MySlice REST API v1

1 API Specification

1.1 Base URL

All URLs referenced in the API documentation begin with the following base URL:

https://portal.onelab.eu/api/v1/

1.2 Resource Format

JSON resource representation is currently supported by the MySlice REST API

1.3 Request options

1.3.1 Field selection

It is possible to specify the resource fields to be returned with a request in the form:

```
GET /<resources>/<id>?fields=name1,name2,other_instance(other_name)
```

1.3.2 Field expansion

An instance reference contained in a resource can be expanded with the use of the `expand` option:

```
GET /<resources>/<id>?expand=authority
```

```
{
    "result": [{
        "created": "2013-12-23T15:50:34+00:00",
        "status": "enabled",
        "projects": [],
        "enabled": "2016-05-31T17:09:49.555000+00:00",
        "authority": {
              “id”: "urn:publicid:IDN++authority+sa",
              “name”: “Name of the Authority”,
              “shortname”: “shortname”
        }
        "updated": "2016-06-08T09:21:43+00:00",
        "pi_users": ["urn:publicid:IDN+onelab+user+myslice",
                     "urn:publicid:IDN+onelab:upmc+user+joshzhou16",
                     "urn:publicid:IDN+onelab:upmc+user+loic_baron"],
        "name": null,
        "hrn": "onelab",
        "slices": [],
        "id": "urn:publicid:IDN+onelab+authority+sa"
    }, {...}]
}
```

1.3.3 Pagination

```
GET /<resources>?expand=authority
```

1.4 Creating, Retrieving, Updating and Deleting resources

RETURN All requests will return:
{ error: null | <string>, debug: null | <string>, result: null | […] }

1.5 Websocket

Websocket /live
{ watch: [< activity | users | projects | slices | resources | testbeds >] }

watch is a list of entities to monitor in the same websocket

Auth: User

---

2 Authentication and Activity

2.1 User authentication and profile

2.1.1 POST /login

#TODO OAuth2

Example using python requests

```
payload = {'email': 'support@myslice.info', 'password': 'my_password'}
r = requests.post("http://localhost:8111/api/v1/login",
                  headers={str('Content-Type'):'application/json'},
                 data=json.dumps(payload),
                 timeout=self.timeout)
cookies = r.cookies
profile = requests.get('http://localhost:8111/api/v1/profile', cookies=self.cookies)
result = profile.text
```

2.1.2 GET /usertoken

This call sends back an encrypted token that contains the user’s information

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJwaV9hdXRoIjpbInVybjpwdWJsaWNpZDpJRE4rb25lbGFiOnVwbW
M6dGVzdCthdXRob3JpdHkrc2EiLCJ1cm46cHVibGljaWQ6SUROK29uZWxhYjp1cG1jOm1vYmljb20rYXV0aG9ya
XR5K3NhIiwidXJuOnB1YmxpY2lkOklETitvbmVsYWI6dXBtYzppZWVlK2F1dGhvcml0eStzYSIsInVybjpwdWJsaWN
pZDpJRE4rb25lbGFiOnVwbWM6dGVzdDIrYXV0aG9yaXR5K3NhIiwidXJuOnB1YmxpY2lkOklETitvbmVsYWI6dXB
tYzphemVxc2QrYXV0aG9yaXR5K3NhIiwidXJuOnB1YmxpY2lkOklETitvbmVsYWI6dXBtYzp3a3hxc2RsY3drK2F1d
...
```

2.1.3 POST /usertoken

The user token can be verified by sending it using POST
This call returns the user’s information

```
{
  "id": "urn:publicid:IDN+onelab:upmc+user+loic_baron",
  "pi_auth": [
    "urn:publicid:IDN+onelab:upmc:test+authority+sa",
    "urn:publicid:IDN+onelab+authority+sa",
    "urn:publicid:IDN+onelab:upmc+authority+sa",
  ],
  "slices": [
    "urn:publicid:IDN+onelab:upmc:test+slice+cloud",
    "urn:publicid:IDN+onelab:upmc:test+slice+xxx",
  ],
  "admin": false,
  "projects": [
    "urn:publicid:IDN+onelab:upmc:test+authority+sa",
  ]
}
```

2.1.4 GET /profile

