

Water Level Data access tutorial

This is a document explaining how one can access the water level data collected from the farm devices. The data is stored on cloudantDB, which is a No-SQL Document based database.

The water level measurement device is solar powered and is designed to work in remote places that may be off the grid.

Cloudant DB Credentials

Username: 8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix

Password: b69d78f8c5aad962b42e278c73cce8238522bdd4a73aaeec73fa8efc11524db3

Host: 8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com

Port: 443

URL: https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix:b69d78f8c5aad962b42e278c73cce8238522bdd4a73aaeec73fa8efc11524db3@8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com"

Database: levelmeters

Database read url: https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find

Device: Proto06V

Included is a Python application to achieve the same. Functionalities of the code are explained on the comments in the application.

This tutorial is based on the Cloudant DB documentation. Use the link below for more information.
<https://console.bluemix.net/docs/services/Cloudant/basics/index.html#cloudant-basics>

Sample document (Data record).

Below is a sample of a data record from the farm water level measurement device.

```
{
  "_id": "002b6de09c5d70a2317abfa3534ea209",
  "_rev": "1-248745a3d6e2fe8ba637cc83d1add7c1",
  "topic": "iot-2/type/LevelMeter/id/Proto06V/evt/status/fmt/formad:14br69:LevelMeter:Proto06V",
  "payload": {
    "data": {
      "height": "60",
      "signal": "25",
      "battery": "4019",
      "temp": "20",
      "humidity": "18",
      "err": "10"
    }
  },
  "deviceId": "Proto06V",
  "deviceType": "LevelMeter",
  "eventType": "status",
  "format": "formad:14br69:LevelMeter:Proto06V",
  "timestamp": "2017-10-26T17:39:12.401Z"
}
```

FYI:

1. **Height** is in centimetres, and this is the measurement from the top of the tank to the water level, to calculate the amount of water in the tank, subtract this value from the overall height of the tank.
2. **Battery** value is in millivolts.
3. **Signal** value is in dBm.
4. **Temperature** value is in degrees Celsius.
5. **Humidity** value is in percentage.
6. **err** value is the count of failed connections from the device.
7. **deviceId** is the identity of the device.
8. **deviceType** is the identifier for all the devices of the same type, in this case levelmeters.
9. **timestamp** is the time the data was sent from the device in UTC.
10. **topic** is the mqtt topic the device used to communicate with the server.

For Analytics purposes, the height is what is required, temperature, humidity, signal level and battery level are for operational purposes.

How to read the data

There are 2 ways in which one can access the database.

1. RESTClient using the HTTP API.
2. Python Application Using HTTP API

For this tutorial we will use the **HTTP API**.

RESTClient using the HTTP API

For this tutorial, we will use a browser add-on on Mozilla Firefox called rest client.

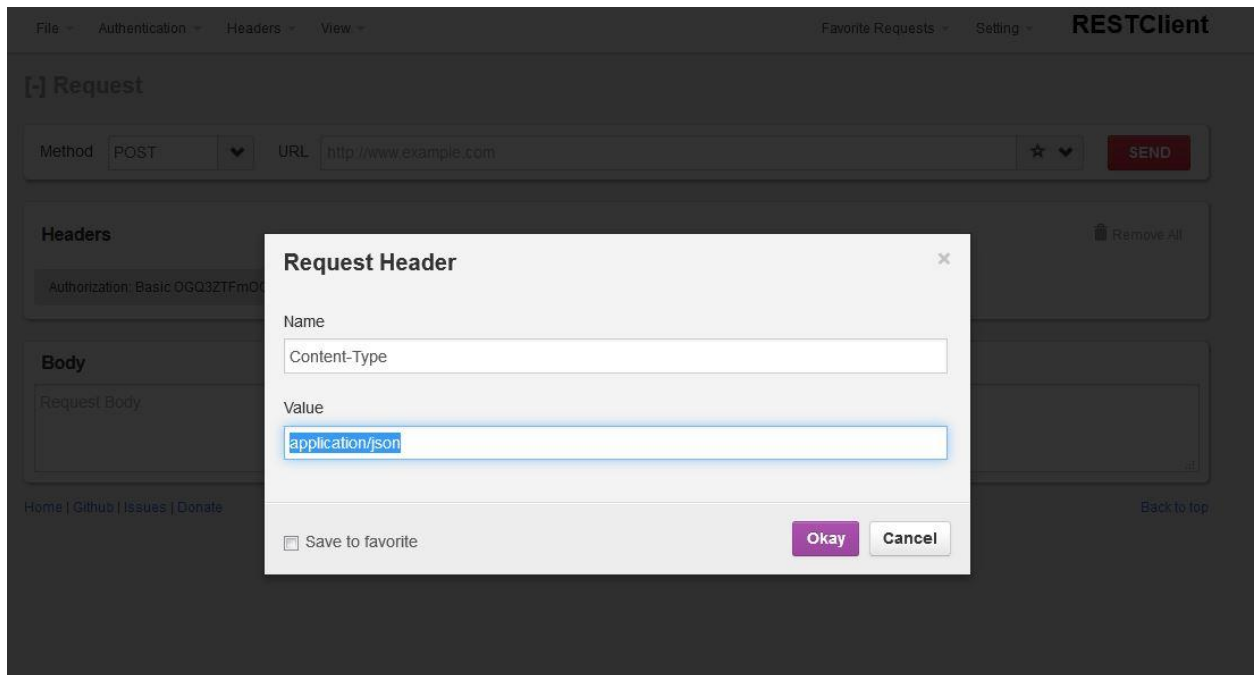
1. Install Restclient on Mozilla Firefox.

