Keyclic english documentation Documentation

Release master

Oct 08, 2019

Contents

Version française

Keyclic is a rise and process feedback app. It allows its users to signal malfunctions, technical problems or every other type of problem to the right organizations.

CHAPTER 1

Content

1.1 Overview

1.1.1 General purpose

The Keyclic app is free and open to registration. Once registered, a user can create feedbacks. A feedback is always linked to a geographic point and may contain one or more pictures.

Users can join or create an organization. This a group of users from the same company, school, association, etc.

Administrators define places where their organization can act and categories (example: road network, lost animal, cleaning, etc). Thus, when a user creates a feedback, the app will display organizations able to deal with the problem and their category and let him choose which organization corresponds the most.

When a user created a feedback, he may choose to make it public or private. If public, the feedback will be visible by the entire community and they can comment it or support it.

Furthermore, the moment the feedback is send, a report is created and transmitted to the chosen organization.

A report corresponds to one and only one feedback. An admin has access to all the reports sent to its organization. For each reports, he can create has many operations as necessary and assign them to members of his organization. Another way is to delegate the report to a partner organization that will deal with it Once every operation resolved, the admin will then close the report and consider the problem resolved.

1.1.2 Useful links

- API's Swagger
- SDK client Javascript source code
- API's Technical documentation

1.1.3 Vocabulary

Feedback

Observation made by a user. It may be a technical problem, a malfunction, a nuisance, etc. Every feedback is made with coordinates from the geolocation. It may contain pictures, be commented and/or be supported.

See *Feedbacks*.

Report

Each feedback can create a report. A report contains the informations from the feedback and can only be seen and modified by the organization it was sent to. The report is the "professional" side of the observation. The organization works from this report: creation of operations, state, delegation, etc.

See *Reports*

Organization

Group of users. It may be a company, a school, an association, a group of worker on a construction site, etc.

See *Organizations*.

Places

Geographic area where an organization can take action.

See *Manage places*.

Category

Business sector of an organization.

See *Manage categories*.

Contribution

A feedback may be supported by other users of the community, to give it more value.

See *Contributions*.

Operation

Task created by an admin on a report. This task is assigned to a member of the organization.

See *Operations*.

Relationship

An admin can choose partner organizations (with mutual consent), which are other organizations he will be able to delegate reports.

See *Manage partnership*.

1.2 Technical considerations

The Keyclic API is RESTful. Every operation is sent through **HTTPS** and is secured with JSON Web Tokens. The data is sent only in **JSON** format.

The Keyclic API's domain name is :

```
api.keyclic.com
```

1.2.1 Useful links

- API's Swagger
- Technical API documentation

1.2.2 Applications and Application's key

Every client of the API must send a key identifying his application in the request header

If you are developing a client to work with an existing Keyclic application, you have to know the application's key. Else if you wish to develop a client for a new application, please contact Keyclic. They will create the new application, configure it and send you the corresponding key.

Examples of an application's key :

- com.keyclic.app
- com.keyclic.city
- com.keyclic.highway

Then every request header has to contain a X-Keyclic-App field with a value like the examples above. Refer to this paragraph *Requests* for the "implementation".

However, users accounts are shared to all Keyclic applications. Therefore, login and register endpoints do not require to provide an application key (cf : *Authentication and connection*).

1.2.3 Requests

In this documentation, each API's endpoint will be described by an HTTP verb and the path to access the resource.

Example :

```
GET /feedbacks
```

This endpoint returns every feedback. Its actual URL is

```
https://api.keyclic.com/feedbacks
```

but to avoid redundancy, in the following examples, we will show neither the protocol nor the domain name.

URL's parameters

In this documentation, URIs variables, such as a resource identifier, a page number, etc, will be between curly brackets. For example, to retrieve a single feedback :

```
GET /feedbacks/{feedback}
```

In the Keyclic API, in accordance with REST principles, the filtering parameters will always be in the query string. Example :

```
GET /feedbacks?page={page}
```

Moreover, for a better visibility, URI's parameters will be written as such and not in their encoded URL form :

```
GET /feedbacks?before=2018-04-22T01:00:00+05:00
```

Headers

Besides conventional HTTP/1.1 headers, Keyclic API accepts and in most cases requires, the header **X-Keyclic-App**, corresponding to the application used (see above : *Applications and Application's key*). For example, to get all feedbacks from the com.keyclic.app application, the request will have to contain the following header :

```
X-Keyclic-App : com.keyclic.app
```

Every endpoint requires this header, except for login and password modification. (refer : *Authentication and connection*)

Also, every request (except login, register and password modification) must contain the Authorization header (see : *Authentication and connection*).

1.2.4 Request and response format

The only type of content accepted by the Keyclic API is JSON. Requests must contain the header :

```
Content-type: application/json
```

and the body will always have to in JSON format. The responses are returned to the JSON format too.