```
{
   "result":{
      "slices":[
         {
            "id":"urn:publicid:IDN+onelab:upmc:totoproj1+slice+test1",
            "hrn":"onelab.upmc.totoproj1.test1",
            "status":"enabled",
            "shortname":"test1",
            "project":"urn:publicid:IDN+onelab:upmc:totoproj1+authority+sa"
         }
      ],
      "enabled":"2016-11-22T14:35:41.373000+00:00",
      "public_key":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDiiBAOF0dxudecEfMta+n6dhNx94KNLnp0FvUuE6s7WHZ5ceX7CB+
Vqi+7M2G9fOtFDd1RH9CrBzswW8ICJd4PWSNXeU3OVxaXNiY9OGIKu1j1sq+M/Z0pYPaJJ2yicF6li1yVpJGkdy3
ql4115tJZUoWSgQ4VIuDrIpW1h/Q/redv9qPfwbBZRsXCrb+1woJs/0i1uBWRsYr9Uv3LzpW2ufLF3DkvhEG16JwN
MVkU5hfAdM2q/vACm2IZMUB755qn4yIDvEVelpWRKAoSLUMk3yEBfMMC3neL+EkB+1rZg6PpBneIPx66C700kq
41akQiPUOq/qB3LE7JPy6XyM+h",
      "first_name":"toto",
      "private_key":"-----BEGIN RSA PRIVATE
KEY-----\nMIIEowIBAAKCAQEA4ogQDhdHcbnXnBHzLWvp+nYTcfeCjS56dBb1LhOrO1h2eXHl\n+wgflaovuzNhvX
zrRQ3dUR/Qqwc7MFvCAiXeD1kjV3lNzlcWlzYmPThiCrtY9bKv\njP2dKWD2iSdsonBepYtclaSRpHct6peNdebSWV
KFkoEOFSLg6yKVtYf0P63nb/aj\n38GwWUbFwq2/tcKCbP9ItbgVkbGK/VL9y86Vtrnyxdw5L4RBteicDTFZFOYXwH
TN\nqv7wAptiGTFAe+eap+MiA7xFXpaVkSgKEi1DJN8hAXzDAt53i/hJAfta2YOj6QZ3\niD8eugu9NJKuNWpElj1Dq
v6gdyxOyT8ul8jPoQIDAQABAoIBABACARnFz7DRKv57\n5D0kSuiojw+MJ90V2OW645Y0d0gJJHdziDFuQwZz/C
Mp6oZrPULoIRNvq0gStD44\nFbnKIZWLEoIcGoFHdTC397Qhd7LTcgy2v6INH/Mschosv4gBEdHNSycbSzFzPJA/\
nJ79vpr3/egc+zcHKRljyB4tIRrHN1FBTVYx0MzBJVzDTk7rGGmNlGftPmBdlFEfv\nlLaXwTYISw8t4kxPiLP2px2BP
0xyIMLJgSMF2nedFgfwrSnhfq9QQq/MjkG7FFcK\nBOUNMftbkl92uShFkqMG3sgNc04qMWkLU++aMnKC7DeTdF
qtWOCuzI0OuGS6ptZF\nqfe2hFMCgYEA5+KfxHvecbZpGb+Qr29i37TLN7mS9Foqx6OByXSVF/O0BqLjGXna\nXb
Y0LkYOpj/Zsx8dm6BBODpGdvTxQgb4MyX4OWaBy2E+IYyMPVHvODRR6E2cUuA/\nqGOldGjN/XH6SpxX6Sssd
unCHBAEBwE+tJZW6ymrMP+uwYC/gZBgYd8CgYEA+hbo\ndAjGwBjhz06SNlUkV2UHjmV2k9qo5YL4RwroBdlFq
7+3TtBYKW/aSibkq2L/CpQT\n5i5b3igBUk6B6oHxvaQCLjbzjV32I1MS49NRP/dnMjb7Ez3N4hgEKZxE5w7OiKso\n
GCWfPgTkOB2HCbOHJxtfU4weIc3xMsLT/3bD/n8CgYB92Cja6lqjndpWJfewriOb\nwrOTD3xoblLPO7ZhDYsoj5MB
Ev9qwQ24U/znrA+hO/+6zUU1Q3iBumapgm84ZS3o\nHFAlXrNMG/8rF+q4ELJh70sQZqZS9+60RTfzADnitSufuE+
hdFPSSTSWJD77Sjqs\nqPaM83U7x1chHu1PtHoKQwKBgQD46BHlYtLFYD5mgPHe57Cq1t/FZtC06X0OKOMI\ne
h9hqtS/0qkT5SLJ4wxknv8aYQYrtiN0BhHjMvfVvlNpXnmGYJTXAFQ5EBqpH8Z+\nP9TvEEKCZNxRU0L1UyfZbjH
Zsg/3UDowhklOhbnK7AB5tFfAoWqoEZ3v0TPNbMFo\n0gDR6wKBgC0vaG8HM9Flu710zBuZtUmnrWw9NiPdeqy
MR+eCMiMmfzQ7EuXqyflU\nj8qQxQfu6ROqFlLtxOo1LgjlieH414qB3qL2prJGGBewYgf4fpjKlietyEmgri06\nfRqT2
wYrGrnVpd0hFVyfGFBJmGNDtoJLDqWn1T7M4ceBcaTQfbqr\n-----END RSA PRIVATE KEY-----
",
      "pi_authorities":[],
      "email":"toto.onelab@yopmail.com",
      "last_name":"titi",
      "created":"2016-11-22T13:35:40+00:00",
      "authority":{
         "status":"enabled",
         "id":"urn:publicid:IDN+onelab:upmc+authority+sa",
         "hrn":"onelab.upmc",
         "name":"Universite Pierre et Marie Curie",
         "shortname":"upmc"
      },
      "id":"urn:publicid:IDN+onelab:upmc+user+toto_onelabqnir",
      "hrn":"onelab.upmc.toto_onelabqnir",
      "projects":[
         {
            "authority":"urn:publicid:IDN+onelab:upmc+authority+sa",
            "id":"urn:publicid:IDN+onelab:upmc:totoproj1+authority+sa",
            "hrn":"onelab.upmc.totoproj1",
            "name":"totoproj1",
            "status":"enabled",
            "shortname":"totoproj1"
         }
      ],
      "status":"enabled",
      "updated":"2016-11-22T13:35:40+00:00",
      "shortname":"toto_onelabqnir"
   }
}
```

2.2 Activity

2.2.1 GET /activity/[<id>]

