invitations/ (*invitation_id*)

Retrieve an invitation.

In order to retrieve an invitation, the authenticated user must be a partner of the organization that the invitation is associated with.

Example request:

```
GET /invitations/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'code': '12345678-1234-1234-1234-123456789012',
    'email': 'john@example.org',
    'status': 'pending',
    'organization_id': 1
}
```

Parameters

- **`invitation_id`** (*`int`*) – the invitation to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **`id`** (*`int`*) – the invitation's unique id
- **`code`** (*`string`*) – the invitation's unique code
- **`email`** (*`string`*) – the email address the invitation is sent to
- **`status`** (*`string`*) – the invitation's status
- **`organization_id`** (*`int`*) – the organization the invitation is related to

Status Codes

- 200 OK – Invitation is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Invitation is not found

`GET /invitations/`(*code*)**`/accept`**

Accept an invitation.

If an invitation's state is 'pending', this endpoint will set the invitation's state to 'accepted' and the authenticated user will be added as a partner to the associated organization. If an invitation's state is 'accepted' or 'cancelled', the invitation cannot be accepted again or accepted at all. In order to accept an invitation, the user must be an authenticated user.

Example request:

```
GET /invitations/12345678-1234-1234-1234-123456789012/accept
HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'code': '12345678-1234-1234-1234-123456789012',
    'email': 'john@example.org',
    'status': 'accepted',
    'organization_id': 1
}
```

Parameters

- **`invitation_id`** (*`int`*) – the invitation to accept

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **`id`** (*`int`*) – the invitation's unique id
- **`code`** (*`string`*) – the invitation's unique code
- **`email`** (*`string`*) – the email address the invitation is sent to
- **`status`** (*`string`*) – the invitation's status
- **`organization_id`** (*`int`*) – the organization the invitation is related to

Status Codes

- 200 OK – Invitation is accepted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Invitation is not found
- 409 Conflict – Invitation status is cancelled

`PUT /invitations/`(*invitation_id*)**`/cancel`**

Cancel an invitation.

If an invitation's state is 'pending', this endpoint will set the invitation's state to 'cancelled'. If an invitation's state is 'accepted' or 'cancelled', the invitation cannot be cancelled at all or cancelled again. In order to cancel an invitation, the authenticated user must be an admin of the organization that the invitation is associated with.

Example request:

```
GET /invitations/1/cancel HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'code': '12345678-1234-1234-1234-123456789012',
    'email': 'john@example.org',
    'status': 'cancelled',
    'organization_id': 1
}
```

Parameters

- **invitation_id** (*int*) – the invitation to cancel

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the invitation's unique id
- **code** (*string*) – the invitation's unique code
- **email** (*string*) – the email address the invitation is sent to
- **status** (*string*) – the invitation's status
- **organization_id** (*int*) – the organization the invitation is related to

Status Codes

- 200 OK – Invitation is cancelled
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Invitation is not found
- 409 Conflict – Invitation status is accepted

Organization

Represents an organization.

Resource	Operation	Description
Organization	*PUT /organizations/(organization_id)*	Update an organization.
	DELETE /organizations/(organization_id)	Delete an organization.
	GET /organizations/(organization_id)	Retrieve an Organization.

PUT /organizations/(*organization_id*)

Update an organization.

In order to update an organization, the authenticated user must be an admin of the organization.

Example request:

```
PUT /organizations/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'name': 'My Organization'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'name': 'My Organization'
}
```

Parameters

- **organization_id** (*int*) – the organization to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the organization's name

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the organization's unique id
- **name** (*string*) – the organization's name

Status Codes

- 200 OK – Organization is updated
- 400 Bad Request – Parameters are missing

- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

DELETE /organizations/(*organization_id*)

Delete an organization.

In order to delete an organization, the authenticated user must be an admin of the organization.

