Internal of Application Servers Application Platform

User Guide

Submitted By:

Group 2

Contents

CONFIGURING THE SENSORS

- ✓ The ***Application Developer (user)*** needs to first register the sensor to the system.
- ✓ Once the sensor is registered all its information like
  - **ID –** Each sensor will have a unique ID associated with it.
  - **Type –** This will indicate the type of sensor like temperature, motion etc.
  - **Location** - This will indicate the position of the sensor.
  - **Gateway Assigned** - Each sensor will have a gateway assigned to itself.
- ✓ Necessary JAR files need to be included for each of the type handler
- ✓ Above information about the sensor will be stored in the repository documents. All the information about the sensor will be stored in JSON format in the database since ***MongoDB*** is used.

Example JSON format

```
{
        "ID" : "Alphanumeric",
        "Type" : "String",
        "Location" {
                "Longitude" : "In Radians"
                "Latitude" : "In Radians"
        },
        … … … …
        "Gateway Assigned" : "String"
}
```

CONFIGURING THE GATEWAY

- ✓ When the ***Gateway*** boots up for the first time it obtains information about the sensors which are registered with the system from the repository documents.

- ✓ Once the gateway boots up successfully it is now ready to receive data from the sensors.
- ✓ Each type of sensor will have its associated ***Type Handler*** with the gateway.
- ✓ If the developer intends to register a new type of sensor with the system then appropriate ***JAR files*** need to be included for the type handler to recognize the newly registered sensor. This would be required to convert the data to the required format.
- ✓ To check whether the gateway has started successfully ***Health Ping*** messages can be used.

APIs to Fetch Data

The Application developer needs to develop a logic server that handles the logic of the query given to the platform. The logic server will process the query sent to the platform requesting the desired data. For e.g. receiving an alert when the temperature of sensor 1 rises above 30 degree Celsius. These kind of conditions are handled by the logic server. The logic server has at its disposal three APIs available:

1. **Query API**: The Query API has parameters such as type, location (GPS coordinates) and range. This Query API return a list of all the sensors that match the criteria of the condition. This is required

for instance when Application demands a temperature of a specific area only. Then the logic server needs to check whether there is a sensor available in that are or in a location nearby. The response is in JSON format containing the list of sensors.

URL: /query/


```
{"sensors":[
    {"id":"1", "type":"temperature",
"location":"103", "alive":"yes"},
    {"id":"2", "type":"humidity",
"location":"101", "alive":"yes"},
    {"id":"3", "type":"smoke", "location":"105",
"alive":"no"},
 ]}
```


2. **Command API**: The Logic Server can register commands onto a Filter server to fulfil the user applications demands. The command can be of any form. Eg. Fetch the data of sensor with id = 2 or Fetch the data of sensor with type = “temperature” and location = “103”. This call is directly registered with the Filter server and it starts sending the required data to logic server as and when it receives it.

   URL: /command/

3. **Callback API**: The CallBack API is the one in which data is sent by filter server based on some condition. For e.g. send me the humidity data when the humidity crosses 25 unit mark. In this case, the data is only sent when the value crosses this limit. All the other times when the value is lower than 25, the data is discarded by Filter server and not forwarded to logic server. The response received is in JSON format which needs to be parsed accordingly by the developer to display on the application.

   URL: /callback/


```
{"sensors":[{
  "id":"1",
  "type":"humidity",
  "location":"103",
  "data":"26.7",
  "timestamp":"reading_time"},
  {
  "id":"2",
  "type":"humidity",
  "location":"101",
  "data":"30.7",
  "timestamp":"reading_time"},
```

```
    {"id":"3",
    "type":"humidity",
    "location":"105",
    "data":"27.2",
    "timestamp":"reading_time"},
 ]}
```

Defining Rules

Rules are conditions defined for some specific action to be taken. The rules can vary from simple conditions to complex conjunctions of different conditions. The actions can be specified with the rules that are to be taken once the specified conditions are met. For e.g. when there is smoke, blow alarm and call this number for help.

```
{"rules":[{
    "condition":"smoke",
    "action":"alarm&call",
    "location":"home",
    },
    {
    "condition":"temperature",
```

```
    "action":"increase",
    "location":"dining",
    "threshold":"30",
    "time":{
        "start": "startTime",
        "end": "endTime"}
    },
]}
```