1.2.5 Send files

Files are sent in base 64 to the API. Here is an example of adding an image to a feedback :

```
POST /feedbacks/{feedback}/images
```

```
{
    "image":"
↪iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAIAAAACDbGyAAAACXBIWXMAAAsTAAALEwEAmpwYAAAAB3RJTUUH4QIVDRUfvq7u+AA
↪"
}
```

1.2.6 Pagination

Endpoints requesting a collection of resources can be paginated with the **page** and **limit** filters. For example, to get the second page of the feedbacks with 5 feedbacks per page :

```
POST /feedbacks?page=2&limit=5
```

By default, *page* is equal to 1 and *limit* to 10. Thus, the endpoint

```
POST /feedbacks
```

returns the first 10 feedbacks.

When a collection is returned, the response will contain informations and links to browse the pages of that collection. Below an example (partial) of a list of feedbacks.

```
{
  "page": 2,
  "limit": 10,
  "pages": 8,
  "total": 72,
  "_links": {
    "self": {
      "href": "/feedbacks?page=2&limit=10"
    },
    "first": {
      "href": "/feedbacks?page=1&limit=10"
    },
    "last": {
      "href": "/feedbacks?page=8&limit=10"
    },
    "next": {
      "href": "/feedbacks?page=3&limit=10"
    },
    "previous": {
      "href": "/feedbacks?page=1&limit=10"
    }
  }
}
```

In the future, we won't precise every time that you may paginate with the *page* et *limit* filters, those are the same for every endpoint returning a collection.

1.2.7 Resource modification

In the Keyclic API, resource modification is made with the PATCH method. Unlike the PUT method, PATCH allows to modify a single or some properties of a resource without sending every property of the modified resource.

Here is an example to change the property *billingEmailAddress* of an organization :

```
PATCH /organizations/{organization}
```

```
{
                "billingEmailAddress": "test@test.com"
    }
```

1.2.8 Errors

Every error send a code 4xx representing the type of error.

When an code 400 (Bad Request) is returned, the reasons are sent.

Errors follow the format vdn.error.

The following example displays a validation error.

```
{
   "@context":"https://github.com/blongden/vnd.error",
   "@type":"ValidationError",
   "message":"Validation failed.",
   "total":1,
   "_embedded":{
      "errors":[
         {
            "@context":"https://github.com/blongden/vnd.error",
            "@type":"Error",
            "message":"Cette valeur ne doit pas \u00eatre vide.",
            "path":"reporter"
         }
      ]
   }
}
```

The field *path* indicates which property triggered the error (here: reporter), and the field *message* explains the error.

1.2.9 State change

Several resources of the API possess a life cycle and a state for a given moment. Those resources are feedbacks, reports and operations.

For these resources, the state is always indicated in the response with the field *state*, and the next possible actions to change this state are displayed in the parameter *stateTransitions*. Example :

```
GET reports/{report}
```

Response (partial) :

```
{
  "type": "Report",
  "id": "cb7118b5-a821-4cf2-9475-0c0d0efdb8d0",
  "state": "NEW",
  "_embedded": {
    "stateTransitions": [
      "accept",
      "refuse"
    ]
  }
}
```

In the example above, the report has a state NEW and the possible actions on its state are *accept* and *refuse*.

Actions on the state of a resource is made through the POST method, with the path and the new value.

For example, to accept a report :

```
POST /reports/{report}/workflow/transition
```

```
{
                "transition": "accept"
}
```

This request will send the following response :

```
{
  "type": "Report",
  "id": "32219286-528a-4f97-b81e-fe7a8cb85707",
  "state": "ACCEPTED",
  "_embedded": {
    "stateTransitions": [
      "refuse",
      "hold",
      "progress"
    ]
  }
}
```

The report's state is now ACCEPTED, and the next actions are *refuse*, *hold* and *progress*.

Actions and states for each kind of resource are described in the appropriate sections of the documentation.

1.3 Authentication and connection

The Keyclic API uses the JSON Web Tokens to secure data transfer. Every request to the API is made with an access token allowing the server to verify the user's identity. When logging in, the user "receives" the token. This page goes through the process of creating an account, modifying an account, getting and using an accessToken.

For more informations on JWT standard, see : JSON Web Tokens official website.

1.3.1 Create a user account

A new user account is created with this request :

```
POST /security/register
```

```
{
    "email":"test@test.com",
    "password":"test"
}
```

This user gets a unique id and the role ROLE_USER, allowing him to use the API's basic functionalities.

1.3.2 Log in

Log in consists in sending one's credentials to the server to get an accessToken which will be used in future requests.

The connection is done with this request:

```
POST /security/login
```

```
{
    "login":"test@test.com",
    "password":"test"
}
```

If the credentials are known to the server, it returns an accessToken.

1.3.3 Use the token

Almost every single request to the API needs the user to be authenticated. The token is sent with the Authorization header with the prefix Bearer.

