RESTful Web API

Specification

- IoT anyware –



Team number 1

June 2015

Revision History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 0.1 | 07,06, 2015 | Initial draft  -. Account Manager and Log Viewer interface are defined |
| 1.0 | 18/06, 2015 | Refine and release v1.0 |
|  |  |  |
|  |  |  |
|  |  |  |

TABLE OF CONTENTS

[Chapter 1. Common 4](#_Toc422422101)

[1.1 Request format 4](#_Toc422422102)

[1.2 Response format 4](#_Toc422422103)

[Chapter 2. Account Manager 5](#_Toc422422104)

[2.1 Register new user account 5](#_Toc422422105)

[2.2 Confirm user account 6](#_Toc422422106)

[2.3 Request new session for user 7](#_Toc422422107)

[2.4 Request new session for node 8](#_Toc422422108)

[2.5 Refresh session 10](#_Toc422422109)

[2.6 Get registered nodes 11](#_Toc422422110)

[2.7 Get profile detail 13](#_Toc422422111)

[2.8 Register a new SA Node 15](#_Toc422422112)

[2.9 Unregister SA Node 16](#_Toc422422113)

[2.10 Grant SA node access permission 17](#_Toc422422114)

[2.11 Transfer ownership of SA Node 19](#_Toc422422115)

[2.12 Get user configurations 20](#_Toc422422116)

[2.13 Update user configurations 21](#_Toc422422117)

[Chapter 3. Log Viewer 23](#_Toc422422118)

[3.1 Get log history 23](#_Toc422422119)

# Chapter 1. Common

## 1.1 Request format

* **RESTful** web **APIs** are introduced to support scalable http request. **HTTPS** is used for [secure](http://en.wikipedia.org/wiki/Network_security) communication.
* GET : retrieve data
* POST : create new data
* PUT : update data
* DELETE : remove data

|  |  |  |
| --- | --- | --- |
| Country | Phase | URL |
| US | Development | http://54.166.26.101:8000 |

## 1.2 Response format

* Web server response for the request of User App and SA node is defined as below format. The response code is following [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) [stand](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes)ards.

|  |  |  |
| --- | --- | --- |
| **Body** | | |
| **Key** | **Description** | |
| **statusCode** | Response result. Detail code is like below. | |
| **Value** | Meaning |
| 200 | Success. |
| 400 | Bad Request.   * Omit mandatory parameter. * Invalid value of parameter. |
| 401 | Authorization Failed. |
| **result** | Corresponding output. If status code is not success, then it represents specific error reason. Each API includes detailed value. | |

# 

# Chapter 2. Account Manager

## 2.1 Register new user account

* User App requests new user account to the web server. The User App adds unique client-id to the header of request message for security purpose. Account manager saves all user information temporary and issues new token. If this procedure is completed, the server(account manager) responses code “200” to User App and sends authentication email to user. The user should follow the authentication procedure to complete the registration via linked URL in the email.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /account/registerNewUser | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| email | Mandatory | User id. The system will send email which including url to proceed authentication. This address is also used for push notification purpose. |
| mobileNumber | Optional | It is used when push notification is needed. |
| nickName | Mandatory | Display name for UI. |
| profileImage | Optional | Display image for UI. |
| password | Mandatory | Sha256 encrypted password. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “email”: “[simpson.homer@gmail.com](mailto:simpson@gmail.com)”,  “nickname” : “homer”,  “password”: “5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8”  } | | |
|  | | |

**Response**

|  |
| --- |
| **Example** |
| {  “statusCode” : “200”,  “result” : {}  } |

## 2.2 Confirm user account

* User requests the authentication of the account via linked URL in the email sent by the server. If the user doesn’t process authentication in 10 minutes the account manager disposes the account request and the temporary token is expired.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /account/emailConfirm | |
| **Method** | GET | |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| token | Mandatory | System generated token. |
| **Example** | | |
| /account/emailConfirm?token=f63be4afb5d758205005b13d49af598f4ba2b242 | | |

**Response**

|  |
| --- |
| **Example** |
| The HTTP redirection to corresponding user page. |