Example request:

```
DELETE /organizations/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **organization_id** (*int*) – the organization to delete

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Organization is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

GET /organizations/(*organization_id*)

Retrieve an organization.

In order to retrieve an organization, the authenticated user must be a member or an admin of the organization.

Example request:

```
GET /organizations/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'name': 'My Organization'
}
```

Parameters

- **`organization_id`** (*int*) – the organization to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **`id`** (*int*) – the organization's unique id
- **`name`** (*string*) – the organization's name

Status Codes

- 200 OK – Organization is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

Anchor Circle

Represents the anchor circle of an organization. See *Circle* for a description of a circle.

Resource	Operation	Description
Organization Anchor Circle	*GET /organizations/(organization_id)/anchor_circle*	Retrieve the anchor circle.

GET /organizations/(*organization_id*)**/anchor_circle**

Retrieve the anchor circle of an organization.

This endpoint retrieves the anchor circle of an organization. Each organization has exactly one circle as its anchor circle. In order to retrieve the anchor circle of an organization, the authenticated user must be a member or an admin of the organization.

Example request:

```
GET /organizations/1/anchor_circle HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'type': 'circle',
    'name': 'My Organization',
    'pupose': 'My Organization's purpose',
    'strategy': 'My Organizations's strategy',
    'parent_circle_id': null,
    'organization_id': 1
}
```

Parameters

- **organization_id** (*int*) – the organization to retrieve the anchor circle of

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the anchor circle's unique id
- **type** (*string*) – the anchor circle's type
- **name** (*string*) – the anchor circle's name
- **purpose** (*string*) – the anchor circle's purpose
- **strategy** (*string*) – the anchor circle's strategy
- **parent_role_id** (*int*) – the role the anchor circle is a child of
- **organization_id** (*int*) – the organization the anchor circle is related to

Status Codes

- 200 OK – Anchor circle is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

Invitations

Represents the invitations to an organization. See *Invitation* for a description of a single invitation.

Resource	Operation	Description
Organization Invitations	*POST /organizations/(organization_id)/invitations*	Invite a user to an
	GET /organizations/(organization_id)/invitations	List invitations to an

POST /organizations/(*organization_id*)**/invitations**

Invite a user to an organization.

This endpoint will send an invitation to a given email address. The newly-created invitation will be in the 'pending' state until the user accepts the invitation. At this point the invitation will transition to the 'accepted' state and the user will be added as a new partner to the organization. In order to invite a user to an organization, the authenticated user must be an admin of the organization.

Example request:

```
POST /organizations/1/invitations HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'email': 'john@example.org'
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    'id': 1,
    'code': '12345678-1234-1234-1234-123456789012',
    'email': 'john@example.org',
    'status': 'pending',
    'organization_id': 1
}
```

Parameters

- **organization_id** (*int*) – the organization the invitation is created for

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **email** (*string*) – the email address the invitation is sent to

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the invitation's unique id
- **code** (*string*) – the invitation's unique code
- **email** (*string*) – the email address the invitation is sent to
- **status** (*string*) – the invitation's status
- **organization_id** (*int*) – the organization the invitation is related to

Status Codes

- 201 Created – Invitation is created
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed

- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

GET /organizations/(*organization_id*)**/invitations**

List invitations to an organization.

This endpoint lists all 'pending', 'accepted' and 'cancelled' invitations to an organization. In order to list invitations to an organization, the authenticated user must be a member or an admin of the organization.

Example request:

```
GET /organizations/1/invitations HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'code': '12345678-1234-1234-1234-123456789012',
        'email': 'john@example.org',
        'status': 'pending',
        'organization_id': 1
    }
]
```

Parameters

- **organization_id** (*int*) – the organization the invitations are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the invitation's unique id
- **code** (*string*) – the invitation's unique code
- **email** (*string*) – the email address the invitation is sent to
- **status** (*string*) – the invitation's status
- **organization_id** (*int*) – the organization the invitation is related to

Status Codes

- 200 OK – Invitations are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired

- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

Members

Represents the members of an organization. See *Partner* for a description of a single member.

Resource	Operation	Description
Organization Members	*GET /organizations/(organization_id)/members*	List members of an organization.

GET /organizations/(*organization_id*)**/members**

List partners of an organization.

This endpoint lists all members of an organization, whether their status is 'active' or not. In order to list the members of an organization, the authenticated user must be a members of the organization.

Example request:

```
GET /organizations/1/members HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'type': 'member',
        'firstname': 'John',
        'lastname': 'Doe',
        'email': 'john@example.org',
        'is_active': True,
        'user_id': 1,
        'organization_id': 1,
        'invitation_id': null
    }
]
```

Parameters

- **organization_id** (*int*) – the organization the members are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the member's unique id

- **type** (*string*) – the member's type
- **firstname** (*string*) – the member's firstname
- **lastname** (*string*) – the member's lastname
- **email** (*string*) – the member's email address
- **is_active** (*boolean*) – the member's status
- **user_id** (*int*) – the user account the member is related to
- **organization_id** (*int*) – the organization the member is related to
- **invitation_id** (*int*) – the invitation the member is related to

Status Codes

- 200 OK – Members are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Organization is not found

Partner

Represents the relationship of a *User* with an *Organization*.

Resource	Operation	Description
Partner	*PUT /partners/(partner_id)*	Update a partner.
	DELETE /partners/(partner_id)	Delete a partner.
	GET /partners/(partner_id)	Retrieve a partner.

PUT /partners/ (*partner_id*)

Update a partner.

In order to update a partner, the authenticated user must be a partner with admin access of the organization that the partner is associated with.

Example request:

```
PUT /partners/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'firstname': 'John',
    'lastname': 'Doe',
    'email': 'john@example.org'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'type': 'member',
    'firstname': 'John',
    'lastname': 'Doe',
    'email': 'john@example.org',
    'is_active': True,
    'user_id': 1,
    'organization_id': 1,
    'invitation_id': null
}
```

Parameters

- **`partner_id`** (*int*) – the partner to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **`firstname`** (*string*) – the partner's firstname
- **`lastname`** (*string*) – the partner's lastname
- **`email`** (*string*) – the partner's email address

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **`id`** (*int*) – the partner's unique id
- **`type`** (*string*) – the partner's type
- **`firstname`** (*string*) – the partner's firstname
- **`lastname`** (*string*) – the partner's lastname
- **`email`** (*string*) – the partner's email address
- **`is_active`** (*boolean*) – the partner's status
- **`user_id`** (*int*) – the user account the partner is related to
- **`organization_id`** (*int*) – the organization the partner is related to
- **`invitation_id`** (*int*) – the invitation the partner is related to

Status Codes

- 200 OK – Partner is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired

- 401 Unauthorized – User is not authorized
- 404 Not Found – Partner is not found

DELETE /partners/(*partner_id*)

Delete a partner.

In order to delete a partner, the authenticated user must be a partner with admin access of the organization that the partner is associated with.

Example request:

```
DELETE /partners/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **partner_id** (*int*) – the partner to delete

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Partner is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Partner is not found
- 409 Conflict – Partner is the only admin of an organization

GET /partners/(*partner_id*)

Retrieve a partner.

In order to retrieve a partner, the authenticated user must be a partner of the organization that the partner is associated with.

Example request:

```
GET /partners/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'type': 'member',
    'firstname': 'John',
    'lastname': 'Doe',
    'email': 'john@example.org',
    'is_active': True,
    'user_id': 1,
    'organization_id': 1,
    'invitation_id': null
}
```

Parameters

- **partner_id** (*int*) – the partner to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the partner's unique id
- **type** (*string*) – the partner's type
- **firstname** (*string*) – the partner's firstname
- **lastname** (*string*) – the partner's lastname
- **email** (*string*) – the partner's email address
- **is_active** (*boolean*) – the partner's status
- **user_id** (*int*) – the user account the partner is related to
- **organization_id** (*int*) – the organization the partner is related to
- **invitation_id** (*int*) – the invitation the partner is related to

Status Codes

- 200 OK – Partner is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Partner is not found

Memberships

Represents the relationships of a partner with roles and circles. See *Role* or *Circle* for a description of a single role or a single circle.

Resource	Operation	Description
Partner Memberships	*GET /partners/(partner_id)/memberships*	List memberships of a partner.

GET /partners/(*partner_id*)**/memberships**

List memberships of a partner.

In order to list the memberships of a partner, the authenticated user must be a partner of the organization that the partner is associated with.

Example request:

```
GET /partners/1/memberships HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'type': 'circle',
        'name': 'My Organization',
        'pupose': 'My Organization's purpose',
        'parent_role_id': null,
        'organization_id': 1
    },
    {
        'id': 2,
        'type': 'lead_link',
        'name': 'Lead Link's name',
        'purpose': 'Lead Link's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 3,
        'type': 'secretary',
        'name': 'Secretary's name',
        'purpose': 'Secretary's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 4,
        'type': 'facilitator',
        'name': 'Facilitator's name',
        'purpose': 'Facilitator's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 5,
        'type': 'custom',
        'name': 'My Role's name',
        'purpose': 'My Role's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    },
    {
        'id': 6,
        'type': 'circle',
        'name': 'My Circle's name',
        'purpose': 'My Circle's purpose',
        'parent_role_id': 1,
        'organization_id': 1
    }
]
```

Parameters

- **partner_id** (*int*) – the partner the memberships are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the role's unique id
- **type** (*string*) – the role's type
- **name** (*string*) – the role's name
- **purpose** (*string*) – the role's purpose
- **parent_role_id** (*int*) – the parent role the role is related to
- **organization_id** (*int*) – the organization the role is related to

Status Codes

- 200 OK – Memberships are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Partner is not found

Policy

Represents a policy.