For example, to get the list of all feedbacks of the com.keyclic.app application, a user uses the endpoint :

```
GET /feedbacks
```

```
Request headers :
    X-Keyclic-App : com.keyclic.app
    Authorization : Bearer eyJhbGciOiJSUzI1NiJ9.
↪eyJyb2xlcyI6WyJPUkdBTklaQVRJT046QURNSU4iLCJST0xFX1VTRVIiXSwidXNlcm5hbWUiOiJ0ZXN0MjJAdGVzdC5jb20iLCJ
↪ZIqbBVSgJaXKj73IPYbFeEfie6FUIflv-ausUO-AAzVjPg8-jdhFv3nqsdOVJvE_
↪AB4bXjME1CRVFI7xD2SYCA8V6E6H2-y0XZE8SN_XTpHGMxDvOP27C2VUNQfPwgeWxjXzlDopo_
↪U9ybAEX4QdFhW14aeRgb9YWMDlzSD6VLgJO-LuprxX668Ajq5X9c8YND4D_p4WRDSQr8pqb3rTY9NQ6O34F-
↪OpDJlAUYj0pwMehYWpywVKJHRMv9xCRRoI8HrU6H3J3wo-K2OtQVJi9XFZ8g8sohw_ZaasG7dohxrO-
↪NtYSrOPXIXPI6kCDRuMi7sce06wfno1bC3jBoc83EhiBSBpDbWL98DSjPbF1SaCeE05aATfM5cMEXbnp8Iwh-
↪QLxglE4M-ZISJ8VooxzJxa7cWLlFW-iu0XWVFWrMbYgmSoU0PKRQB47w_
↪IOPxjWzDeMUTSA3esDwkxsYlNdS9SZe201EvI6zur5Ayot0PEGfAgex6Ew-
↪eKOHAfnuDiqeLQLbWs4Y69FO2DooWUhkfVGdl-IGglDPgk2AOs3w19e7mx-Gmm8DlUUr-
↪bK61NPPQ8dy7HPjXnU63-jbA17MAjHaRTO4eKopcZMWbpL-jgQjJltV3R5_0qNODaHCS_
↪auZs2cyqFN0HL9Rred5g7t6Fxyk-8MyyX0GiTyHsp3c
```

All requests to the Keyclic API need an accessToken in the headers, except for the following endpoints :

- create an account (POST /security/register)
- log in (POST /security/login)
- password change request (POST /security/password/change-request)
- password change (POST /security/password/change/{changePasswordToken})

1.3.4 Password change

A user who wishes to modify his password performs the next steps.

First, he sends a request to change his password :

```
POST /security/password/change-request
```

```
{
    "email":"test@test.com"
}
```

This request sends an email to the user with a link ending with a verification token. Example :

```
https://domain.com/#/password-reset/jrtVqBLxxoSA0c2hpsOBN-JQGQHGN3YXsKPMG1PWWWA
```

In the link above, jrtVqBLxxoSA0c2hpsOBN-JQGQHGN3YXsKPMG1PWWWA is the token to be sent in the next request. This part of the URL *https://domain.com/#/password-reset/* depends on the application used.

Then the user can change his password with :

```
POST /security/password/change/jrtVqBLxxoSA0c2hpsOBN-JQGQHGN3YXsKPMG1PWWWA
```

```
{
    "password":"newPassword"
}
```

1.3.5 Editing user data

For data other than the password, the user will request on this endpoint :

```
PATCH /people/{user}
```

For more information on PATCH request, see *Resource modification*.

For example, to change its last name :

```
{
    "familyName": "Family name"
}
```

A user can change the following data : his *familyName*, his *givenName*, his *image*, his *job* and his *email*.

1.4 Organization members and roles

When a user is part of an organization, he may receive from none to five roles, which define a set of permissions and possibles actions.

These are five roles :

- Administrator
- Agent
- Operator
- Analytic
- Export

A user may be part of several organizations, therefore his roles among the organizations may differ and are completely independent. A user who gets affiliated with an organization becomes a *member*. A *member* possesses the same basic functionalities as a simple user.

1.4.1 Administrator

Every user can create his own organization.

The user who created the organization becomes member and admin of that organization, his role is *Admin*.

The admin has the following permissions and privileges :

- Add new members,
- Change members' roles,
- Manage the organization's categories,
- Manage the organization's places,
- Manage the organization's partners,
- Consult and manage reports received and operations linked to these reports.

See : *Organizations*

See : *Reports*

Note : A user can be *ORGANIZATION:ADMIN* of several organizations and an organization can have several *ORGANIZATION:ADMIN*.

1.4.2 Agent

« Agent » is a special role suitable to members who only make feedbacks for their organization.

An agent has the possibility to :

- Activate Pro Mode (cf *Feedbacks by an agent*)

Note : A user can only be an ORGANIZATION:AGENT of one organization but an organization can have several ORGANIZATION:AGENT.

1.4.3 Operator

This role adds the following possibilities :

- Receive an assignment to an operation,
- Edit an operation (if assigned to that member) : change name, add pictures, change state, etc.

All members share those rights and other may stack if the member has more roles.

Note : An *ORGANIZATION:OPERATOR* can be member of several organizations and an organization can have several *ORGANIZATION:OPERATOR*.

1.4.4 Analytics

The role *Analytics* allows the member to access the organization statistics.