## 2.3 Request new session for user

* User App requests a **session id** for the “event bus” access. The email/password of user account should be included in the request message.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /session/createUser | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| email | Mandatory | User id. |
| password | Optional | Sha256 encrypted password. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “statusCode” : “200”,  “result” : {  “email”: “[simpson.homer@gmail.com](mailto:simpson@gmail.com)”,  “password” : “5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8”  }  } | | |

**Response**

|  |  |
| --- | --- |
| **result** | |
| **Key** | **Description** |
| session | Created session. It has an expired time. |
| refreshToken | Token which is used when session is expired. This token is never expired. |
| expires | Time limit of session in UTC timestamp manner. |
| **Example** | |
| {  “statusCode” : “200”,  “result” : {  “session”: “44155ac6-9983-48c7-bcc0-e21623424898”,  “refreshToken”: “fa3aa284-c94d-4b8c-9cec-3613b2a4a8f8”,  “expires”: 1433802701  }  } | |

## 2.4 Request new session for node

* SA Node requests a **session id** for the “event bus” access. The **nodeid** should be included in the request message. If the nodeid is not registered to any user account manager, the server rejects this request.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /session/createNode | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| nodeId | Mandatory | SA Node identification. (eg. serial number) |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “nodeId” : “14:10:9f:dc:23:a3”  } | | |

**Response**

|  |  |
| --- | --- |
| **result** | |
| **Key** | **Description** |
| session | Created session. It has an expire time. |
| refreshToken | Token which used when session is expired. This token never expires. |
| expires | Time limit of session in UTC timestamp manner. |
| **Example** | |
| {  “statusCode”: 200,  “result” : {  “session”: “44155ac6-9983-48c7-bcc0-e21623424898”,  “refreshToken”: “fa3aa284-c94d-4b8c-9cec-3613b2a4a8f8”,  “expires”: 1433802701  }  } | |

## 2.5 Refresh session

* After session expired, User App requests the reissuing of the session with **refreshToken** and former session id.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /session/refresh | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| session | Mandatory | Created session. |
| refreshToken | Mandatory | Token which corresponding session. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “session”: “44155ac6-9983-48c7-bcc0-e21623424898”,  “refreshToken”: “fa3aa284-c94d-4b8c-9cec-3613b2a4a8f8”  } | | |

**Response**

|  |  |
| --- | --- |
| **result** | |
| **Key** | **Description** |
| session | Created session. It has an expire time. |
| refreshToken | Token which used when session is expired. This token never expires. |
| expires | Time limit of session in UTC timestamp manner. |
| **Example** | |
| {  “statusCode”: 200,  “result” : {  “session”: “0473602c-473a-48e5-b6be-b41fc14ffd6c”,  “refreshToken”: “fa3aa284-c94d-4b8c-9cec-3613b2a4a8f8”,  “expires”: 1433811374  }  } | |

## 2.6 Get registered nodes

* User App requests an SA Nodes list to show it to user. The response includes the sensors/actuators list and the property of each SA node. The response does not include the status of sensors/actuators equipped on each SA node.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/getNodeList | |
| **Method** | GET | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| session | Mandatory | Created session |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  /user/getNodeList?session=0473602c-473a-48e5-b6be-b41fc14ffd6c | | |