```
filter: {
        action : [ <create|update|delete|add|remove>, … ],
        status : [ <new|pending|denied|approved|waiting|running|success|error|warning>, … ],
        object : [ <authority|user|project|slice|resource>, … ]
}
```

Auth: User

2.2.2 GET /requests/[<id>]

Auth: User

Request list or request with <id>

2.2.3 PUT /requests/<id>

{ action: <approve|deny|message>, message: <string> }

Auth: User

---

3 Entities

3.1 Authorities

3.1.1 GET /authorities

Permissions: everybody can get the list of authorities, needed for registration
If user is not logged in retrieve only id, shortname, name

Auth: Public

```
{
    "result": [{
        "status": "enabled",
        "Name": “Universite Pierre et Marie Curie“,
        "hrn": "onelab.upmc",
        "id": "urn:publicid:IDN+onelab:upmc+authority+sa"
    }, {...}]
}
```

Auth: Admin, PI, User part of Authority

```
{
    "result": [{
        "users": ["urn:publicid:IDN+onelab+user+myslice",
                  "urn:publicid:IDN+onelab+user+gurtov"],
        "created": "2013-12-23T15:50:34+00:00",
        "status": "enabled",
        "projects": [],
        "enabled": "2016-05-31T17:09:49.555000+00:00",
        "authority": "urn:publicid:IDN++authority+sa",
        "updated": "2016-06-08T09:21:43+00:00",
        "pi_users": ["urn:publicid:IDN+onelab+user+myslice",
                     "urn:publicid:IDN+onelab:upmc+user+joshzhou16",
                     "urn:publicid:IDN+onelab:upmc+user+loic_baron"],
        "name": null,
        "hrn": "onelab",
        "slices": [],
        "id": "urn:publicid:IDN+onelab+authority+sa"
    }, {...}]
}
```

Test:
test_get_authorities_0_unauth - *compares number of returned records from API and RethinkDB*
test_get_authorities_1_as_pi - *compares number of returned records from API and RethinkDB for PI user*
test_get_authorities_2_check_columns_unauth - *check the list of returned keys for API with expected values, when user unauthorized*
test_get_authorities_3_check_columns_as_pi - *check the list of returned keys for API with expected values from rethinkdb*
test_get_authorities_4_check_result_as_pi - *Check if returned values from API are correct in comparison to Rethinkdb values*
Normally this test will fail if previous test fails

3.1.2 GET /authorities/<id>

Permissions: Only logged in users
Auth: User is a member of this authority or is Admin

```
{
    "result": [{
        "users": ["urn:publicid:IDN+onelab+user+myslice",
                  "urn:publicid:IDN+onelab+user+gurtov"],
        "created": "2013-12-23T15:50:34+00:00",
        "status": "enabled",
        "projects": [],
        "enabled": "2016-05-31T17:09:49.555000+00:00",
        "authority": "urn:publicid:IDN++authority+sa",
        "updated": "2016-06-08T09:21:43+00:00",
        "pi_users": ["urn:publicid:IDN+onelab+user+myslice",
                     "urn:publicid:IDN+onelab:upmc+user+joshzhou16",
                     "urn:publicid:IDN+onelab:upmc+user+loic_baron"],
        "name": null,
        "hrn": "onelab",
        "slices": [],
        "id": "urn:publicid:IDN+onelab+authority+sa"
    }]
}
```

Tests:

test_get_authorities_id_0_as_unauth - Checking if API responds <<permission denied>> as unauth
test_get_authorities_id_1_as_pi - Checking if API responded with the same values as rethinkdb for the PI User

3.1.3 GET /users/authorities

Auth: User

Retrieve the authorities of the authenticated user

```
{
    "result": [{
        "hrn": "onelab.upmc",
        "id": "urn:publicid:IDN+onelab:upmc+authority+sa",
        "name": "Universite Pierre et Marie Curie",
        "pi_users": ["urn:publicid:IDN+onelab:upmc+user+loic_baron"],
        "projects": ["urn:publicid:IDN+onelab:upmc:ieee+authority+sa", ...],
        "authority": "urn:publicid:IDN+onelab+authority+sa",
        "updated": "2016-06-16T16:22:44+00:00",
        "users": ["urn:publicid:IDN+onelab:upmc+user+r_adomirklacza", ...],
        "status": "enabled",
        "created": "2013-12-23T16:00:07+00:00",
        "enabled": "2016-05-31T17:09:57.261000+00:00",
        "slices": ["urn:publicid:IDN+onelab:upmc+slice+newSlice", ...]
    }, … ]
}
```

Tests:
test_get_users_authorities_0_as_unauth - *Checking if API responds <<permission denied>> as unauth*
Open for the all (for the registration page)
test_get_users_authorities_1_as_pi - *Checking if API responded with the same values as rethinkdb as PI user*

3.1.4 POST /authorities

{ 'name': 'Example Authority', 'shortname': 'example_authority', 'authority':<root_authority> }
Create new authority (Service generates a request if user is not pi of root authority)