Note : A user can have the role *ORGANIZATION:ANALYTICS* for several organizations and an organization can have several members with the role *ORGANIZATION:ANALYTICS*.

1.4.5 Export

The role *Export* allows the member to export his organization's reports to the Excel format.

Note : A user can have the role *ORGANIZATION:EXPORT* for several organizations and an organization can have several members with the role *ORGANIZATION:EXPORT*.

1.4.6 Retrieving users

To get all of the application's users :

```
GET /people
```

To get a specific user :

```
GET /people/{user}
```

To filter users by pattern in email :

```
GET /people?search[email]=martin
```

To get members of an organization :

```
GET /organizations/{organization}/members
```

1.4.7 Example

Retrieving a user resource will display information about his membership(s), like the organization he is a part of, what roles he has and other miscellaneous details.

```
GET /people/5020c6ea-ca07-42d1-994f-d90b86703b1a/memberships
```


```
{
    "page": 1,
    "limit": 10,
    "pages": 1,
    "total": 1,
    "_links": {
        "self": {
            "href": "/people/5020c6ea-ca07-42d1-994f-d90b86703b1a/memberships?page=1&
↪limit=10"
        },
        "first": {
            "href": "/people/5020c6ea-ca07-42d1-994f-d90b86703b1a/memberships?page=1&
↪limit=10"
        },
        "last": {
            "href": "/people/5020c6ea-ca07-42d1-994f-d90b86703b1a/memberships?page=1&
↪limit=10"
        }
    },
    "_embedded": {
        "items": [
            {
                "id": "b0e7e28f-5b91-4c73-875e-8f34aa03553a",
                "roles": [
                    "ORGANIZATION:AGENT"
                ],
                "createdAt": "2018-02-27T10:00:00+02:00",
                "_links": {
                    "self": {
                        "href": "/organizations/84d36093-b8bc-47ad-bc8a-a043b3e301a9/
↪members/b0e7e28f-5b91-4c73-875e-8f34aa03553a",
```


(continues on next page)

(continued from previous page)


```
                        "iriTemplate": {
                            "mapping": {
                                "organization": "84d36093-b8bc-47ad-bc8a-a043b3e301a9
↪",
                                "member": "b0e7e28f-5b91-4c73-875e-8f34aa03553a"
                            }
                        }
                    },
                    "person": {
                        "href": "/people/5020c6ea-ca07-42d1-994f-d90b86703b1a",
                        "iriTemplate": {
                            "mapping": {
                                "person": "5020c6ea-ca07-42d1-994f-d90b86703b1a"
                            }
                        }
                    },
                    "organization": {
                        "href": "/organizations/84d36093-b8bc-47ad-bc8a-a043b3e301a9",
                        "iriTemplate": {
                            "mapping": {
                                "organization": "84d36093-b8bc-47ad-bc8a-a043b3e301a9"
                            }
                        }
                    }
                },
                "_embedded": {
                    "availableRoles": [
                        "ORGANIZATION:ADMIN",
                        "ORGANIZATION:ANALYTICS",
                        "ORGANIZATION:EXPORT",
                        "ORGANIZATION:READ_ONLY"
                    ]
                }
            }
        ]
    }
}
```


This json shows the user :

- is a member of an organization whose id is 84d36093-b8bc-47ad-bc8a-a043b3e301a9
- has the role ORGANIZATION:ADMIN : he is an admin of the organization 84d36093-b8bc-47ad-bc8a-a043b3e301a9
- has the role ORGANIZATION:AGENT : he is an agent of the organization 84d36093-b8bc-47ad-bc8a-a043b3e301a9
- is part of only one organization
- joined the organization February 27, 2018

CAUTION: The user id (5020c6ea-ca07-42d1-994f-d90b86703b1a) is not the same as the member id (b0e7e28f-5b91-4c73-875e-8f34aa03553a). The API identifies a member and a user as two differents entities.

1.5 Organizations

In the Keyclic app, an organization is an entity such as a company, a corporation, an association, a school, etc. to which feedbacks can be sent to be treated.

One or more members can be *Administrator* of that organization. An organization has at least one administrator.

An *Administrator* can manage the scope of operation of the organization by creating categories and places. When a user creates a feedback, geographic coordinates of that feedback are always automatically transmitted. Thus, the app can display all organizations in that place. Then the user has acces to information helping it choose the most fitting organization.

1.5.1 Creation

All users can create an organization :

```
POST /organizations
```

```
{
    "name":"organization name",
    "billingEmailAddress":"test@test.com",
    "notificationEmailAddress":"test@test.com"
}
```

The user becomes member and admin of this new organization.

To get all organizations of an application :

```
GET /organizations
```

It's possible to filter results by geographic point (see below : *Manage places*) :

```
GET /organizations?geo_coordinates=+44.851404209987386-0.5762618780136108
```

1.5.2 Manage members

To add a new member to the organization :

```
POST /organizations/{organization}/members
```

```
{
    "person":"63d07fc5-f4e6-471c-a8cc-3c3f227c8c2d"
}
```

This endpoint is reserved to a user who is *Administrator* and member of the organization {organization}.