**Response**

|  |
| --- |
| **result** |
| List of SA node which include installed sensors and related profile information. |
| **Example** |
| {  “statusCode”: 200,  “result”: [  {  “id” :  “owner” : true,  “profiles”: [  {“name”: “indoor\_light\_sensor”, “profile”: “com.lge.sensor.light.led”},  {“name”: “outdoor\_light\_sensor”, “profile”: “com.lge.sensor.light.led”},  {“name”: “temp\_sensor”, “profile”: “com.lge.sensor.temp”},  {“name”: “humidity\_sensor”, “profile”: “com.lge.sensor.humidity”},  {“name”: “door\_actuator”, “profile”: “com.lge.actuator.door”},  {“name”: “door\_sensor”, “profile”: “com.lge.sensor.door”},  {“name”: “alarm\_sensor”, “profile”: “com.lge.sensor.alarm”},  {“name”: “presence\_sensor”, “profile”: “com.lge.sensor.presence”}  ]  },  “SA2” : {  “owner” : false,  “profiles”: [  {“name”: “indoor\_light\_sensor”, “profile”: “com.lge.sensor.light.led”},  {“name”: “outdoor\_light\_sensor”, “profile”: “com.lge.sensor.light.led”},  {“name”: “temp\_sensor”, “profile”: “com.lge.sensor.temp”},  {“name”: “humidity\_sensor”, “profile”: “com.lge.sensor.humidity”},  {“name”: “door\_actuator”, “profile”: “com.lge.actuator.door”},  {“name”: “door\_sensor”, “profile”: “com.lge.sensor.door”},  {“name”: “alarm\_sensor”, “profile”: “com.lge.sensor.alarm”},  {“name”: “presence\_sensor”, “profile”: “com.lge.sensor.presence”}  ]  }  ]  } |

## 2.7 Get profile detail

* User App requests the details of profile. The profile includes attributes of the SA node.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /profile/getDetail | |
| **Method** | GET | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| profile | Mandatory | Name of profile. |
| session | Mandatory | Created session. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  /profile/getDetail?target=com.lge.sensor.door&session=0473602c-473a-48e5-b6be-b41fc14ffd6c | | |

**Response**

|  |
| --- |
| **result** |
| Detailed profile information w/ Json format. |
| **Example** |
| {  “statusCode”: 200,  “result”: {  "id": "com.genetic.door",  "value": {  "access": "rw",  "type": "list",  "valueList": [  "open",  "close"  ],  "description": "The Door"  }  }  } |

## 2.8 Register a new SA Node

* User App requests the SA node registration with **nodeId** of the new SA node. Account manager sends the confirmation of the temporal registration to the User App. To complete the registration, User should manipulate the node to establish connection to the Account manager. If nodeId is already registered, temporally registered, or invalid, Account manager sends the error code. If the Account Manager doesn’t receive the connection request from the node for a specified amount of time, the Account Manager revokes the temporal registration.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/registerNode | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| nodeId | Mandatory | SA Node identification. (eg. serial number) |
| session | Mandatory | Created session. |
| nickName | Optional | Display name of the SA Node. |
| virtual | Optional | Add this parameter if want to add virtual SA node. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “nodeId”: “14:10:9f:dc:23:a3”,  “session”: “0473602c-473a-48e5-b6be-b41fc14ffd6c”.  “nickName”: “My SANode”  } | | |

**Response**

|  |
| --- |
| **result** |
| Success or corresponding error output |
| **Example** |
| {  “statusCode”: 200,  “result”: {}  } |

## 2.9 Unregister SA Node

* User App requests the SA node unregister with **nodeId** of the node. The AccountManager deletes the nodeId from the account. If the account’s access permission of the node is “owner”, **nodeId** is deleted from all other account, and the node deletion event is published to all relevant accounts.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/unregisterNode | |
| **Method** | DELETE | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| nodeId | Mandatory | SA Node identification. (eg. serial number) |
| session | Mandatory | Created session. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “nodeId”: “14:10:9f:dc:23:a3”,  “session”: “0473602c-473a-48e5-b6be-b41fc14ffd6c”  } | | |

**Response**

|  |
| --- |
| **result** |
| Success or corresponding error output |
| **Example** |
| {  “statusCode”: 200,  “result”: {}  } |

## 2.10 Grant SA node access permission

* User App requests to grant SA node access permission to other user with **nodeId** and **targetUser**. The AccountManager adds nodeId to targetUser with permission “use”. If the requester is not the owner of the node, an error code is sent.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/shareNode | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| targetUser | Mandatory | Email address of target user. |
| session | Mandatory | Created session. |
| nodeId | Mandatory | SA Node identification. (eg. serial number) |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “targetUser”: “[simpson.marge@gmail.com](mailto:simpson.marge@gmail.com)”,  “session”: “0473602c-473a-48e5-b6be-b41fc14ffd6c”,  “nodeId”: “14:10:9f:dc:23:a3”  } | | |

