INTERNALS OF APPLICATION SERVER (Spring 2015)

USER MANUAL

SUBMITTED BY:

GROUP: 3

G. UMA MAHESWARA RAO (201250819)

HANUMANTHA REDDY S (201250822)

CHERUKUPALLI PRABHAKAR (201250810)

S.D. NAGESWARA PAVAN KUMAR (201150836)

THAKKAR HEMAL NARENDRA (201350890)

AVINASH CHANDWANI (201350882)

SHELKE NAMITA RAVINDRA (201405633)

SINGARAJU SUGUNA MALLIKA 201450866)

J VAMSHI VIJAY KRISHNA(201450865)

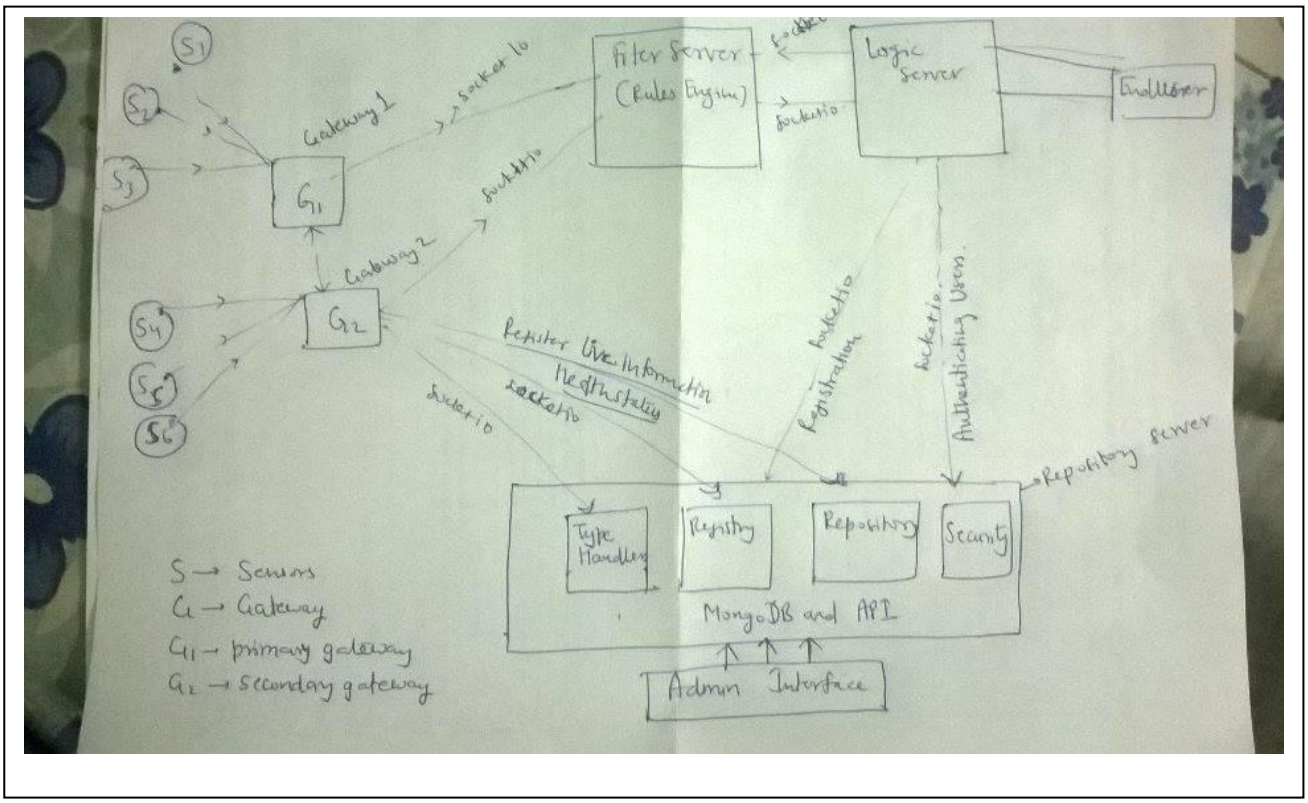
Table of Contents

1. Overview

The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable it to achieve greater value and service by exchanging data with the manufacturer, operator and/or other connected devices.The objective of the IOT (Internet of things) platform is to provide an integrated, secure and easy development platform for the IOT applications. This platform is intended to host multiple types of IoT applications.