```
{
        "name": "Universite Pierre et Marie Curie",
        "shortname": "upmc",
        "domains": ["upmc.fr","lip6.fr"],
        "authority": "urn:publicid:IDN+onelab+authority+sa",
        "users": [
                {...},
                ],
        "pi_users": [
                "urn:publicid:IDN+onelab:upmc+user+loic_baron",
                {...},
                {"email":"email@domain.com"},
                ],
        "projects": [
                {...},
                ],
}
```

Auth: Public
Anyone can request the creation of a new authority at registration.
The request has to be approved by Admin.

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.1.5 PUT /authorities/<id>

Auth: Admin
Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.1.6 DELETE /authorities/<id>

Auth: PI of the authority or Admin
Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.2 Projects

3.2.1 GET /projects

Get all the public projects
Auth: User
Get all the projects of the authority of the logged in PI

Auth: PI of an authority
Get all the projects either private or public over all authorities
Auth: Admin

3.2.2 GET /projects/<id>

Permissions: can a user get the project of another user?
If the user is PI of the project or PI of the Authority or Admin
Auth: User

```
{
    "result": [{
        "id": "urn:publicid:IDN+onelab:upmc:z1+authority+sa",
        "updated": "2016-06-09T15:04:01+00:00",
        "users": [],
        "enabled": "2016-06-10T15:13:24.370000+00:00",
        "authority": {
               "status": "enabled",
               "id": "urn:publicid:IDN+onelab:upmc+authority+sa",
               "hrn": "onelab.upmc",
               "name": "Universite Pierre et Marie Curie"
        },
        "shortname": "z1",
        "created": "2016-06-09T15:04:01+00:00",
        "status": "enabled",
        "hrn": "onelab.upmc.z1",
        "pi_users": ["urn:publicid:IDN+onelab:upmc+user+loic_baron"],
        "name": "z1",
        "slices": []
    }, { … }]
}
```

3.2.3 GET /authorities/projects

Permissions: can a user get the list of users of a given authority? Only if he is part of the same authority?
Auth: User
Retrieve the projects of the authority of the logged in user

3.2.4 GET /authorities/<id>/projects

Permissions: can a user get the list of users of a given authority? Only if he is part of the same authority?
Auth: Admin
Retrieve the projects for the authority with <id>

```
{
    "result": [{
                        "shortname": "rescom_dem",
                        "hrn": "onelab.upmc.rescom_dem",
                        "pi_users":
                        ["urn:publicid:IDN+onelab:upmc+user+ciro_scognamiglio516016"],
                        "id": "urn:publicid:IDN+onelab:upmc:rescom_dem+authority+sa",
                        "status": "enabled",
                        "name": null,
                        "slices": ["urn:publicid:IDN+onelab:upmc:rescom_dem+slice+iotrobot"],
                        "updated": "2015-06-23T13:16:55+00:00",
                        "created": "2015-06-23T13:16:51+00:00",
                        "authority": {
                               "name": "Universite Pierre et Marie Curie",
                               "id": "urn:publicid:IDN+onelab:upmc+authority+sa",
                               "hrn": "onelab.upmc",
                               "status": "enabled"
                        },
                        "users": [],
                        "enabled": "2016-06-10T14:51:05.824000+00:00"
                }, {...}]
}
```

3.2.5 GET /users/projects

Auth: User

Retrieve the projects of the authenticated user

3.2.6 GET /users/<id>/projects

Permissions: can a user get the list of projects of another user?

Auth: Admin, PI of an authority can see users of this authority

Retrieve the projects of user <id>

```
{
    "result": [{
                        "status": "enabled",
                        "authority": {
                               "id": "urn:publicid:IDN+onelab:upmc+authority+sa",
                               "status": "enabled",
                               "name": "Universite Pierre et Marie Curie",
                               "hrn": "onelab.upmc"
                        },
                        "shortname": "mobicom",
                        "pi_users": ["urn:publicid:IDN+onelab:upmc+user+fadwa_boubekeur",
                              "urn:publicid:IDN+onelab:upmc+user+loic_baron"],
                        "enabled": "2016-06-10T15:13:24.248000+00:00",
                        "updated": "2015-09-05T07:59:55+00:00",
                        "id": "urn:publicid:IDN+onelab:upmc:mobicom+authority+sa",
                        "slices": ["urn:publicid:IDN+onelab:upmc:mobicom+slice+mobicom_demo",
                                "urn:publicid:IDN+onelab:upmc:mobicom+slice+exper_mobicom"],
                       "created": "2015-09-04T08:39:28+00:00",
                       "hrn": "onelab.upmc.mobicom",
                       "name": null,
                       "users": []
                }, {...}]
}
```

3.2.7 POST /projects

{ 'name': 'Example Project, 'description': 'Project Description' }
Create new project (Event will be processed by the service, generates a request if necessary)

Auth: User

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.2.8 PUT /projects/<id>

