Abstract

AEON open API allows you to implement your resources management in your own way. This is a key point of AEON, your applications could have flexibility enough, not only to send/receive data, but also to configure your environment in a very dynamic way. For example, a chat application could manage rooms creating and deleting entities and channels regarding the needs.

If you are a developer, this should be your section. Here you will find a detailed description of the REST API for resources managment and pub/sub functionality.

Table of Contents

API Summary

- Default
  - POST - Login [/login]
  - GET - Logout [/logout]
  - Users
    - GET - List All Users [/users]
    - POST - Create User [/users]
  - GET - User info [/users/user]
  - User Collection
    - GET - Get user [/users/{user_id}]
    - DELETE - Delete User [/users/{user_id}]
  - Update User
    - PUT - Update User Password [/users/{user_id}/updatePassword]
  - Update User
    - GET - Remember User Password [/users/{user_id}/rememberPassword]
  - Verify User Password
    - PUT - Reset User Password [/users/{user_id}/rememberPassword/{code}]
  - Entities
    - GET - Get all entities [/entities]
    - POST - Create entity [/entities]
  - Entities Collection
    - GET - Entity Info [/entities/{entity_id}]
    - PUT - Update Entity [/entities/{entity_id}]
    - DELETE - Delete Entity [/entities/{entity_id}]
  - Channels
    - GET - Get channels [/entities/{entity_id}/channels]
    - POST - Create channel [/entities/{entity_id}/channels]
  - Channels Collection
    - GET - Channel Info [/entities/{entity_id}/channels/{channel_id}]
    - PUT - Update Channel [/entities/{entity_id}/channels/{channel_id}]
    - DELETE - Delete Channel [/entities/{entity_id}/channels/{channel_id}]

User operations

Documentation for user's management. Most of the AEON's functionalities requires registered users. This registered users will make use of the AEON's functionality to create entities and channels.

- /login
- /logout
- /users
- /users/user
- /users/{user_id}
- /users/{user_id}/updatePassword

- /users/{user_id}/rememberPassword
- /users/{user_id}/rememberPassword/{code}

Entities operations

Documentation for entity’s management. AEON entities are the basic resource to organize your different communication channels. An entity could mean whatever you want, a truck, a box of fish, a chat application, a chat room, your personal mobile, whatever.. By the moment, entities contains just a little description.

Only users can create new entities, and these entities will contain the different channels.

- /entities
- /entities/{entity_id}

Channels operations

AEON channels are the most valuable resource, actually, you are here because you want communication channels. A channel is a communication resource allowing publication an subscription mechanisms.

Channels need to be organized into an existing entity and contains: - A little description - The pub/sub urls - A list of allowed subscribers

Operations to work with channels are:

- /entities/{entity_id}/channels
- /entities/{entity_id}/channels/{channel_id}

API Specification

Default

Login **[/login]**

Login

POST /login

Logging process with an existing user and password (POST /users/) . If the process results “ok” you will receive a cookie with your session. This cookie will be used in most of the operations.

Request (application/json)

Response 200 (application/json)

Go to example

Logout **[/logout]**

Logout

GET /logout

Logout process. The user set in the cookie will be unset from it. The cookie won’t be deleted from the server.

Response 200 (application/json)

Go to example

Users [/users]

List All Users

GET /users

Get list of registered users. Just basic information. Preconditions:

- You need to be logged: (Login)

Response 200 (application/json)

Go to example

Create User

POST /users

Create/Register a new user into the system. By the moment, it only includes information about user and password.

Request (application/json)

Response 200 (application/json)

Go to example

User info [/users/user]

User info

GET /users/user

Retrieves the information related to the user identified in the cookie.

Response 200 (application/json)

Go to example

User Collection **[/users/{user_id}]**

- Parameters:
  - user_id - User id

Get user

GET /users/{user_id}

Get complete information of an specific user. Preconditions:

- In order to get extra information of an existing user you need to be logged
- You need to be logged as (user_id): (Login)

Response 200 (application/json)

Go to example

Delete User

DELETE /users/{user_id}

Delete an specific user. When a user is deleted, all the entities and the information assigned will be deleted too. Preconditions:

- In order to delete an existing user you need to be logged
- You need to be logged as (user_id): (Login)

Response 200 (application/json)

Go to example

Update User **[/users/{user_id}/updatePassword]**

- Parameters:
  - user_id - User id

Update User Password

PUT /users/{user_id}/updatePassword

Change the user password.

Response 200 (application/json)

Go to example

Update User **[/users/{user_id}/rememberPassword]**

This function is in charge of helping to remember a forgotten or missed password. Mainly, an existing user id will be requested (users are registered with emails). Thus, the user will receive an email with extra information in his email address. The extra information contains a temporal code used to reset the password with (Reset the user’s password). Generate a temporal code to change the user’s password. (user_id) is the email of an existing user. Through this email he will receive the necessary information to (Reset the user’s password).