Resource	Operation	Description
Policy	*PUT /policies/(policy_id)*	Update a policy.
	DELETE /policies/(policy_id)	Delete a policy.
	GET /policies/(policy_id)	Retrieve a policy.

PUT /policies/ (*policy_id*)

Update a policy.

Example request:

```
PUT /policies/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'title': 'Policy's new title'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'title': 'Policy's new title',
    'domain_id': 1
}
```

Parameters

- **policy_id** (*int*) – the policy to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the policy's title

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the policy's unique id
- **title** (*string*) – the policy's title
- **domain_id** (*int*) – the domain the policy is related to

Status Codes

- 200 OK – Policy is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Policy is not found

DELETE /policies/(*policy_id*)

Delete a policy.

Example request:

```
DELETE /policies/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **policy_id** (*int*) – the policy to delete

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Policy is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Policy is not found

GET /policies/(*policy_id*)

Retrieve a policy.

Example request:

```
GET /policies/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'title': 'Policy's title',
    'domain_id': 1
}
```

Parameters

- **policy_id** (*int*) – the policy to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the policy's unique id
- **title** (*string*) – the policy's title
- **domain_id** (*int*) – the domain the policy is related to

Status Codes

- 200 OK – Policy is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Policy is not found

Role

Represents a role. A role defines domains to control and accountabilities to perform. A *Partner* can be assigned to core roles (e.g. facilitator, secretary, lead link, rep link, cross link) as well as custom roles.

Resource	Operation	Description
Role	*PUT /roles/(role_id)*	Update a role.
	DELETE /roles/(role_id)	Delete a role.
	GET /roles/(role_id)	Retrieve a role.

PUT /roles/(*role_id*)

Update a role.

Example request:

```
PUT /roles/5 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'name': 'My Role's new name',
    'purpose': 'My Role's new purpose'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 5,
    'type': 'custom',
    'name': 'My Role's new name',
    'purpose': 'My Role's new purpose',
    'parent_role_id': 1,
    'organization_id': 1
}
```

Parameters

- **role_id** (*int*) – the role to update

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the role's name
- **purpose** (*string*) – the role's purpose

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the role's unique id
- **type** (*string*) – the role's type
- **name** (*string*) – the role's name
- **purpose** (*string*) – the role's purpose
- **parent_role_id** (*int*) – the parent role the role is related to
- **organization_id** (*int*) – the organization the role is related to

Status Codes

- 200 OK – Role is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

DELETE /roles/(*role_id*)

Delete a role.

Example request:

```
DELETE /roles/5 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **role_id** (*int*) – the role to delete

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Role is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found
- 409 Conflict – Role type is other than custom
- 409 Conflict – Role is an anchor circle of an organization

GET /roles/(*role_id*)

Retrieve a role.

Example request:

```
GET /roles/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 5,
    'type': 'custom',
    'name': 'My Role's name',
    'purpose': 'My Role's purpose',
    'parent_role_id': 1,
    'organization_id': 1
}
```

Parameters

- **role_id** (*int*) – the role to retrieve

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the role's unique id
- **type** (*string*) – the role's type

- **name** (*string*) – the role's name
- **purpose** (*string*) – the role's purpose
- **parent_role_id** (*int*) – the parent role the role is related to
- **organization_id** (*int*) – the organization the role is related to

Status Codes

- 200 OK – Role is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

Accountabilities

Represents the accountabilities of a role. See *Accountability* for a description of a single accountability.

Resource	Operation	Description
Role Accountabilities	*POST /roles/(role_id)/accountabilities*	Add an accountability to a role.
	GET /roles/(role_id)/accountabilities	List accountabilities of a role.

POST /roles/(*role_id*)**/accountabilities**

Add an accountability to a role.

