Internet of Things Apps Platform

A User Guide

Submitted by

Group Number - 5

Team 1 :

Sourabh Dhanotia	201405605
Lokesh Walase	201405597
Pankaj Shipte	201405614

Team 2 :

Atul Rajmane	201405529
Prerna Chauhan	201405544
Aditi Jain	201405549
Abhinaba Sarkar	201405616

Team 3 :

Veerendra Bidare	201405571
Swapnil Pawar	201405513
Abhishek Mungoli	201405577

Under the guidance of

Mr. Ramesh Loganathan

For the course

Internals of Application Server

IIIT, Hyderabad

March, 2015

Abstract

The user manual below is to assist Application admin and Application developer to get started with building an IoT application. Application admin uses a few configuration files and Application developer uses REST APIs to fetch data.

Contents

Chapter 1

Configuration Manual for the Application Admin

This manual guides the app admin through the configuration level settings that need to be done before an application tier can start using the platform. Therein, user needs to update a few files viz. sensor registry, sensor config dictionary, application rules dictionary. All these files are maintained in a JSON format. Let's look at the updates required in each of those files one by one.

1.1 Sensor Registry

As the name suggests this file holds all the sensors registered with the platform. Repository entry describes key attributes of the sensor viz ID, application which it is a part of, family that it belongs to.

An entry should be added to this file for every new sensor being plugged in to the platform. The incoming data from only the registered sensors are allowed to make it past the Gateways.

Any number of sensors belonging to the same or different family/ies can be registered with the platform.

```
{
    "SensorOneID" : {
        "app" : "App1",
        "family" : "typeX"
    },
    "SensorTwoID" : {
        "app" : "App2",
        "family" : "typeY"
    },
    "SensorThreeID" : {
        "app" : "App2",
        "family" : "typeX"
    },
    ...
    ...

    // new entry goes here
}
```

Listing 1: Snapshot of the registry file at any given time

It can be noted in the above snapshot of the repository that each entry conforms to the JSON format.

1.2 Sensor configuration dictionary

This is the file that holds the specifications of the data generation attributes of sensors. They include -

1. Protocol the sensor uses to communicate with gateways. It can be one of BLE, Bluetooth, WiFi, etc.

2. Payload the sensor generates. For each data key its length and any additional setting can be provided

3. Payload length in bytes

4. Interval at which the sensor generates and pushes the payload. The interval is in milli seconds.

5. Family the sensor belongs to

6. Application the sensor is a part of

The fifth and sixth attributes above do not have a dedicated key but they themselves form the key for a given entry. And, Except for the payload object all the attributes follow the similar and fixed length format.

```
{
    "appOne_FamilyOne" : {
        "payloadLength" : "someInteger",
        "protocol" : "XXX",
        "pushInterval" : "someInteger",
        "payload": {
            "dataKey1" : {
                "length" : "someInteger",
                // less than or equal to payloadLength
            }
        }
    },
    ...,
    ...,
    // new entry goes here.
}
```

Listing 2: Snapshot of the sensor configuration file

An entry needs to be added for a given family of sensors and a given application. So, if your application uses sensors belonging to different families you are required add multiple entries, each of them corresponding to one family.

Note that key for each entry is a concatenation of the application id and the sensor family.

1.3 Rules configuration dictionary

Each application using the platform can have multiple rules for each family of sensors. These rules range from simplistic threshold rules to more complex composite rules that take into consideration the time of the day, the nature of the data, etc.

The admin can add as many rules required to this file. The filter server makes runs the data received against these rules and notifies the application tier to take the action specified.

For each of the rules, key is the concatenation of the application name and sensor family. Members of a rule are its name, actual rule, action that needs to taken. Except the actual rule all other members follow a fixed format as in rule object can have varying number of members.

```
{
    "AppOne_FamilyOne" : {
        "name" : "threshold",
        "action" : "switchOnShower", // some action
        "rule" : {
            "lowerThreshold" : "someValue1",
            "upperThreshold" : "someValue2"
        }
    },
    "AppOne_FamilyTwo" : {
        "name" : "danger",
        "action" : "blowAlarm", // some action
        "rule" : {
            "startTime" : "8:00 hours",
            "endTime" : "20:00 hours",
            "lowerThreshold" : "someValue1",
            "upperThreshold" : "someValue2"
        }
    },
    "AppTwo_FamilyOne" : {
        "name" : "danger",
        "action" : "evacuationNotification", // some action
        "rule" : {
            "gasType" : "poisonous"
        }
    },
    ...,
    ...,
    // new rule goes here
}
```

Listing 3: Snapshot of rules dictionary

Chapter 2

REST API User Manual for Application Developer

The primary way for apps to read data from the platform is using REST APIs. It's a low-level HTTP-based API that you can use to query data, and a variety of other tasks that an app-tier might need to do. The REST API will have multiple versions available, and the first version of which will have the following APIs exposed.

1. /v1/<sensor_family> "sensor_type" will be the one registered in the sensor registry.

   Payload -

   start:<start timestamp>

   end:<end timestamp>

   access_key: <secret key which the app uses to communicate with API>

   Eg: /v1/temperature, /v1/humidity

   The above API will provide data of the specific type of sensors within the time window specified by the start time and end time. The response will

be in the following format -

```
{
    "version": "1.0",
    "timestamp": "<response timestamp generated at server>",
    "comments": "<comment specific to this response, can be empty>",

    "request": {
    "sensorFamily": "<sensor_family_sent_in_request>",
    "startTime": "<start_time_sent_in_request>",
    "endTime": "<start_time_sent_in_request>",
    },

"data": {
    "sensors-data" : [
    {"sensorId" : "<Id>",
    "recording" : "<composite object specific to type of sensor>",
    "timestamp" : "<timestamp of recording by sensor>",
    "locationId":"<location Id>"
    },
    {"sensorId" : "<Id>",
    "recording" : "<composite object specific to type of sensor>",
    "timestamp" : "<timestamp of recording by sensor>",
    "locationId":"<location Id>"
    },
    ...,
    ...,

    ]
}
```

Listing 4: The response of /v1/<sensor_family>

2. /v1/location/<locationId>

   Payload -

   start:<start timestamp>

   end:<end timestamp>

   access_key: <secret key which the app uses to communicate with API>

   Eg: /v1/100

The above API will provide data of all type of sensors at that location within the time window specified by the start time and end time. The response will be in the following format -

```
{
    "version": "1.0",
    "timestamp": "<response timestamp generated at server>",
    "comments": "<comment specific to this response, can be empty>",
    "request": {
"locationId": "<location_id_sent_in_request>",
"startTime": "<start_time_sent_in_request>",
"endTime": "<start_time_sent_in_request>",
},
"data": {
    "sensors-data" : [
    {"sensorId" : "<Id>",
    "recording" : "<composite object specific to type of sensor>",
    "timestamp" : "<timestamp of recording by sensor>"
    },
    {"sensorId" : "<Id>",
    "recording" : "<composite object specific to type of sensor>",
    "timestamp" : "<timestamp of recording by sensor>"
    },
    ...,
    ...,

    ]
}
```

Listing 5: The response of /v1/location/<locationId>