This guide provides a brief overview of the platform functional design along with step by step procedure to let an administrator install and configure the development platform. This also includes a description of API exposed by various components of the platform that could be utilized by users (administrator and developers) of this platform.

2. Design Reference



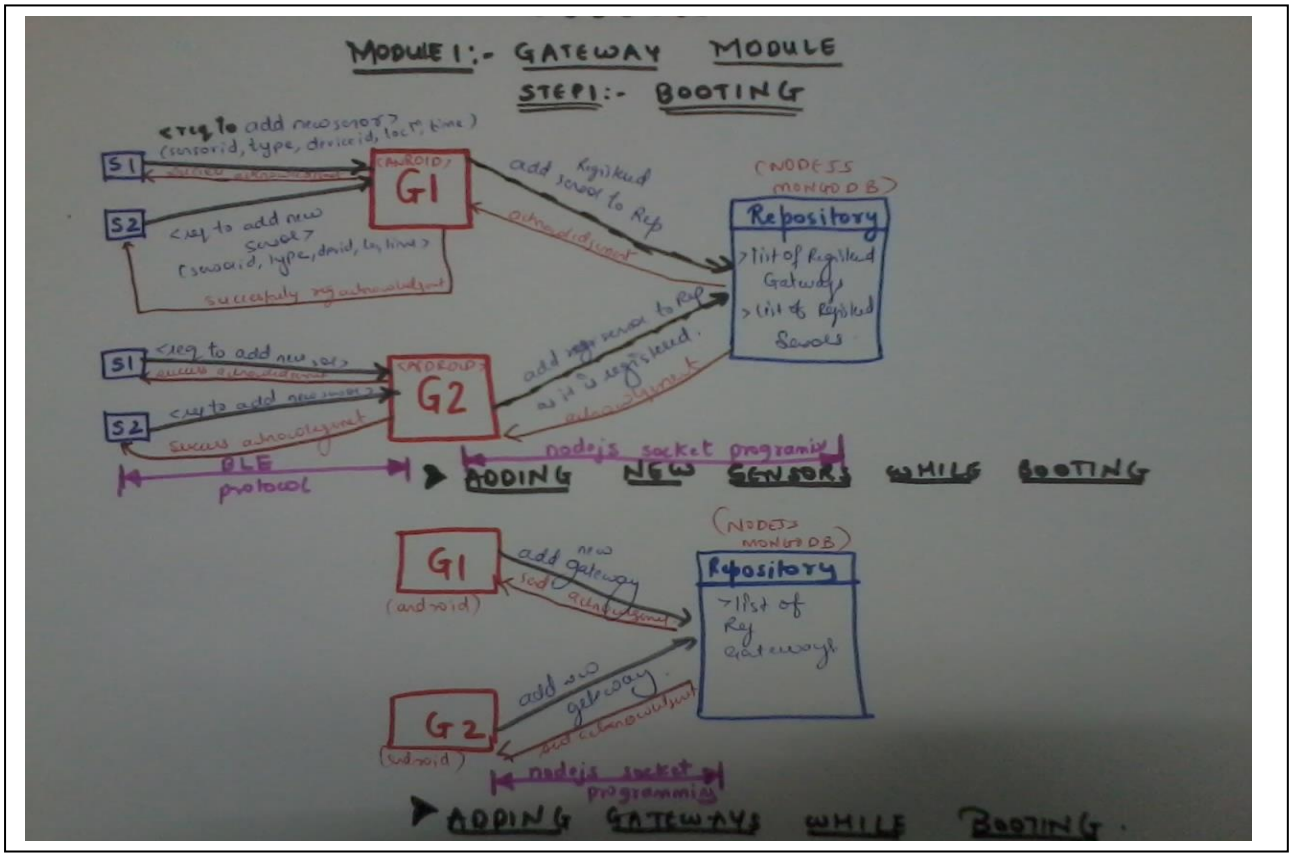
Overall modular layout of IOT Platform

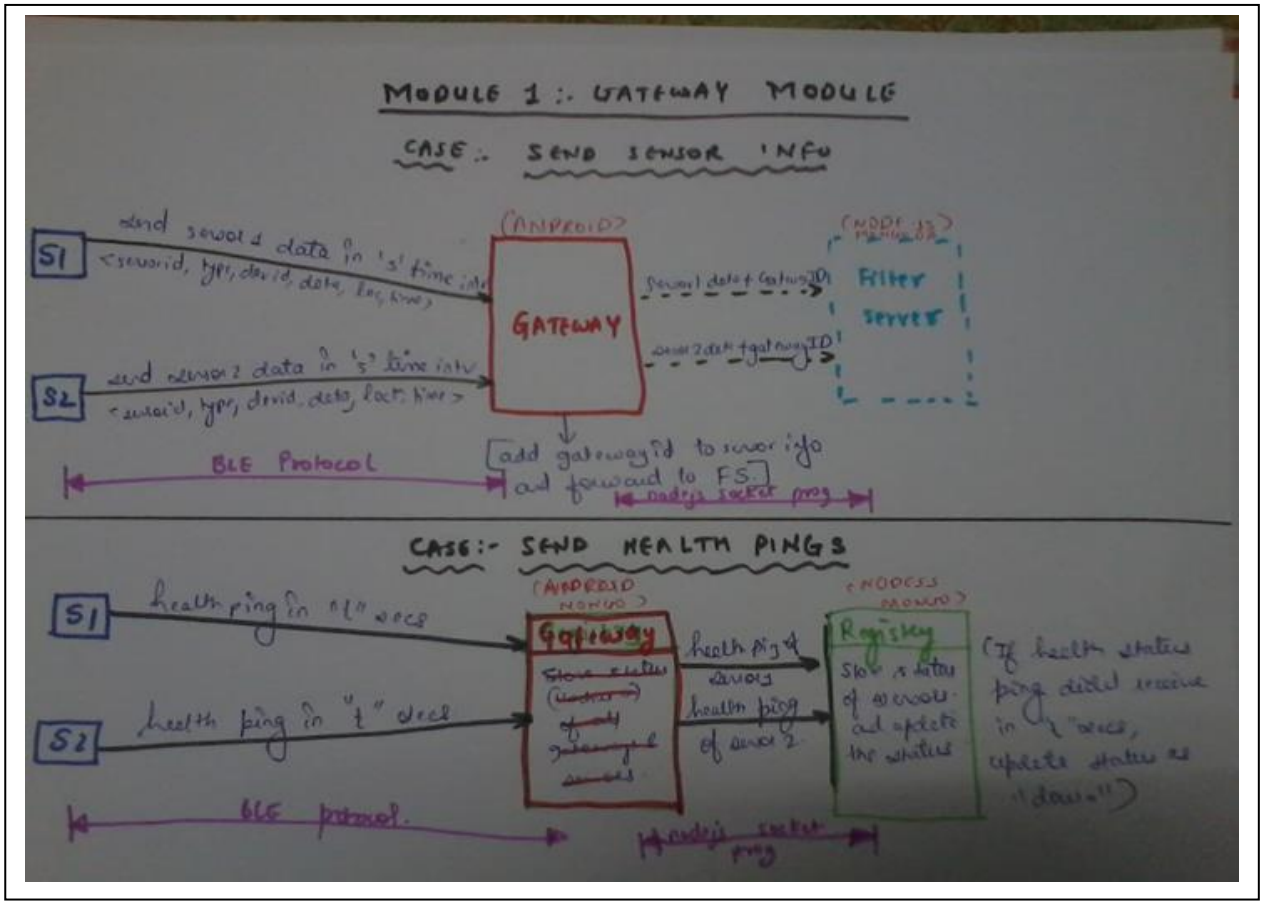
2.1 **Gateway Module**

Setup of this module comprise of start of the whole system, this module is the source of information for the whole system to work. This starts with registering the sensors and gateways. This also includes authentication and persisting their extra information into the repository module. This module also involves inter- communication between these modules to let this work in an integrated manner. Two types of information travel between these modules, one is health ping status information of sensor and gateway and the second is the actual recorded data from the sensor.