<https://addons.mozilla.org/en-US/firefox/addon/restclient/>

2. Once installed, open RESTclient on Mozilla. You should see the page below.



3. Populate the content type header, click on headers, set the **name** to **Content-type** and **value** to **application/json** and click Okay.



4. Populate the request method, URL and Body.
 - a. Set Method to **POST**
 - b. Set URL to https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find
 - c. Set the Body to

```
{"selector":{"timestamp":{"$gt":"2017-11-07T12:17:48.412Z","$lt":"2017-11-08T12:17:48.412Z"},"deviceId":"Proto06V"},"sort":[{"timestamp":"desc"]}}
```

The above is a selector object that is set on the body of the object to retrieve the data.

The above statement will retrieve data for the **Proto06V** device, between **2017-11-07T12:17:48.412Z** and **2017-11-08T12:17:48.412Z** in a **descending order**

The time record in the database is in UTC.

Once all is populated, the page should be similar to this.

File Authentication Headers View Favorite Requests Setting RESTClient

[+] Request

Method POST URL https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find ☆ SEND

Headers Remove All

Content-Type: application/json

Body

```
{
  "selector": {
    "timestamp": {
      "$gt": "2017-11-07T12:17:48.412Z",
      "$lt": "2017-11-08T12:17:48.412Z"
    },
    "deviceId": "Proto06V"
  },
  "sort": [
    [
      "timestamp",
      "desc"
    ]
  ]
}
```

5. Click on Send to send the request. If you are not logged on to Cloudant on the browser, you will get a prompt to enter the username and password given above.

File Authentication Headers View Favorite Requests Setting RESTClient

[+] Request

Method POST URL https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find ☆ SEND

Headers Remove All

Content-Type: application/json

Body

```
{
  "selector": {
    "timestamp": {
      "$gt": "2017-11-07T12:17:48.412Z",
      "$lt": "2017-11-08T12:17:48.412Z"
    },
    "deviceId": "Proto06V"
  },
  "sort": [
    [
      "timestamp",
      "desc"
    ]
  ]
}
```

Authentication Required

https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com is requesting your username and password. The site says: "Cloudant Private Database"

User Name: 8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix

Password:

OK Cancel

[+] Response

Response Headers

Status Code 401 Unauthorized

6. If everything was set up correctly, you should get a response status code 200 on the response headers and the collected data from the response body.

The screenshot shows the RESTClient interface. The Request tab is active, displaying a POST request to `https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find`. The Headers section shows `Content-Type: application/json`. The Body contains a JSON selector query. The Response tab is also active, showing a 200 OK status code and various response headers.

Request

Method: POST URL: `https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find`

Headers

Content-Type: application/json

Body

```
{\"selector\":{\"timestamp\":{\"$gt\":\"2017-11-07T12:17:48.412Z\",\"$lt\":\"2017-11-08T12:17:48.412Z\"},\"deviceid\":\"Proto06V\"},\"sort\":{\"timestamp\":\"desc\"}}
```

Response

Response Headers

```
1. Status Code : 200 OK
2. Cache-Control : must-revalidate
3. Content-Type : application/json
4. Date : Wed, 08 Nov 2017 14:25:44 GMT
5. Server : CouchDB/2.0.0 (Erlang OTP/17)
6. Strict-Transport-Security : max-age=31536000
7. Transfer-Encoding : chunked
8. X-Cloudant-Backend : bm-cc-uk-01
9. X-Cloudant-Request-Class : query
10. X-Content-Type-Options : nosniff
11. X-Couch-Request-ID : bcc842e50c
12. X-Frame-Options : DENY
13. via : 1.1 1b1.bm-cc-uk-01 (Gluon/1.41.0)
```

The screenshot shows the RESTClient interface. The Request tab is active, displaying a POST request to `https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find`. The Headers section shows `Content-Type: application/json`. The Body contains a JSON selector query. The Response tab is also active, showing a 200 OK status code and various response headers. The Response Body (Raw) is displayed, showing a large JSON array of documents.