- Parameters:
  - user_id - User id

Remember User Password

GET /users/{user_id}/rememberPassword

Response 200 (application/json)

Go to example

Verify User Password **[/users/{user_id}/rememberPassword/{code}]**

This function is in charge of resetting a forgotten or missed password. The user needs to provide an existing user id (users are registered with emails), the new password and a temporal code generated by (Remember the user’s password). Changes user’s password of (user_id) using the retrieved (code) by (Remember the user’s password).

- Parameters:
  - user_id - User id
  - code - Verfication code sent to the user by email

Reset User Password

PUT /users/{user_id}/rememberPassword/{code}

Request (application/json)

Response 200 (application/json)

Go to example

Entities **[/entities]**

Get all entities

GET /entities

Get list of entities owned by the logged user. Preconditions:

- You need to be logged: (Login)

Response 200 (application/json)

Go to example

Create entity

POST /entities

Create a new entity into the system. By the moment, entities only contains information about description and a list of channels. Some preconditions:

- You need to be logged: (Login)

The entity will be created and owned by the logged user.

Request (application/json)

Response 200 (application/json)

Go to example

Entities Collection **[/entities/{entity_id}]**

- Parameters:
    - entity_id - Entity Id

Entity Info

GET /entities/{entity_id}

Get complete information of an specific entity. Preconditions:

- In order to get extra information of an existing entity you need to be logged
- You need to be logged (Login) as the owner of (entity_id):

Response 200 (application/json)

Go to example

Update Entity

PUT /entities/{entity_id}

Update the name or the description of an entity. Preconditions:

- In order to delete an existing user you need to be logged
- You need to be logged as the owner of the entity: (Login)

Request (application/json)

Response 200 (application/json)

Go to example

Delete Entity

DELETE /entities/{entity_id}

Delete an specific entity. When an entity is deleted, the channels that belongs to it will be deleted too. Preconditions:

- In order to delete an existing entity you need to be logged
- You need to be logged as the owner of the entity: (Login)

Response 200 (application/json)

Go to example

Channels **[/entities/{entity_id}/channels]**

- Parameters:
  - entity_id: Entity Id

Get channels

GET /entities/{entity_id}/channels

Get the list of channels of an specific entity. Preconditions:

- You need to be logged: (Login)
- You need to be logged as the owner of the (entity_id)

Response 200 (application/json)

Go to example

Create channel

POST /entities/{entity_id}/channels

Create a new channel into the specific entity. Channels contains information about description and the pub/sub mechanisms. Some preconditions: - You need to be logged: (Login) - You need to be logged as the owner of the (entity_id) where your are requesting a new channel

The channel will be attached to the entity, together with Pub_Url and a (Sub_Url).

Request (application/json)

Response 200 (application/json)

Go to example

Channels Collection **[/entities/{entity_id}/channels/{channel_id}]**

- Parameters:
  - entity_id - Entity Id
  - channel_id - Channel Id

Channel Info

GET /entities/{entity_id}/channels/{channel_id}

Get complete information of an specific channel. Preconditions:

- You need to be logged: (Login)
- You need to be logged as the owner of the (entity_id)
- (channel_id) has to be attached to the list of channels of (entity_id)

Response 200 (application/json)

Go to example

Update Channel

PUT /entities/{entity_id}/channels/{channel_id}

Updates the information of an specific channel. Preconditions:

- You need to be logged: (Login)
- You need to be logged as the owner of the (entity_id)
- (channel_id) has to be attached to the list of channels of (entity_id)

Request (application/json)

Response 200 (application/json)

Go to example

Delete Channel

DELETE /entities/{entity_id}/channels/{channel_id}

Delete an specific channel. Preconditions:

- You need to be logged: (Login)
- You need to be logged as the owner of the (entity_id)
- (channel_id) has to be attached to the list of channels of (entity_id)

Response 200 (application/json)

Go to example

Examples

Default

Login

[/login]

Login **POST /login**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
  "username": "john",
  "password": "john",
  "type": "user"
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
  "code": 200,
  "desc": "ok"
}
```

Go to specification

Logout

[/logout]

Logout **GET /logout**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
  "code": 200,
  "desc": "ok"
}
```

Go to specification

Users

[/users]

List All Users **GET /users**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
        [
            {
                "username": "john",
                "type": "user",
                "_id": "john"
            },
            {
                "username": "jammes",
                "type": "user",
                "_id": "jammes"
            }
        ]
}
```

Go to specification

Create User **POST /users**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "username": "john",
    "password": "john",
    "type": "user",
    "_id": "john"
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
        [
            {
                "username": "john",
                "password": "john",
                "type": "user",
                "_id": "john"
            }
        ]
}
```