The gateway acts as intermediate between sensor and filter server & sensor and registration. Gateway is like a router, which communicates with other and there is a

well defined network topology used so that the failures are well taken care of. The repository also has an interface for administrator to let add new sensors and gateways, and also attach type handler for each type of sensor, which adapts to the required format of the data. The below figures explain the echo system of gateway and the interfaces and the some of the features of the gateway such as getting sensor information and health pings from the sensor.

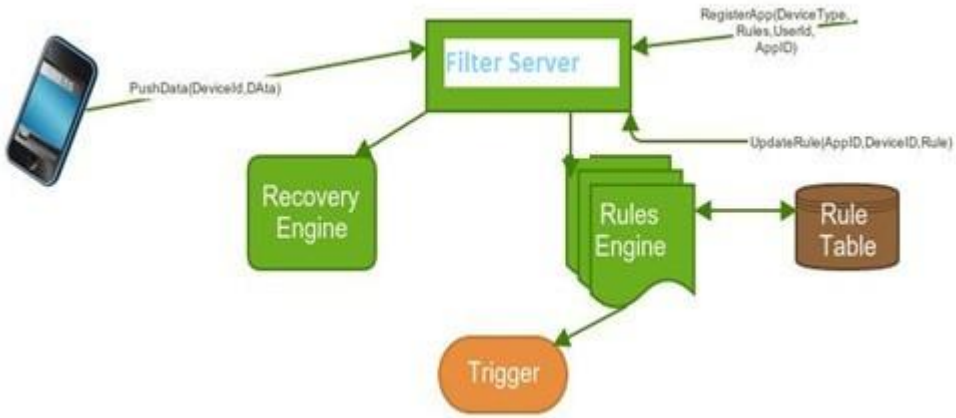


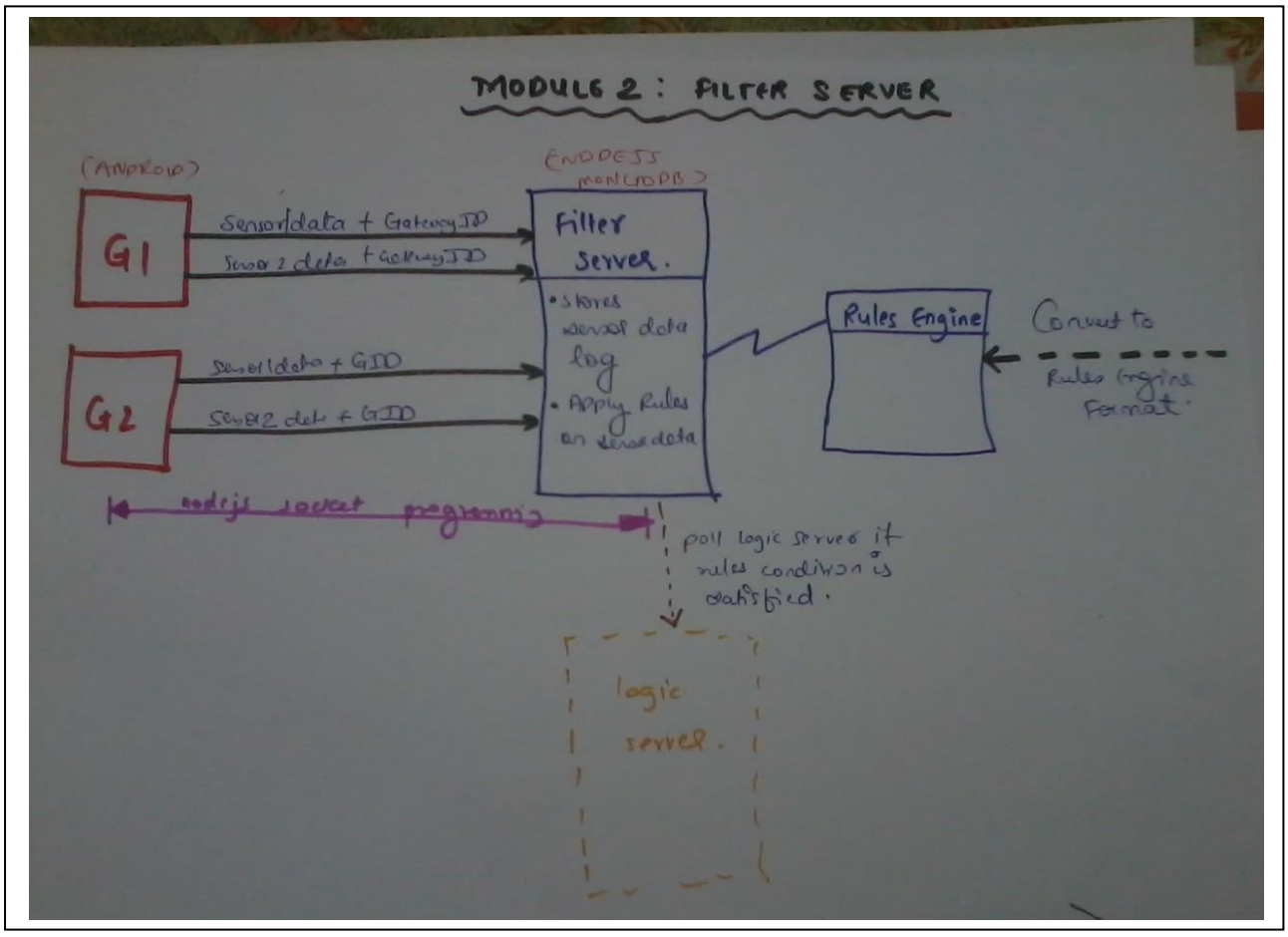


2.2. FilterServer Module

Filter Server will take data from sensory devices and perform actions according the rules given by the user.