```
{
                       "shortname": "rescom_dem",
                       "hrn": "onelab.upmc.rescom_dem",
                       "pi_users":
                       ["urn:publicid:IDN+onelab:upmc+user+ciro_scognamiglio516016", ...],
                       "id": "urn:publicid:IDN+onelab:upmc:rescom_dem+authority+sa",
                       "status": "enabled",
                       "name": null,
                       "slices": ["urn:publicid:IDN+onelab:upmc:rescom_dem+slice+iotrobot",
...],
                       "authority": "urn:publicid:IDN+onelab:upmc:rescom_dem+authority+sa",
                       "users": [...],
   }
```

Auth: User is PI of the project or Admin

pi_users is processed by an Event ADD PI to Project

Return:
{
events: [<id>, <id>,...],
result: "success",

error: None,
debug: None,
}

3.2.9 DELETE /projects/<id>

Auth: User is PI of the project or PI of the Authority or Admin

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.3 Users

3.3.1 GET /users

Permissions: can a user get the complete list of users?
Auth: Admin or PI of an authority can get the users of his authority

```
{
    "result": [{
                       "created": "2014-09-18T08:15:52+00:00",
                       "status": "enabled",
                       "hrn": "onelab.csu.mayyash",
                       "updated": "2014-09-18T08:15:52+00:00",
                       "authority": {
                              "id": "urn:publicid:IDN+onelab:csu+authority+sa",
                              "name": "Chicago State University",
                              "hrn": "onelab.csu",
                              "status": "enabled"
                       },
                       "projects": [],
                       "id": "urn:publicid:IDN+onelab:csu+user+mayyash",
                       "shortname": "mayyash",
                       "email": "mayyash@csu.edu",
                       "enabled": "2016-05-27T16:40:18.025000+00:00"
               }, { … }]
}
```

3.3.2 GET /users/<id>

Permissions: can a user get the data of another user?
Auth: Admin or PI of an authority can see the users of his authority

```
{
    "result": [{
        "status": "enabled",
        "enabled": "2016-05-27T16:40:18.025000+00:00",
        "hrn": "onelab.csu.mayyash",
        "id": "urn:publicid:IDN+onelab:csu+user+mayyash",
        "email": "mayyash@csu.edu",
        "shortname": "mayyash",
        "updated": "2014-09-18T08:15:52+00:00",
        "created": "2014-09-18T08:15:52+00:00",
        "projects": [],
        "authority": {
               "status": "enabled",
               "name": "Chicago State University",
               "hrn": "onelab.csu",
               "id": "urn:publicid:IDN+onelab:csu+authority+sa"
        }
    }]
}
```

3.3.3 GET /authorities/users

Auth: User
Retrieve the users of the authority of the logged in user

3.3.4 GET /authorities/<id>/users

Permissions: can a user get the list of users of a given authority? Only if he is part of the same authority?
Auth: Admin
Retrieve the users of authority<id>

```
{
    "result": [{
        "shortname": "gurtov",
        "hrn": "onelab.gurtov",
        "updated": "2015-03-11T08:56:42+00:00",
        "projects": [],
        "email": "gurtov@hiit.fi",
        "created": "2015-03-11T08:56:42+00:00",
        "id": "urn:publicid:IDN+onelab+user+gurtov",
        "status": "enabled",
        "authority": {
               "name": null,
               "id": "urn:publicid:IDN+onelab+authority+sa",
                "hrn": "onelab",
                "status": "enabled"
        },
        "enabled": "2016-05-27T16:40:17.622000+00:00"
    }, {...}]
}
```

3.3.5 GET /projects/<id>/users

Permissions: can a user get the list of users of a given project?
Auth: User
Retrieve the users for project <id>

```
{
    "result": [{
        "shortname": "loic_baron",
        "id": "urn:publicid:IDN+onelab:upmc+user+loic_baron",
        "updated": "2016-06-10T15:52:15+00:00",
        "projects": [ … ],
        "email": "loic.baron@lip6.fr",
        "enabled": "2016-05-27T16:40:38.436000+00:00",
        "hrn": "onelab.upmc.loic_baron",
        "status": "enabled",
        "authority": {
                "status": "enabled",
                "id": "urn:publicid:IDN+onelab:upmc+authority+sa",
                "hrn": "onelab.upmc",
                "name": "Universite Pierre et Marie Curie"
        },
        "created": "2013-12-23T16:21:02+00:00"
    }, {...}]
}
```

3.3.6 GET /slices/<id>/users

Retrieve the users that are in the slice/id

3.3.7 POST /users

{ authority: <authority_id>, first_name: <string>, last_name: <string>, email: <email>, password: <string>, terms: <bool> }
Auth: Client

Create a new user. A new event (request) is created and put on pending