To get organization's members :

```
GET /organizations/{organization}/members
```

To remove a member from the organization, an admin will request :

```
DELETE /organizations/{organization}/members/{member}
```

1.5.3 Manage places

An *Administrator* can create places, corresponding to areas where the organization can take actions :

```
POST /organizations/{organization}/places
```

```
{
    "name": "Test",
    "polygon":
    {
        "rings":
        [
            {
                "points":
                [
                    {
                        "longitude": 2.373991012573242,
                        "latitude": 48.84088179130599
                    },
                    {
                        "longitude": 2.3763084411621094,
                        "latitude": 48.84205393836751
                    },
                    {
                        "longitude": 2.376694679260254,
                        "latitude": 48.84189859515306
                    },
                    {
                        "longitude": 2.3787975311279297,
                        "latitude": 48.84041574931067
                    },
                    {
                        "longitude": 2.376115322113037,
                        "latitude": 48.839031720249054
                    },
                    {
                        "longitude": 2.373991012573242,
                        "latitude": 48.84088179130599
                    }
                ]
            }
        ],
        "srid": 4326
    }
}
```

To get all places of the application :

```
GET /places
```

This request may be filtered by organization and/or geographic points :

```
GET /places?geo_coordinates=+44.851404209987386-0.5762618780136108&organization=
↪{organization}
```

1.5.4 Manage categories

Categories are the business sectors of an organization. An *Administrator* can create a new category with a name, a color and an icon. The icon is chosen from Font Awesome.

```
POST /organizations/{organization}/categories
```

```
{
    "name":"Category's name",
    "color":"#ff0000",
    "icon":"fa-bug"
}
```

Those 3 properties can be edited with a PATCH request (see : *Resource modification*).

To get all categories of the application :

```
GET /categories
```

This request may be filtered by organization and/or geographic points :

```
GET /categories?geo_coordinates=+44.851404209987386-0.5762618780136108&organization=
↪{organization}
```

1.5.5 Manage partnership

An organization can have partners, i.e organizations affiliated with it. This relationship is one-sided : an organization A is a partner of organization B, but B is not necessarily one of B.

The partnership means that an *Administrator* can delegate a report to a partner organization. In the previous example, A can delegate a report to B, but B cannot delegate to A.

To add a new partner to the organization, an admin will send the request :

```
POST /organizations/{organization}/relationships
```

```
{
    "organization":"84d36093-b8bc-47ad-bc8a-a043b3e301a9"
}
```

To get an organization's partners :

```
GET /organizations/{organization}/relationships
```

The request is only available for admins.

1.6 Feedbacks

A feedback is always made in a given geographic point. This point is the most important and only mandatory component of a feedback. Optional parameters are the description, category, visibility and picture(s).

All users can make feedbacks.

1.6.1 Creation

```
POST /feedbacks/issues
```

Minimal example of a feedback:

```
{
    "businessActivity": "4bff7cb9-0fd2-4b44-9b0e-f6d17bb4ef36",
    "geo": {
        "elevation": 1,
        "point": {
            "latitude": 44.851343,
            "longitude": -0.576326
        }
    }
}
```

Complete example:

```
{
    "businessActivity": "4bff7cb9-0fd2-4b44-9b0e-f6d17bb4ef36",
    "category": "b0d007d5-e6ad-4113-b2b5-d8a1858a2fb1",
    "description": "Mon feedback 5",
    "geo": {
        "elevation":1,
        "point": {
            "latitude":44.851343,
            "longitude":-0.576326
        }
    },
    "visibility": "VISIBILITY_PUBLIC"
}
```

The default visibility of the observation is VISIBILITY_PRIVATE if this one is not provided in the payload.

Then the user can add one or more pictures to his feedback:

```
POST /feedbacks/{feedback}/images
```

```
{
    "image":"
↪iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAIAAAACDbGyAAAACXBIWXMAAAsTAAALEwEAmpwYAAAAB3RJTUUH4QIVDRUfvq7u+A
↪"
}
```

For more informations on adding images, see *Send files*.

1.6.2 Match with an organization

The Keyclic service doesn't just collect feedbacks, it sends them if possible, as *Reports*, to organizations capable of treating the feedback. Three cases are possible when transmitting a feedback :

- If the feedback's coordinates aren't in a place, then no organization will receive a report on this feedback.
- If the feedback's coordinates are in a place, then the report is sent the the organization in charge of the place.

- If the feedback's coordinates are in a place where two or more organizations can take action, and the user didn't specify a specific business activity, then several reports are generated and sent to all organizations in the place. The first one to accept will treat the problem.

For more informations about places, see *Manage places*.