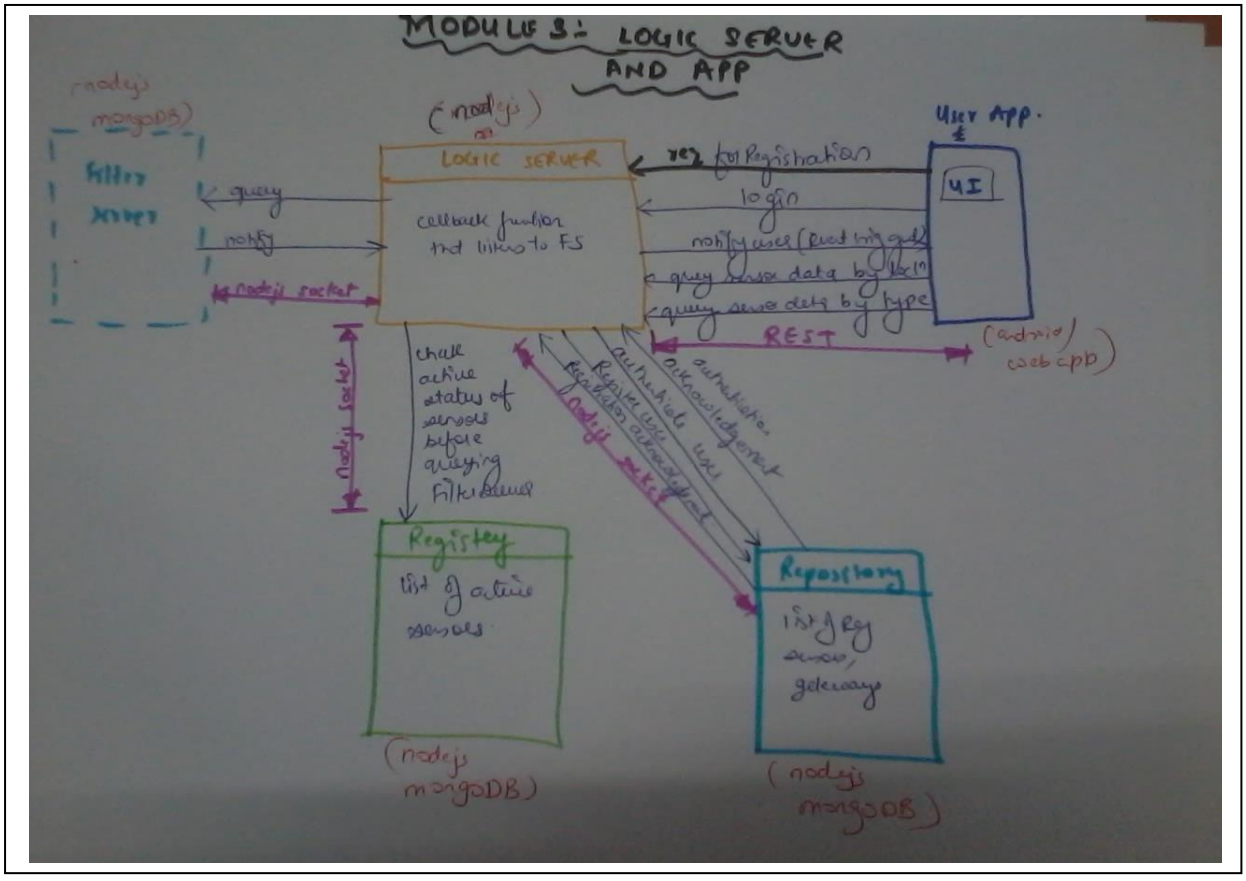
- Register the application and pair user with corresponding devices.
- Take rules from user and convert rules into the action using rules engine.
- Take data from the devices and if any rule applies then take appropriate action
- Before taking action it will check for authorization.





2.3. LogicServer Module

It will have “RegistrationAPI” which will register the user in the Registry subModule. It will have “LoginAPI” which checks for user authentication (via Registry) and make service available to the authenticated user. It will contain the logic of the task that has to be done if the sensor data has crossed the threshold value . Eg Notifying the user etc.It will contain the “QueryAPI” which will handle the “query request” for the sensor data by its location or type or range.



3. Getting Started –

3.1 Installation and Configuration (Administrator Tasks)

3.1.1. Steps to set up Gateway

A network node equipped for interfacing with another network that uses different protocols.
A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversion. Gateways, also called protocol converters, can operate at any network layer.

Technology needed

- The computer or the program can be configured to perform the tasks of gateway. So we can take the Android phone as

"Gateway" . The android should have Bluetooth facility for communicating with (via BLE)

Installation steps

- Install the android program (.apk) to android phone that will act as "Gateway".
- Turn the bluetooth On (if the Gateway program does not turn it on)

How to start and stop it

- Start the Gateway program as any other android app i.e. by clicking on it.
- To stop it , the gateway app has to be closed.

3.1.2. Steps to set up Filter Server

Filter Server is the container for rules engine which is implemented using nools rules engine (built on nodejs technology stack)

Technology needed

- Nodejs module should be installed on this server for rules engine to work.
- Mongo DB need to be installed to store the live sensor data.

Installation steps

- **C**opy paste the nodejs scripts.

How to start and stop it

- Start the filter server by running the nodejs script file , by typing the following line in command terminal:
  "nodejs filterServer.js"
- To stop the Filter Server , type following line in command terminnal:
  "killall nodejs"

3.1.3. Steps to set up Repository

This module provides the information of registered sensors , gateways and registered front-end users

Technology needed

- Nodejs needed to expose it on particular port so that others can communicatewith it to read/insert/update mongoDB containing information of registered sensors ,gateways and registered front-end users.
- Mongo DB need to be installed to store the info registered sensors ,gateways and registered front-end users.