```
{
  "first_name": "Loic",
  "last_name": "Test",
  "shortname": "loic_test",
  "email": "loic.test@yopmail.com",
  "authority": "urn:publicid:IDN+onelab:upmc+authority+sa",
  "terms": true,
  "generate_keys": true,
  "private_key": null,
  "public_key": null,
  "credentials": [],
  "certificate": "",
  "status": "enabled",
  "password": "12341234",
  "keys": [],
  "projects": [
    "urn:publicid:IDN+onelab:upmc:test+authority+sa"
  ],
  "slices": [
    "urn:publicid:IDN+onelab:upmc:test+slice+cloud"
  ],
  "pi_authorities": [
    "urn:publicid:IDN+onelab:upmc+authority+sa",
    "urn:publicid:IDN+onelab:upmc:test+authority+sa"
  ],
  "logs": []
}
```

Slices are automatically processed by the service based on the list of projects
projects -> ADD PI to Project
pi_authorities -> ADD PI to authority
TBD: these have to be processed once the user is created

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.3.8 PUT /users/<id>

Auth: User
Create a new user. A new event (request) is created and put on pending

```
{
  "id": "urn:publicid:IDN+onelab:upmc+user+loic_testpgoy",
  "first_name": "Loic 1",
  "last_name": "Test 1",
  "shortname": "loic_test",
  "email": "loic.test@yopmail.com",
  "authority": "urn:publicid:IDN+onelab:upmc+authority+sa",
  "terms": true,
  "generate_keys": true,
  "private_key": null,
  "public_key": null,
  "credentials": [],
  "certificate": "",
  "status": "enabled",
  "password": "12341234",
  "keys": [],
  "projects": [
    "urn:publicid:IDN+onelab:upmc:test+authority+sa"
  ],
  "slices": [
    "urn:publicid:IDN+onelab:upmc:test+slice+cloud"
  ],
  "pi_authorities": [
    "urn:publicid:IDN+onelab:upmc+authority+sa",
    "urn:publicid:IDN+onelab:upmc:test+authority+sa"
  ],
  "logs": []
}
```

Slices are automatically processed by the service based on the list of projects

Projects, authority and pi_authorities can be either ids or a dict

```
"projects": [
   "urn:publicid:IDN+onelab:upmc:test+authority+sa"
 ],
```

OR

```
"projects": [
        {
          "status": "enabled",
          "authority": "urn:publicid:IDN+onelab:upmc+authority+sa",
          "shortname": "test",
          "name": "Test",
          "hrn": "onelab.upmc.test",
          "id": "urn:publicid:IDN+onelab:upmc:test+authority+sa"
        }
],
```

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.3.9 DELETE /users/<id>

Auth: User PI on the authority of the user to be deleted or PI of an upper authority

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.4 Slices

3.4.1 GET /slices

3.4.2 GET /slices/<id>

Auth: Users

3.4.3 GET /projects/<id>/slices

Permissions: can a user get the list of slices of a given project?
Auth: User
Retrieve the slices for project <id>

```
{
    "result": [{
        "shortname": "cloud",
        "users": ["urn:publicid:IDN+onelab:upmc+user+loic_baron"],
        "name": "onelab.upmc.test.cloud",
        "created": "2015-10-05T14:21:06+00:00",
        "updated": "2015-10-05T14:21:06+00:00",
        "id": "urn:publicid:IDN+onelab:upmc:test+slice+cloud",
        "status": "enabled",
        "project": {
               "shortname": "test",
               "name": null,
               "id": "urn:publicid:IDN+onelab:upmc:test+authority+sa",
               "status": "enabled",
               "hrn": "onelab.upmc.test",
               "authority": "urn:publicid:IDN+onelab:upmc+authority+sa"
        },
        "authority": {
               "status": "enabled",
               "name": "Universite Pierre et Marie Curie",
               "hrn": "onelab.upmc",
               "id": "urn:publicid:IDN+onelab:upmc+authority+sa"
        },
        "hrn": "onelab.upmc.test.cloud",
        "enabled": "2016-03-21T18:15:48.498000+00:00"
    }, {...} ]
}
```

3.4.4 GET /users/slices

Auth: User

Retrieve the slices of the authenticated user

3.4.5 GET /users/<id>/slices?expand=true

IT DOES NO WORK

Permissions: can a user get the list of slices of another user?

Auth: User

Retrieve the slices of user <id>