1.6.3 Feedbacks by an agent

An *Agent* can post feedbacks the same way as every user. What's more, an agent can enter in "pro mode". To do so, just put in the body of the request, the "proMode" field with the value "true". Thus, his feedback will be treated differently:

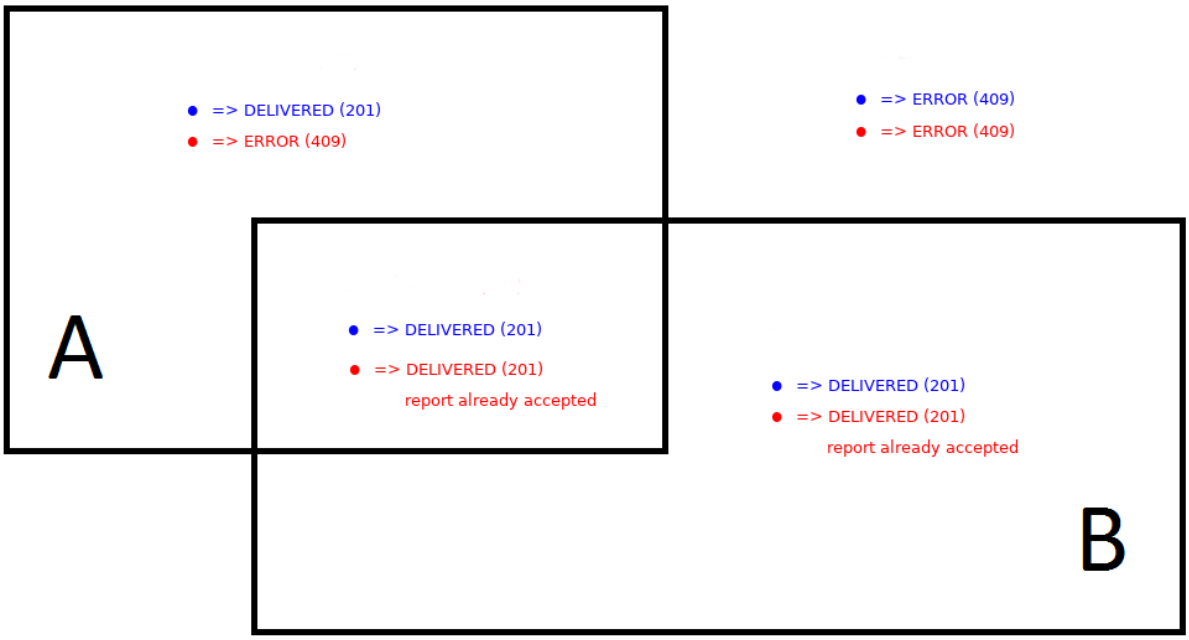
- If his feedback is within a place of his organization, a report is created.
- If his feedback is outside a place of his organization, then the feedback is refused.

1.6.4 Normal mode vs "Pro mode"

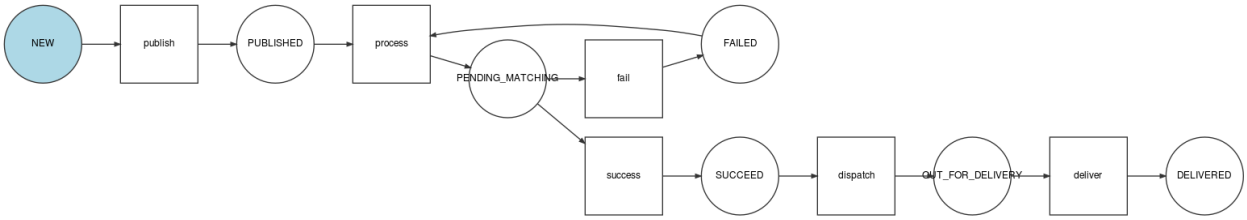
On the figure below, square A represents a place belonging to organization A, and square B to organization B.

Each dot is a feedback made by **a member of organization B**.

- In blue: feedbacks made in normal mode.
- In red: feedbacks made in pro mode (pro mode set to true in the request).



1.6.5 Life cycle overview



1.6.6 Get feedbacks

To get feedbacks, request the following endpoint:

```
GET /feedbacks
```

This request only returns feedbacks whose state is DELIVERED.

Some criteria may help filter feedbacks.

By state: state parameter

For example, to filter delivered feedbacks, a user will send the request:

```
GET /feedbacks?state=DELIVERED
```

Around a point: geo_near parameter

Example:

```
GET /feedbacks?geo_near[radius]=1000&geo_near[geo_coordinates]=+44.8-0.5
```

will return feedbacks within a 1000 meters radius from a point at latitude +44.8 and longitude 0.5.

Within a GeoHash: geo_hash parameter

Geohash is a public domain geocoding system [...] which encodes a geographic location into a short string of letters and digits. (Source: Wikipedia)

For more informations on Geohash, see:

- GeoHash official website
- GeoHash explorer

Feedbacks may be filtered with Geohash like this:

```
GET /feedbacks?geo_hash[]=ezzx&geo_hash[]=ezzz
```

This will return feedbacks between geohashes ezzx and ezzz.

By time period: before and after parameters

Example:

```
GET /feedbacks?after=2017-01-10T00:00:00+05:00&before=2017-02-22T23:59:59+05:00
```

will return feedbacks made between January 10 and February 22

Dates are written in the format: ISO 8601.