.Installation steps

- Copy the nodejs api code
- Make the mongoDB schema for storing info of :
  - Registered Sensors
  - Registered Gateways
  - Registered Users

Note: Repository,Registry,Security module installed on same machine but different ports.

How to start and stop it

- To start the Repository server , run the nodejs script by typing the following line in command terminal:
  "nodejs repositoty.js"

- As repository , registry and security run on same server , to stop only repository (without impacting other two) the process id of repository has to be found out and the process has to be killed using kill command, shown below:
  ps aux | grep repository.js(Find the process ID)
  kill -9 PROCESS_ID(To kill/ stop repository)

3.1.4. Steps to set up Registry

This module contains the live status of all sensors and gateways i.e active status

Technology needed

- Nodejs needed to expose it on particular port so that others can communicate with it to read/update mongoDB containing information of active status of sensors and gateways.
- Mongo DB need to be installed to store the info of active sensors and gateways.

Installation steps

- Copy the nodejs scripts api
- Make the mongoDB schema for storing info of :
  - Active status of Sensors and gateways.

Note: Repository,Registry,Security module installed on same machine but different ports.

How to start and stop it

- To start the registry server , run the nodejs script by typing the following line in command terminal:
  "nodejs registry.js"

- As repository , registry and security run on same server , to stop only registry (without impacting other two) the process id of registry has to be found out and the process has to be killed using kill command, shown below:
  ps aux | grep registry.js(Find the process ID)
  kill -9 PROCESS_ID(To kill/ stop registry)

3.1.5. Steps to set up LogicServer

The Logic server will contain the logic of the task that has to be done when the filter server polls it .

Technology needed

- Nodejs api need to be exposed to filter server to poll it wen the certain task has to be performed when the rules are met.

Installation steps

- Copy the nodejs scripts api

How to start and stop it

- Start the filter server by running the nodejs script file , by typing the following line in command terminal:
  "nodejs logicServer.js"
- To stop the Filter Server , type following line in command terminnal:
  "killall nodejs"

3.1.6 Steps to set up Security Module

This module helps securing system by holding credentials for the registered end users.

Technology needed

- Nodejs needed.
- Mongo DB need to be installed.

Installation steps

- Copy the nodejs scripts api

How to start and stop it

- To start the Security server , run the nodejs script by typing the following line in command terminal:
  "nodejs security.js"

- As repository , registry and security run on same server , to stop only security module (without impacting other two) the process id of security has to be found out and the process has to be killed using kill command, shown below:

  ps aux | grep security.js(Find the process ID)
  kill -9 PROCESS_ID(To kill/ stop security)

3.2 API reference

The API referenced can be used by users of this platform to interact with platform components.

3.2.1 Repository

AddGateway (<location>) : The addGateway method is used to let a new Gateway entry be added to the repository of the platform. This method could be used to a intial setup of Gateways during initial setup by admin via command line utility. Also whenever a gateway goes down, during the subsequent boot-up phase this method is called by the gateway to validate its registration with platform and update its current boot time. The location of the gateway would be sent as input param for this method. The return value sent by Repositroy serves as acknowledgement and would be the gateway id that the gateway stores in process.

params : Location co-ordinates
return value : gateway_id

AddSensor (sensorType, gatewayId, <location>, typeHandler) : The addSensor is used to let a new sensor entry be added to the repository of the platform. This method could be used during initial setup by admin via command line utility. Also whenever a sensor goes down, during the subsequent boot-up phase this method is called to validate the registration of the sensor with platform and update its current boot time. The type of sensor, id of the gateway sensor is assigned , location of sensor and the type handler for the protocol used to communicate with the sensor are the input parameters for this method. The return value sent by Repositroy serves as acknowledgement and would be the sensor id that the sensor stores in process.

params: type_sensor, gateway_id. Location co-ordinates, typeHandler
return value : sensor_id