```
{
    "result": [{
        "updated": "2015-07-24T08:58:48+00:00",
        "created": "2015-07-24T08:58:48+00:00",
        "users": ["urn:publicid:IDN+onelab:certhple+user+harniavis",
                  "urn:publicid:IDN+onelab:uth+user+ardadouk",
                  "urn:publicid:IDN+onelab:certhple+user+dostavro"],
        "authority": {
               "name": "Centre for Research and Technology Hellas",
               "id": "urn:publicid:IDN+onelab:certhple+authority+sa",
               "status": "enabled",
               "hrn": "onelab.certhple"
        },
        "shortname": "nitos_test",
        "name": "onelab.certhple.nitos_test.nitos_test",
        "project": {
               "authority": "urn:publicid:IDN+onelab:certhple+authority+sa",
               "hrn": "onelab.certhple.nitos_test",
               "name": null,
               "id": "urn:publicid:IDN+onelab:certhple:nitos_test+authority+sa",
               "status": "enabled",
               "shortname": "nitos_test"
        },
        "enabled": "2016-03-21T18:15:41.126000+00:00",
        "id": "urn:publicid:IDN+onelab:certhple:nitos_test+slice+nitos_test",
        "status": "enabled",
        "hrn": "onelab.certhple.nitos_test.nitos_test"
    }, { ... }]
}
```

3.4.6 GET /resources/<id>/slices

Is this implemented ??
Auth: Client

3.4.7 POST /slices

{ 'name': 'Example Slice, 'shortname': 'example_slice' , projet : {“id”: “urn:publicid:IDN+onelab:upmc:nitos_tut+authority+sa“}}

Auth: User

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.4.8 PUT /slices/<id>

Auth: User

```
{
    "updated": "2015-07-24T08:58:48+00:00",
    "created": "2015-07-24T08:58:48+00:00",
    "users": ["urn:publicid:IDN+onelab:certhple+user+harniavis",
              "urn:publicid:IDN+onelab:uth+user+ardadouk",
              "urn:publicid:IDN+onelab:certhple+user+dostavro"],
    "authority": {
        "name": "Centre for Research and Technology Hellas",
        "id": "urn:publicid:IDN+onelab:certhple+authority+sa",
        "status": "enabled",
        "hrn": "onelab.certhple"
    },
    "shortname": "nitos_test",
    "name": "onelab.certhple.nitos_test.nitos_test",
    "project": {
        "authority": "urn:publicid:IDN+onelab:certhple+authority+sa",
        "hrn": "onelab.certhple.nitos_test",
        "name": null,
        "id": "urn:publicid:IDN+onelab:certhple:nitos_test+authority+sa",
        "status": "enabled",
        "shortname": "nitos_test"
    },
    "enabled": "2016-03-21T18:15:41.126000+00:00",
    "id": "urn:publicid:IDN+onelab:certhple:nitos_test+slice+nitos_test",
    "status": "enabled",
    "hrn": "onelab.certhple.nitos_test.nitos_test",
    "resources": ["urn:publicid:IDN+iotlab+node+a8-100.grenoble.iot-lab.info"],
    "leases": ["fbd6298f-953b-43cb-a9db-f4a0c7558333"],
}
```

UPDATE Slice Event

TBD When the resources are not a list of ids but a list of dict
(to configure the resource with sliver_type and image)

Leases: handled by POST /leases and DELETE /leases/<id>

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}
/// this is not tested, because this requires reservation of real resources so maybe automate testing is not the best option

3.4.9 DELETE /slices/<id>

Auth: User is a member of the slice

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.5 Resources

3.5.1 GET /resources[?timestamp_start=<XXX>&timestamp_end=<XXX>]

Optional params
timestamp_start <timestamp>
timestamp_end <timestamp>

Returns Resources [available during this period of time]
TODO: resources reserved within this time range (reserved:true|false / default false)

Auth: Client

3.5.2 GET /resources/<id>

Auth: Client

3.5.3 GET /slices/<id>/resources

3.5.4 GET /testbeds/<id>/resources[?timestamp_start=<XXX>&timestamp_end=<XXX>]

Optional params
timestamp_start <timestamp>
timestamp_end <timestamp>

Returns Resources of the testbed [available during this period of time]
TODO: resources reserved within this time range (reserved:true|false / default false)

3.5.5 GET /testbeds/<id>/leases?timestamp_start=<XXX>&timestamp_end=<XXX>

params
timestamp_start <timestamp>
timestamp_end <timestamp>

Returns Leases

3.5.6 POST /resources

Mandatory fields:

- name
- testbed (testbed_id OR user_id (F-Interop))
- owner (user_id)

Optional:

- visibility: public | protected (project members) | private (user)
- type
- flavor
- image
- version
- configuration (script)
- country
- city
- location (GPS coordinates)
- protocols: [coap, oneM2M, ipv6]
- ... (extensions)

Generated

- id
- manager
- status: pending | provisioning | ready

Examples of a VM to create on an openstack server (will be created on slice update)
{ 'name': 'server 1', 'shortname': 'server_1', id:'<URN>', testbed:'<testbed id>', manager: 'urn:publicid:IDN+onelab-cloud+authority+am',visibility:'public|private|protected', type:'openstack', flavor:'m1.tiny|m1.small|m1.large', image:'cirros-0.3.3-x86_64|ubuntu-14.04', status:'pending|provisioning|ready' }

Examples of a user's IoT device (no provisioning, user manually install the F-Interop agent)
{ 'name': 'iot 1', 'shortname': 'iot_1', id:'<URN>', testbed:'<user id>', manager: '<user id>',visibility:'public|private|protected', type:'software implementation | M3 node', version: 'x.x', protocols: [coap, OneM2M], status:'pending|ready', ip:'<ip>', country:'France', location:'<GPS coordinates>', city:'Paris' }
Auth: User

TBD: these resources created into RethinkDB it is just local
Event CREATE Resource -> Service will insert into RethinkDB.
This event doesn't trigger any myslicelib call (no impact on testbeds).

Once the resource is added to a Slice (PUT /slice/<id>), it may trigger Events to reserve the resource at the testbed.

For user's owned resources added to a slice, the GUI will provide a tutorial how to manually connect the resource (Experimental plane).

TBD: GUI
Some testbeds allow the creation of resources some don't.
Which fields are mandatory depending on the testbed and the type of resource selected

Return:
{?
result: "success",
error: None,
debug: None,
}

3.5.7 PUT /resources/<id>

Condition: Resource has field owner and owner is not null

We can not change the following fields of a resource, it would change the id generated:
- name
- testbed

For user's owned resources: No impact on testbeds
UPDATE Resource

Virtual resources like OpenStack:
Search for a slice that uses this resource.
Update of resources should trigger a Slice Update Event.
In the worker of slice management service, we have to compare the resource properties with the previous values... TBD How to do that?
Then slice.removeResource(r) & slice.addResource(r)
This will remove the previous values and add the new ones.
For now, the VM will be destroyed and a new one will be created on Slice Update.
This behavior is a feature of the OpenStack AM.
Auth: User that own the resource
or is a member of a slice in which the resource is used (for virtual resources)

Return:
{
result: "success",

error: None,
debug: None,
}

3.5.8 DELETE /resources/<id>

Condition: Resource has field owner and owner is not null
Deletion of resources should trigger a Slice REMOVE resource Event
Auth: User that own the resource
or is a member of a slice in which the resource is used (for virtual resources)

Return:
{
result: "success",
error: None,
debug: None,
}

3.6 Leases

3.6.1 GET /leases