By organization

```
GET /feedbacks?organization={organization}
```

1.6.7 Comments

Users may comment feedbacks:

```
POST /feedbacks/{feedback}/comments
```

```
{
    "text": "My comment"
}
```

To get comments on a feedback:

```
GET /feedbacks/{feedback}/comments
```

1.6.8 Contributions

A user can also support a feedback this the following request, without body:

```
POST /feedbacks/{feedback}/contributions
```

To get all supports to a feedback:

```
GET /feedbacks/{feedback}/contributions
```

1.7 Reports

When a feedback is delivered (see : *Life cycle overview*), a report is created.

An *Administrator* gets all his organization's reports with :

```
GET /organizations/{organization}/reports
```

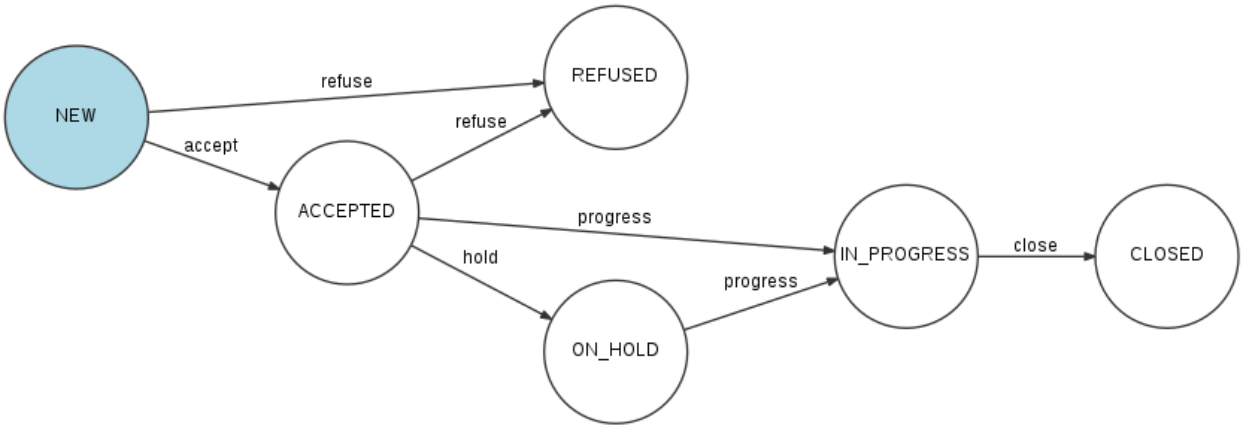
And a specific report with :

```
GET /reports/{report}
```

1.7.1 Life cycle

When a new report is generated by a feedback, its state is NEW.

Below is a representation of each possible state for a report and the actions on that report leading to this state.



There is a unique endpoint to change a report's state :

```
PATCH /reports/{report}/state
```

For example, to change the report's state from NEW to ACCEPTED, the admin will send the previous request with the following body :

```
{
    "transition": "accept"
}
```

A report can only be closed (state CLOSED), if every operation linked to this report has been either resolved or refused (see paragraph below *Operations*).

1.7.2 Operations

An operation is an action assigned to a member of the organization in relation with a report.

This endpoint sends back all operations resulting from a report :

```
GET /reports/{report}/operations
```

Creation and modification of an operation

An admin creates an operation on a report with this request :

```
POST /operations
```

```
{
    "description":"operation's description",
    "name":"operation's name",
    "report":"cb7118b5-a821-4cf2-9475-0c0d0efdb8d0"
}
```

A newly created operation has the state NEW.

One or more images can be added to an operation :

```
POST /operations/{operation}/images
```

```
{
    "image":"
↪iVBORw0KGgoAAAANSUhEUgAAAAUAAAAFCAIAAAACDbGyAAAACXBIWXMAAAsTAAALEwEAmpwYAAAAB3RJTUUH4QIVDRUfvq7u+AA
↪"
}
```

The description of an operation can be modified with the request :

```
PATCH /operations/{operation}
```

```
{
    "description": "New description"
}
```

Assignment

To assign an operation to a member, the admin sends the following request :

```
POST /operations/{operation}/assign
```

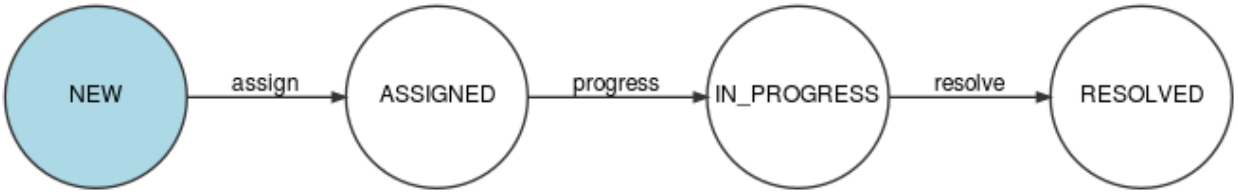
```
{
    "member": "{member}",
}
```

where {member} is the member's id the operation is assigned to.