**Response**

|  |
| --- |
| **result** |
| Success or corresponding error output |
| **Example** |
| {  “statusCode”: 200,  “result”: {}  } |

## 2.11 Transfer ownership of SA Node

* User App requests to transfer the ownership of SA node to other user with **nodeId** and **targetUser**. The AccountManager modifies **targetUser**’s access permission of **nodeId** to “own”, and requester’s to “use”. If the requester is not the owner of the node, an error code is sent.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/transferOwner | |
| **Method** | POST | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| targetUser | Mandatory | Email address of target user. |
| session | Mandatory | Created session. |
| nodeId | Mandatory | SA Node identification. (eg. serial number) |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “targetUser”: “[simpson.marge@gmail.com](mailto:simpson.marge@gmail.com)”,  “session”: “0473602c-473a-48e5-b6be-b41fc14ffd6c”,  “nodeId”: “14:10:9f:dc:23:a3”  } | | |

**Response**

|  |
| --- |
| **result** |
| Success or corresponding error output |
| **Example** |
| {  “statusCode”: 200,  “result”: {}  } |

## 2.12 Get user configurations

* User App requests the User configuration information. The information includes the log time window.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/getConfiguration | |
| **Method** | GET | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| session | Mandatory | Created session. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  /user/getConfiguration?session=0473602c-473a-48e5-b6be-b41fc14ffd6c | | |

**Response**

|  |  |
| --- | --- |
| **result** | |
| **Key** | **Description** |
| loggingHour | How long the system maintain user log data. |
| **Example** | |
| {  “statusCode”: 200,  “result”: {  “loggingHour” : 72  }  } | |

## 2.13 Update user configurations

* User App requests the User configuration update. The configuration includes the sensor value and command history log time window.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /user/setConfiguration | |
| **Method** | PUT | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| session | Mandatory | Created session. |
| loggingHour | Mandatory | How long the system maintain user log data. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  {  “session”: “0473602c-473a-48e5-b6be-b41fc14ffd6c”,  “loggingHour”: 24  } | | |

**Response**

|  |
| --- |
| **result** |
| Success or corresponding error output |
| **Example** |
| {  “statusCode”: 200,  “result”: {}  } |

# Chapter 3. Log Viewer

## 3.1 Get log history

* User App requests the sensor value and command history log. The log is stored for a specified amount of time. Logs are presented in descending order of time.

**Request**

|  |  |  |
| --- | --- | --- |
| **URI** | /log/getHistory | |
| **Method** | GET | |
| **Header** | | |
| **Key** | **Mandatory** | **Description** |
| content-type | Mandatory | Media type of the entity-body. Fixed to “application/json” |
| x-client-id | Mandatory | Application unique identify string. |
| **Body** | | |
| **Key** | **Mandatory** | **Description** |
| session | Mandatory | Created session. |
| **Example** | | |
| header  {  “content-type” : “application/json”,  “x-client-id” : “75f9e675-9db4-4d02-b523-37521ef656ea”  }  body  /log/getHistory?session=0473602c-473a-48e5-b6be-b41fc14ffd6c | | |

**Response**

|  |  |
| --- | --- |
| **result** | |
| List of logs which include user control request and status change event from registered SA Node(s). | |
| **Key** | **Description** |
| topic | Event name. It could be control request from User or status change event from SA Node. |
| data | Corresponding event data. |
| timestamp | When the event is triggered. |
| from | Who trigger the event? It could be an User or a SA Node. |
| **Example** | |
| {  "result": [  {  "msg": {  "door": "open",  "name": "SA1",  "publisher": "simpon.homer@gmail.com"  },  "msg\_type": "control",  "node": "SA1",  "timestamp": "Sun, 14 Jun 2015 03:30:58 GMT"  },  {  "msg": {  "door": "close"  },  "msg\_type": "status",  "node": "SA1",  "timestamp": "Sun, 14 Jun 2015 03:29:13 GMT"  }  ],  "statusCode": 200  } | |