```
[{
      "slice_id": "urn:publicid:IDN+iotlab+slice+tanaka_slice",
      "parser": "iotlab",
      "start_time": 1475718144,
      "id": "fbd6298f-953b-43cb-a9db-f4a0c7558333",
      "duration": 3000,
      "end_time": 1475718194,
      "resources": [
        "urn:publicid:IDN+iotlab+node+m3-131.lille.iot-lab.info",
        ".....”,
      ],
      "testbed": "urn:publicid:IDN+iotlab+authority+am"
}]
```

3.6.2 GET /leases[?timestamp_start=<XXX>&timestamp_end=<XXX>]

3.6.3 GET /testbeds/<id>/leases[?timestamp_start=<XXX>&timestamp_end=<XXX>]

3.6.4 GET /resources/<id>/leases

3.6.5 POST /leases

Requires slice_id, start_time, duration, resources, testbed

```
{
      "slice_id": "urn:publicid:IDN+iotlab+slice+tanaka_slice",
      "resources": [
        "urn:publicid:IDN+iotlab+node+m3-131.lille.iot-lab.info",
        ".....”,
      ],
      "testbed": "urn:publicid:IDN+iotlab+authority+am",
      "duration": 3000,
      "start_time": 1475718144,
      "end_time": 1475718194,
}
```

Optional parameters:

- **start_time**: if not specified, it will be now + 5 min (adding 2 minutes just to be sure that the start_time won't expire before reaching the testbed)
- **end_time**: can be calculated based on start_time and duration

TODO: testbed parameter should be optional
We have to take care of spliting leases per testbed in the backend (REST API)

- **testbed**: we can post a lease with resources from different testbeds, the API will split the request in different leases sent to the relevant testbeds, the result will be several leases.

TODO: use only PUT slices
PUT slices should handle both resources and leases
It would be easier for the frontend...
The backend (REST API) should then analyze if the resources need to be scheduled or not depending on the testbed.hasLeases
Should we also split the request in several calls?

Auth: User is a member of the slice linked to this lease

Return:
{
events: [<id>],
result: "success",
error: None,
debug: None,
}

3.6.6 PUT /leases/<id>

TBD: does it make sens?
It seems it does with the Nitos implementation...
Auth: User who is a member of the slice linked to this lease

Return:
{
events: [<id>],
result: "success",
error: None,
debug: None,
}

3.6.7 DELETE /leases/<id>

Does it make sens? Is it supported by the testbeds?
Auth: User who is a member of the slice linked to this lease
Return:
{
events: [<id>],
result: "success",
error: None,
debug: None,
}

3.7 Testbeds

3.7.1 GET /testbeds

3.7.2 GET /testbeds/<id>

3.7.3 POST /testbeds

Admin feature to add a new testbed
Auth: Admin

TBD: no event will be created, but the config of the portal should be changed

Return:
{
result: "success",
error: None,
debug: None,
}

3.7.4 PUT /testbeds/<id>

Admin feature to update a testbed
Auth: Admin

TBD: if we disable a testbed, all resources should be set as NOT available

Return:
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}

3.7.5 DELETE /testbeds/<id>

Admin feature to delete a testbed
Auth: Admin

TBD: we should remove all resources related to this testbed, update the slices using these resources.

Return:

```
{
events: [<id>, <id>,...],
result: "success",
error: None,
debug: None,
}
```