Example request:

```
POST /roles/1/accountabilities HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'title': 'Accountability's title'
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    'id': 1,
    'title': 'Accountability's title',
    'role_id': 1
}
```

Parameters

- **role_id** (*int*) – the role the accountability is added to

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **title** (*string*) – the accountability's title

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the accountability's unique id
- **title** (*string*) – the accountability's title
- **role_id** (*int*) – the role the accountability is related to

Status Codes

- 201 Created – Accountability is added
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

GET /roles/(*role_id*)**/accountabilities**

List accountabilities of a role.

Example request:

```
GET /roles/1/accountabilities HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'title': 'Accountability's title',
        'role_id': 1
    }
]
```

Parameters

- **role_id** (*int*) – the role the accountabilities are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the accountability's unique id
- **title** (*string*) – the accountability's title
- **role_id** (*int*) – the role the accountability is related to

Status Codes

- 200 OK – Accountabilities are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

Circle

Converts a role to a *Circle* and vice versa.

Resource	Operation	Description
Role Circle	*PUT /roles/(role_id)/circle*	Add circle properties to a role.
	DELETE /roles/(role_id)/circle	Remove circle properties from a role.

PUT /roles/(*role_id*)**/circle**

Add circle properties to a role.

Example request:

```
PUT /roles/5/circle HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **role_id** (*int*) – the role to add circle properties to

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Circle properties are added to role
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired

- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found
- 409 Conflict – Role type is other than custom

DELETE /roles/(*role_id*)**/circle**

Remove circle properties from a role.

Example request:

```
DELETE /roles/5/circle HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **role_id** (*int*) – the role to remove circle properties from

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Circle properties are removed from role
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found
- 409 Conflict – Role is other than circle
- 409 Conflict – Role is an anchor circle of an organization

Domains

Represents the domains of a role. See *Domain* for a description of a single domain.

Resource	Operation	Description
Role Domains	*POST /roles/(role_id)/domains*	Add a domain to a role.
	GET /roles/(role_id)/domains	List domains of a role.

POST /roles/(*role_id*)**/domains**

Add a domain to a role.

Example request:

```
POST /roles/1/domains HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'title': 'Domain's title'
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    'id': 1,
    'title': 'Domain's title',
    'role_id': 1
}
```

Parameters

- **role_id** (*int*) – the role the domain is added to

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **title** (*string*) – the domain's title

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the domain's unique id
- **title** (*string*) – the domain's title
- **role_id** (*int*) – the role the domain is related to

Status Codes

- 201 Created – Domain is added
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

GET /roles/(*role_id*)**/domains**

List domains of a role.

Example request:

```
GET /roles/1/domains HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'title': 'Domain's title',
        'role_id': 1
    }
]
```

Parameters

- **role_id** (*int*) – the role the domains are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the domain's unique id
- **title** (*string*) – the domain's title
- **role_id** (*int*) – the role the domain is related to

Status Codes

- 200 OK – Domains are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

Members

Represents the members of a role. See *Partner* for a description of a single member.

Resource	Operation	Description
Role Members	*GET /roles/(role_id)/members*	List members of a role.
	PUT /roles/(role_id)/members/(partner_id)	Assign a partner to a role.
	DELETE /roles/(role_id)/members/(partner_id)	Unassign a partner from a role.

GET /roles/(*role_id*)**/members**

List members of a role.

Example request:

```
GET /roles/1/members HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'type': 'admin',
        'firstname': 'John',
        'lastname': 'Doe',
        'email': 'john@example.org',
        'is_active': True,
        'user_id': 1,
        'organization_id': 1,
        'invitation_id': null
    }
]
```

Parameters

- **role_id** (*int*) – the role the members are listed for

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the partner's unique id
- **type** (*string*) – the partner's type
- **firstname** (*string*) – the partner's firstname
- **lastname** (*string*) – the partner's lastname
- **email** (*string*) – the partner's email address
- **is_active** (*boolean*) – the partner's status
- **user_id** (*int*) – the user account the partner is related to
- **organization_id** (*int*) – the organization the partner is related to
- **invitation_id** (*int*) – the invitation the partner is related to

Status Codes

- 200 OK – Members are listed
- 400 Bad Request – Token is not well-formed

- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found

PUT /roles/(*role_id*)**/members/**
partner_id Assign a partner to a role.