Registry

GetActiveSensorsByLocation (<location>) : This method can be used to retrieve a list of active sensors with a given location. The location co-ordinates are passed as input parameters. The return value includes a list of sensor object types providing the details of the sensors active within a given location.

params: Location co-ordinates
return value : <list_active_sensors>

GetActiveSensorsByTypeLocation(<type>, <location>) : This method can be used to retrieve a list of active sensors with a given location and given type. The location co-ordinates and types of sensor are passed as input parameters. The return value includes a list of sensor object types providing the details of the sensors active within a given location and type

params: Location co-ordinates, type_sensor
return value : <list_active_sensors>

GetActiveSensorByGatway(<gateway_id>) : This method can be used to retrieve a list of active sensors associated with a given Gateways. The id of the gateway is passed as input parameters. The return value includes a list of sensor object types providing the details of the sensors active within a given location. The primary utility of this API is during its boot-up phase of a Gateway to retrieve the list of active sensors associated with it before fail-over.

params: gateway_id
return value : <list_active_sensors>

GetActiveGatewaysByLocation (<location>): This method can be used to retrieve a list of active Gateways with a given location. The location co-ordinates are passed as input parameters. The return value includes a list of gateway object types providing the details of the gateways active within a given location.

params: Location co-ordinates
return value : <list_gateway_servers>

Logic Server

GetAllApplications() : This method can be used by the developer to provision a list of registered applications for mobile users.

params: none
return value : <list_applications>

GetApplication(app_title) : This method could be used by developer to retrieve a reference to specific application and be able to further update the properties of that application.

params: application_title
return value : <ref_application>

RegisterApplication(app_title) : This method can be used by a developer to provision a means to create a new application and register the same with the help of logic server. The return value would be application id

params: application_title
return value : <app_id>

GetSensorTypes() : This method can be used by a developer to provision a means to let end user retrieve a list of Types of sensor devices available for setting up an application.

params: none
return value : <list_sensor_types>

GetSensorLocations() : This method can be used by a developer to provision a means to let the user retrieve a list of locations with installed sensors . The return value would be a list of location object types providing the details of the locations active with active sensors within.

params: none
return value : <list_locations>

QuerySensorDataByTypeLocation() : This method can be used by developer to provision a means to query sensor data pertaining to a given sensor type and location. The return value would be list of data records collated by querying the registry for corresponding sensor type and location

params: Location co-ordinates, type_sensor
return value : <list_gateway_servers>

GetRuleSchema() : This method can be used by a developer to provision a means to load the Rule template for updates to the rule configuration for a given application. The return value would involve a response with json object providing a structure with default template of the rule configuration

params: none
return value : <rules_schema_config>

UpdateSchema(app_id, <rule_schema_config>) : This method would facilitate developer to provision a means to save the values from rule schema to the database and link the new instance of the rule configuration to the corresponding application. The response would include a acknowledgement token to confirm the completion of update operation.

params: application_id, <rule_schema_config>
return value : ACK

UpdateConfig() : This facilitates a admin user to configure the platform parameters like device types, device count and interaction mode and control the heterogentity of the application.

params: app_id, rule_schma_config
return value : <ACK>

3.3 Application Development and Usage

The installation of platform is a pre-requisite for it be used for development and deployment of client applications. The platform specific libraries are need to be referenced by the developer of the client application for successful registration. The client application needs to be build with appropriate linked flag to the library file corresponding to logic server. The application development starts by saving the application source files, config files as a new project template. The compressed format of these files can be shipped for deployment on various devices as a compressed archive.

This application on deployment on devices should let an end user of the application with -

- Registration process: To use the Iot platform , the end-user has to be registered with the system by providing its details.
- Login process: To get sensor data and notifications , user has to login to the system with his credentials.
- Set Rules : The user has to set the rules so that he gets notification once the rule is satisfied by sensor data.
- To view sensor data by location / by type : The user can anytime ask for sensor data and he can view it as by the sensor location or by its type