

AEON Cloud Messaging GE

DATE: 30 June 2015

[View in Apiary](#) 

Editors

- José Gato Luis
- Javier García Hernández
- Elisa Herrmann

Copyright

Due the interoperability needs of these architectures, AEON has been fully designed considering Free Libre Open Source Software technologies. The project has been designed and implemented by the "Transport and Trade Logistics Sector in the Research & Innovation department in ATOS Spain". AEON Platform is released as Open Source.

The Cloud Messaging GE specification is licensed under the FIWARE Open specification license (http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Open_Specification_Legal_Notice_%28essential_patents_license%29)

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 management and pub/sub functionality.

Status of this document

The AEON platform is under constant development. We are working to improve the capabilities offered by it.

Table of Contents

AEON Cloud Messaging GE	1
Editors	1
Copyright	1
Abstract	1
Status of this document	1
Table of Contents	2
API Summary	4
Acknowledgements	4
User operations	4
Entities operations	5
Channels operations	5
API Specification	6
Default	6
Login	6
Login	6
Logout	6
Logout	6
Users	7
List All Users	7
Create User	7
User info	8
User info	8
User Collection	9
Get user	9
Delete User	9
Update User	10
Update User Password	10
Update User	10
Remember User Password	10
Verify User Password	11
Reset User Password	11
Entities	12
Get all entities	12
Create entity	12
Entities Collection	12
Entity Info	12
Update Entity	13
Delete Entity	13
Channels	14
Get channels	14
Create channel	14
Channels Collection	14
Channel Info	15
Update Channel	15
Delete Channel	15
Examples	16
Default	16
Login	16
Login	16
Logout	17
Logout	17
Users	17
List All Users	17
Create User	18
User info	19
User info	19
User Collection	19
Get user	19
Delete User	20
Update User	20
Update User Password	20
Update User	21
Remember User Password	21
Verify User Password	21
Reset User Password	22
Entities	22
Get all entities	22
Create entity	23
Entities Collection	24
Entity Info	24

Update Entity	25
Delete Entity	25
Channels	26
Get channels	26
Create channel	26
Channels Collection	27
Channel Info	27
Update Channel	28
Delete Channel	29

References	30
------------	----

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}]

Acknowledgements

Thanks to all the AEON team (Jose, Elisa, Gonzalo, Germán and myself) that makes this possible.

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/login/login\)](http://docs.aeon.apiary.io/#reference/default/login/login)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/logout/logout\)](http://docs.aeon.apiary.io/#reference/default/logout/logout)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/users/list-all-users\)](http://docs.aeon.apiary.io/#reference/default/users/list-all-users)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/users/create-user\)](http://docs.aeon.apiary.io/#reference/default/users/create-user)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/user-info/user-info\)](http://docs.aeon.apiary.io/#reference/default/user-info/user-info)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/user-collection/get-user\)](http://docs.aeon.apiary.io/#reference/default/user-collection/get-user)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/user-collection/delete-user\)](http://docs.aeon.apiary.io/#reference/default/user-collection/delete-user)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/update-user/update-user-password\)](http://docs.aeon.apiary.io/#reference/default/update-user/update-user-password)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/update-user/remember-user-password\)](http://docs.aeon.apiary.io/#reference/default/update-user/remember-user-password)

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 - Verification 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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/verify-user-password/reset-user-password\)](http://docs.aeon.apiary.io/#reference/default/verify-user-password/reset-user-password)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/entities/get-all-entities\)](http://docs.aeon.apiary.io/#reference/default/entities/get-all-entities)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/entities/create-entity\)](http://docs.aeon.apiary.io/#reference/default/entities/create-entity)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/entities-collection/entity-info\)](http://docs.aeon.apiary.io/#reference/default/entities-collection/entity-info)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/entities-collection/update-entity\)](http://docs.aeon.apiary.io/#reference/default/entities-collection/update-entity)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/entities-collection/delete-entity\)](http://docs.aeon.apiary.io/#reference/default/entities-collection/delete-entity)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/channels/get-channels\)](http://docs.aeon.apiary.io/#reference/default/channels/get-channels)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/channels/create-channel\)](http://docs.aeon.apiary.io/#reference/default/channels/create-channel)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/channels-collection/channel-info\)](http://docs.aeon.apiary.io/#reference/default/channels-collection/channel-info)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/channels-collection/update-channel\)](http://docs.aeon.apiary.io/#reference/default/channels-collection/update-channel)

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](#)

[View in Apiary \(http://docs.aeon.apiary.io/#reference/default/channels-collection/delete-channel\)](http://docs.aeon.apiary.io/#reference/default/channels-collection/delete-channel)

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"
}
```


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-ff02-481d-a1db-9d69c6bfff73",
  "subscriptionsurl": "http://130.206.81.70:3000/subscribe/d203158e-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](#)

References

- http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FI-WARE_Open_Specification_Legal_Notice_%28essential_patents_license%29