Example request:

```
PUT /roles/5/members/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **role_id** (*int*) – the role the partner is assigned to
- **partner_id** (*int*) – the partner who is assigned to the role

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Partner is assigned to role
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found
- 404 Not Found – Partner is not found
- 409 Conflict – Role is not associated with partner's organization

DELETE /roles/(*role_id*)**/members/**
partner_id Unassign a partner from a role.

Example request:

```
DELETE /roles/5/members/1 HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Parameters

- **role_id** (*int*) – the role the partner is unassigned from
- **partner_id** (*int*) – the partner who is unassigned from the role

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – Partner is unassigned from role
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 404 Not Found – Role is not found
- 404 Not Found – Partner is not found

User

Represents a user. A user is related to an *Organization* through a *Partner*.

Resource	Operation	Description
User	*PUT /me*	Update the authenticated user.
	DELETE /me	Delete the authenticated user.
	GET /me	Retrieve the authenticated user.

PUT /me

Update the authenticated user.

Example request:

```
PUT /me HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'firstname': 'John',
    'lastname': 'Doe',
    'email': 'john@example.org'
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'google_id': '123456789',
    'firstname': 'John',
    'lastname': 'Doe',
    'email': 'john@example.org',
    'is_active': True
}
```

```
```

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **firstname** (*string*) – the user's firstname
- **lastname** (*string*) – the user's lastname
- **email** (*string*) – the user's email

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the user's id
- **google_id** (*string*) – the user's google id
- **firstname** (*string*) – the user's firstname
- **lastname** (*string*) – the user's lastname
- **email** (*string*) – the user's email
- **is_active** (*boolean*) – the user's status

Status Codes

- 200 OK – User is updated
- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized

DELETE /me

Delete the authenticated user.

This endpoint sets the authenticated user's account and partnerships with organizations to 'inactive'. By signin-up again with the same google account, the user's account is reactivated. To rejoin an organization, a new invitation is needed.

Example request:

```
DELETE /me HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 204 No Content
```

Request Headers

- Authorization – JSON Web Token to authenticate

Status Codes

- 204 No Content – User is deleted
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized

GET /me

Retrieve the authenticated user.

Example request:

```
GET /me HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    'id': 1,
    'google_id': '123456789',
    'firstname': 'John',
    'lastname': 'Doe',
    'email': 'john@example.org',
    'is_active': True
}
```

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the user's id
- **google_id** (*string*) – the user's google id
- **firstname** (*string*) – the user's firstname
- **lastname** (*string*) – the user's lastname
- **email** (*string*) – the user's email
- **is_active** (*boolean*) – the user's status

Status Codes

- 200 OK – User is retrieved
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized

Organizations

Represents the organizations of the authenticated user. See *Organization* for a description of a single organization.

Resource	Operation	Description
User Organizations	*POST /me/organizations*	Create an organization.
	GET /me/organizations	List the user's organizations.

POST /me/organizations

Create an organization.

This endpoint creates a new organization with an anchor circle and adds the authenticated user as an admin to the organization.

Example request:

```
POST /me/organizations HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json

{
    'name': 'My Organization'
}
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    'id': 1,
    'name': 'My Organization'
}
```

Request Headers

- Authorization – JSON Web Token to authenticate
- Content-Type – data is sent as application/json or application/x-www-form-urlencoded

Request JSON Object

- **name** (*string*) – the organization's name

Response Headers

- Content-Type – data is received as application/json

Response JSON Object

- **id** (*int*) – the organization's id
- **name** (*string*) – the organization's name

Status Codes

- 201 Created – Organization is created

- 400 Bad Request – Parameters are missing
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized
- 409 Conflict – Organization cannot be created

GET /me/organizations

List organizations for the authenticated user.

This endpoint only lists organizations that the authenticated user is allowed to operate on as a member or an admin.

Example request:

```
GET /me/organizations HTTP/1.1
Host: example.com
Authorization: Bearer <token>
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

[
    {
        'id': 1,
        'name': 'My Organization'
    }
]
```

Request Headers

- Authorization – JSON Web Token to authenticate

Response Headers

- Content-Type – data is received as application/json

Response JSON Array of Objects

- **id** (*int*) – the organization's id
- **name** (*string*) – the organization's name

Status Codes

- 200 OK – Organizations are listed
- 400 Bad Request – Token is not well-formed
- 401 Unauthorized – Token has expired
- 401 Unauthorized – User is not authorized

HTTP ROUTING TABLE