In progress and resolved operation

After being assigned, the operation will be changed to "in progress" then "resolved", either by the assigned member or the admin.

Life cycle of an operation



Comments

It's possible to comment an operation :

```
POST /operations/{operation}/comments
```

```
{
    "text":"My comment"
}
```

To get all comments on an operation :

```
GET /operations/{operation}/comments
```

Logs of an operation

An admin can see the history of an operation :

```
GET /operations/{operation}/logs
```

1.7.3 Delegation

An admin can delegate a report to a partner organization.

See : *Manage partnership*

To delegate a report, an admin sends the following request :

```
POST /organizations/{organization}/delegates
```

```
{
    "report":"cb7118b5-a821-4cf2-9475-0c0d0efdb8d0",
    "organization":"a31d9ab7-9476-45f2-8cc7-033bf40bbcfa"
}
```

where {organization} is the organization's id delegating the report. And a31d9ab7-9476-45f2-8cc7-033bf40bbcfa is the receiving organization's id.

This report is shared between the delegating organization and the receiving organization. The latter will treat this report the same has every other report.

The partner organization may itself delegate this report to one of its partners and so on.

1.7.4 Export

An admin can export every report from his organization to the Excel format:

```
POST /organizations/{organization}/reports/exports
```

An archive containing the Excel file listing all reports and the associated images will be emailed to the authenticated admin.

1.8 Applications

An *application* in the Keyclic service limits the resources to a single domain. It means the use of specific clients app or website. For the eponymous client of the service, the *application* is "com.keyclic.app". Every client app and website using the key will share the same partition. (Here: the iPhone app Keyclic, the Android app Keyclic and the website https://app.keyclic.com)

Other *applications* exist in the Keyclic service with different keys, it's the case of "Vinci Mon Autoroute" available on iPhone and Android.

For example, from "Vinci Mon Autoroute":

- it is impossible to "send" a feedback sent to this *application* with the key "com.keyclic.app" and vice-versa,
- it is impossible to list feedbacks sent to the *application* with the key "com.keyclic.app" and vice-versa.

1.9 Notifications

The Keyclic service can notify users when some actions are made.

1.9.1 Notifications sent by action

Action	Target	Push	Wall	Mail
User register	The user			✓
Password change request	The user			✓
Feedback is commented	The user who made the feedback Users who commented or supported to the feedback	✓	✓	
Report is created	Administrators	✓	✓	✓
Report is accepted	The user who made the feedback Users who commented or supported to the feedback	✓	✓	
Report is closed	The user who made the feedback Users who commented or supported to the feedback	✓	✓	
Report is refused	The user who made the feedback Users who commented or supported to the feedback	✓	✓	
Report is delegated	Administrators	✓	✓	✓
Add document to report	Administrators	✓	✓	
Operation assigned	Member assigned to the operation	✓	✓	✓
Operation resolved	Administrator who created the operation	✓	✓	
Operation remind	Opertors The user who made the feedback	✓	✓	
Operation is commented	Member assigned to the operation Administrator who created the operation Members who commented the operation	✓	✓	
Feedback to review	Reviewer	✓	✓	

When a user makes an action which triggers a notification where it would be targeted. This user doesn't receive a notification. For example, if a member assigned to an operation leaves a comment on the operation, it doesn't receive a notification saying it commented the operation.

1.10 Integration

1.10.1 Webhooks

Unlike an API that requires continuous requests, the Keyclic service offers webhooks when certain events occur (see list below).

It is an easy and efficient way to call a service or application to get notifications and break free from the constant check.

Webhooks have various uses, such as :

- collect reports created for your data warehouse,
- sync reports and operations with your IS (ex: CMMIS, . . .),
- send notifications

1.10.2 Webhook creation and configuration

Webhooks can be created upon request from our service.

A webhook consists of the event for which you wish to be notified and a URL to where the notification should be sent.

Several webhooks can be created unlimitedly for each event type.

1.10.3 Event types

Event	Description
reportCreated	New report created
reportStateChanged	Report's state changed
operationCreated	New operation created
operationStateChanged	Operation's state changed
operationRemoved	Operation removed

1.10.4 Receipt of webhook notification

A webhook notification is sent in JSON format in the HTTP request body to the configured URL for that webhook. It contains the event name and a payload consisting of the resource details of the event. The resource varies according to the event.

note : The resource has the same serialization when requested to the API and in a webhook notification, it is possible to run through resources linked or embedded thanks to the HAL representation of our API.

```
{
    "event": "reportCreated",
    "payload": {
        "type": "Report",
        "id": "b0e7e28f-5b91-4c73-875e-8f34aa03553a",
        "state": "NEW",
        "createdAt": "2018-02-27T10:00:00+02:00",
        "updatedAt": "2018-02-27T10:00:00+02:00",
        "_links": {
            "feedback": "...",
            "operations": "...",
            "organization": "...",
            "tracking": "...",
        },
        "_embedded": {
            "stateTransitions": "...",
            "tracking": "..",
        }
    }
}
```

Partial example of a webhook notification received after the a new report is created.