Go to specification

User info

[/users/user]

User info **GET /users/user**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
        {
            "_id": "userID",
            "type": "user",
            "username": "username"
        }
    ]
}
```

Go to specification

User Collection

[/users/{user_id}]

Get user **GET /users/{user_id}**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
        {
            "_id": "userID",
            "type": "user",
            "username": "username"
        }
    ]
}
```

Go to specification

Delete User **DELETE /users/{user_id}**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
  "code": 200,
  "desc": "ok",
}
```

Go to specification

Update User

[/users/{user_id}/updatePassword]

Update User Password **PUT /users/{user_id}/updatePassword**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
    ]
}
```

Go to specification

Update User

[/users/{user_id}/rememberPassword]

Remember User Password **GET /users/{user_id}/rememberPassword**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result":
        [
            "You will receive an email with information for reset."
        ]
}
```

Go to specification

Verify User Password

[/users/{user_id}/rememberPassword/{code}]

Reset User Password **PUT /users/{user_id}/rememberPassword/{code}**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
     "password": "newPassword",
     "type": "user"
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
    ]
}
```

Go to specification

Entities

[/entities]

Get all entities **GET /entities**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
    [
        {


            "_id": "entityid",
            "channels": [ … ],
            "entitydescription": "entitydescription",
            "entityname": "entityname",
            "owner": "userid",
            "type": "entity"

        },
    ...
    ]
}
```

Go to specification

Create entity **POST /entities**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
  "entityname": "entityname",
  "entitydescription": "entitydescription",
  "type": "entity",
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
        [
            {
                "entityname": "entityname",
                "entitydescription": "entitydescription",
                "type": "entity",
                "owner": "userid",
                "channels": ["channelname"]
                }
            ]
}
```

Go to specification

Entities Collection

[/entities/{entity_id}]

Entity Info **GET /entities/{entity_id}**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
    [
        {

            "_id": "entityid",
            "channels": [ … ],
            "entitydescription": "entitydescription",
            "entityname": "entityname",
            "owner": "userid",
            "type": "entity"

        },
    ]
}
```

Go to specification

Update Entity **PUT /entities/{entity_id}**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
  "entityname": "entityname modified",
  "entitydescription": "entity description modified",
  "type": "entity"
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
    ]
}
```

Go to specification

Delete Entity **DELETE /entities/{entity_id}**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
    ]
}
```

Go to specification

Channels

[/entities/{entity_id}/channels]

Get channels **GET /entities/{entity_id}/channels**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
    [
        {
            "_id": "51e7be461fb1a1f179000002",
            "channelName": "channelname",
            "channeldesc": "channeldesc",
            "pubID": "4ca05d1f-ff02-481d-a1db-9d69c6bfff73",
            "subID": "d203158e-bac8-49ee-b2d3-1fd82729fa09"
        },
        ...
    ]
}
```

Go to specification

Create channel **POST /entities/{entity_id}/channels**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "channelName": "channelname",
    "channeldesc": "channeldesc",
    "type": "channel"
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
}
```

Go to specification

Channels Collection

[/entities/{entity_id}/channels/{channel_id}]

Channel Info **GET /entities/{entity_id}/channels/{channel_id}**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok"
    "result":
        [
            {
            "_id": "51e7be461fb1a1f179000002",
            "channelName": "channelname",
            "channeldesc": "channeldesc",
            "pubID": "4ca05d1f-ff02-481d-a1db-9d69c6bfff73",
            "subID": "d203158e-bac8-49ee-b2d3-1fd82729fa09",
            "puburl": "http://130.206.81.70:3000/publish/4ca05d1f-ff02-48
1d-a1db-9d69c6bfff73",
            "subscriptionsurl": "http://130.206.81.70:3000/subscribe/d203
158e-bac8-49ee-b2d3-1fd82729fa09",
            "subscriptions":
                [
                    {
                        _id: "53bfe41faf5e6e0200000016",
                        desc: "demo-sub",
                        id: "demo-sub",
                        ip: "http://130.206.81.70:3000",
                        subkey: "Delivery Item 1-10925842-queu"
                    },
                    ...
                ]
            }
        ]
}
```

Go to specification

Update Channel **PUT /entities/{entity_id}/channels/{channel_id}**

Request (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "channelName": "new channelname",
    "channeldesc": "new channeldesc",
    "type": "channel"
}
```

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
    ]
}
```

Go to specification

Delete Channel **DELETE /entities/{entity_id}/channels/{channel_id}**

Response 200 (application/json)

Headers

```
Content-Type: application/json
```

Body

```
{
    "code": 200,
    "desc": "ok",
    "result": [
    ]
}
```

Go to specification