ABIDI RESTful API for Data Offloading in IoT context

Documentation

Version 1.0

Authors:
Lluc Bono Rosselló, Gonzalo Calderón Lillo

Introduction

This API aims to facilitate communication between the different entities in an IoT architecture. It provides a generic data schema for any type of sensor and methods to exchange data according to pre-established protocols.

In addition, all the necessary resources for its execution are included in the directory, as well as a Dockerfile that allows the application to be deployed directly on any machine after installing Docker [1].

Installation

The execution of this application only requires the download of the directory and its execution via Docker Compose.

So once you have made sure that Docker is installed on the machine where it is to be run, simply run the following command in the terminal:

```
docker-compose -f local.yml up
```

Scheme

The scheme of this API only contains one generic model. This model aims to represent any instance of data for an individual sensor. Therefore, the specific values that are case-dependent (parameters of the specific sensor) are contained in the field *Values*.

Payload

Generic Model that contains message data and JSON data from sensor/edge device

`id` integer

Automatically generated Unique identifier for the given message.

`ip` string required

IP of the sensor

Example: `"b216::1a10:4e00:501:15"`

`date` integer required

Date from the message

Example: `1637678232454`

`type` string required

Type of the Sensor

Example: `BatSense`

`created` string<date>

`valid` boolean

Set to True if values are within the range

Default: `true`

`values` array[object] required

JSON Containing data from an edge device. This data can have different structure depending on the sensor and use case

- **`id`** string
- **`date`** integer
- **`parameterId`** string
- **`value`** number

Methods

Several methods are initially proposed and try to capture the main uses of these applications in this context. Therefore, there are methods for: posting data (single or multiple; getting data (single or multiple, as well as generic queries); deleting data (single or multiple) in order to empty the database when the data is no longer needed on that device.

Post Data from Single Device

POST `http://localhost:8000/ec/payloads/`

Post Data from a single Sensor

Request

Body

application/json

- **ip** string — required
  >= 1 characters
- **date** number — required
- **type** string — required
  >= 1 characters
- **values** array[object] — required
  >= 1 items
  - **id** string — required
    >= 1 characters
  - **date** number — required
  - **parameterId** string — required
    >= 1 characters
  - **value** number — required

Post Data from Multiple Devices

```
POST http://localhost:8000/ec/payloads/multiple
```

Post Data from multiple sensors. This method should be used to move data from the server to the edge, if needed.

Request

Body

application/json

Array of JSON objects containing several Sensor data.

array of:

- **ip** string — required
  - >= 1 characters
- **date** number — required
- **type** string — required
  - >= 1 characters
- **values** array[object] — required
  - >= 1 items
  - **id** string — required
    - >= 1 characters
  - **date** number — required
  - **parameterId** string — required
    - >= 1 characters
  - **value** number — required

Get Data from ID

`GET` `http://localhost:8000/ec/payloads/{id}/`

Get message from its ID

Request

Path Parameters

`id` number — required

ID of the message to get or delete

Get List of Messages from Date

`GET` `http://localhost:8000/ec/payloads/offload/`

Get List of messages from a given date. This method should be used to offload data to the server.

Request

Query Parameters

`date` number — required

Date to get messages from

Get List of Messages with Validation

```
GET http://localhost:8000/ec/payloads/valid_items/
```

Get list of messages containing (or not) validation errors. Validation errors are usually configured regarding missing data or values out of range. Valid param should be a boolean: True or False.

Request

Query Parameters

`valid` boolean required

To get values from sensors with errors or not.

Get List of Messages from device IP

`GET` `http://localhost:8000/ec/payloads/sensor/`

Get list of messages from a device (or devices) connected through a given IP

Request

Query Parameters

`ip` string — required

Ip of the device to get messages from.

Responses 200

OK

Delete Data from ID

```
DELETE  http://localhost:8000/ec/payloads/{id}/
```

Request

Path Parameters

`id` number required

ID of the message to get or delete

Delete a List of Messages from Date

```
DELETE http://localhost:8000/ec/payloads/offload/
```

Delete a list of messages containing a given date. This method should be used to remove data that has already been moved to the server and has no longer use in the edge.

Request

Query Parameters

`date` number required

Date to delete messages from

Get Data from ID

```
GET http://localhost:8000/ec/payloads/{id}/
```

Get message from its ID

Request

Path Parameters

`id` number required

ID of the message to get or delete

Examples

For instance, once the API is running on one server, edge devices can send data using the POST method described above. In this example the data is sent by making use of the terminal and the format described in the methods:

```
curl --request POST \
 --url http://localhost:8000/ec/payloads \
 --header 'Content-Type: application/json' \
 --data '{
 "ip": "b216::1a10:4e00:501:15",
 "date": 1637678232454,
 "type": "BatSense",
 "values": [
   {
     "id": "b216::1a10:4e00:501:14-PAPP1637678232454",
     "date": 1637678232454,
     "parameterId": "b216::1a10:4e00:501:14-PAPP",
     "value": 0
   },
   {
     "id": "b216::1a10:4e00:501:14-EAPP1637678232454",
     "date": 1637678232454,
     "parameterId": "b216::1a10:4e00:501:14-EAPP",
     "value": 417
   },
   {
     "id": "b216::1a10:4e00:501:14-PACT1637678232454",
     "date": 1637678232454,
     "parameterId": "b216::1a10:4e00:501:14-PACT",
     "value": 0
   },
   {
     "id": "b216::1a10:4e00:501:14-IL1637678232454",
     "date": 1637678232454,
     "parameterId": "b216::1a10:4e00:501:14-IL",
     "value": 0
   },
   {
     "id": "b216::1a10:4e00:501:14-UL1637678232454",
     "date": 1637678232454,
     "parameterId": "b216::1a10:4e00:501:14-UL",
     "value": 229
   },
   {
     "id": "b216::1a10:4e00:501:14-EACT1637678232454",
     "date": 1637678232454,
     "parameterId": "b216::1a10:4e00:501:14-EACT",
     "value": 141
   }
 ]
}'
```

In case that data wants to be offloaded to another device, a GET method can be used following a similar procedure as provided:

```
curl --request GET \
 --url http://localhost:8000/ec/payloads/id \
 --header 'Content-Type: application/json'
```

References

[1] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.