Request

Method: POST URL: `https://8d7e1f8e-1f7d-4d64-8788-490379bcbad5-bluemix.cloudant.com/levelmeters/_find`

Headers

Content-Type: application/json

Body

```
{\"selector\":{\"timestamp\":{\"$gt\":\"2017-11-07T12:17:48.412Z\",\"$lt\":\"2017-11-08T12:17:48.412Z\"},\"deviceid\":\"Proto06V\"},\"sort\":{\"timestamp\":\"desc\"}}
```

Response

Response Headers

Response Body (Raw)

```
{\"docs\":[
  {\"_id\":\"a4f60b65dc3bb691fdf2fd08f9558c9f\", \"_rev\":\"1-2d8c2a6e60f0011ae24f52da9c439bab\", \"topic\":\"iot-2/type/LevelMeter/1d/Proto06V/evt/status/fmt/format:14br69:LevelMeter:Proto06V\", \"payload\":{\"data\":{\"height\":\"66\", \"signal\":\"25\", \"battery\":\"4189\", \"temp\":\"28\", \"humidity\":\"14\", \"err\":\"347\"}}, \"deviceId\":\"Proto06V\", \"deviceType\":\"LevelMeter\", \"eventType\":\"status\", \"format\":\"format:14br69:LevelMeter:Proto06V\", \"timestamp\":\"2017-11-08T12:15:28.266Z\"},
  {\"_id\":\"83360bc4015c52569437a55b69fd4d06\", \"_rev\":\"1-2fe516c65c77ce45193a0618b64b15c6\", \"topic\":\"iot-2/type/LevelMeter/1d/Proto06V/evt/status/fmt/format:14br69:LevelMeter:Proto06V\", \"payload\":{\"data\":{\"height\":\"65\", \"signal\":\"25\", \"battery\":\"4189\", \"temp\":\"28\", \"humidity\":\"14\", \"err\":\"347\"}}, \"deviceId\":\"Proto06V\", \"deviceType\":\"LevelMeter\", \"eventType\":\"status\", \"format\":\"format:14br69:LevelMeter:Proto06V\", \"timestamp\":\"2017-11-08T12:10:25.818Z\"},
  {\"_id\":\"5908ced6e51e756eda9521cd30f1df48\", \"_rev\":\"1-a84585ea2f039bc8f2eeb0cbb5a0e3a0\", \"topic\":\"iot-2/type/LevelMeter/1d/Proto06V/evt/status/fmt/format:14br69:LevelMeter:Proto06V\", \"payload\":{\"data\":{\"height\":\"65\", \"signal\":\"25\", \"battery\":\"4182\", \"temp\":\"28\", \"humidity\":\"14\", \"err\":\"347\"}}, \"deviceId\":\"Proto06V\", \"deviceType\":\"LevelMeter\", \"eventType\":\"status\", \"format\":\"format:14br69:LevelMeter:Proto06V\", \"timestamp\":\"2017-11-08T12:05:23.254Z\"},
  {\"_id\":\"1574577a6829871fbbf294916e585e41\", \"_rev\":\"1-df5eb743fb21da688fa09c78e3bc7a30\", \"topic\":\"iot-2/type/LevelMeter/1d/Proto06V/evt/status/fmt/format:14br69:LevelMeter:Proto06V\", \"payload\":{\"data\":{\"height\":\"64\", \"signal\":\"25\", \"battery\":\"4182\", \"temp\":\"28\", \"humidity\":\"14\", \"err\":\"347\"}}, \"deviceId\":\"Proto06V\", \"deviceType\":\"LevelMeter\", \"eventType\":\"status\", \"format\":\"format:14br69:LevelMeter:Proto06V\", \"timestamp\":\"2017-11-08T12:00:20.857Z\"},
  {\"_id\":\"383bf2d83f7dd9461bf546270f45b864\", \"_rev\":\"1-d18f0bd0a2a7336faa4a2e5155dae97e\", \"topic\":\"iot-2/type/LevelMeter/1d/Proto06V/evt/status/fmt/format:14br69:LevelMeter:Proto06V\", \"payload\":{\"data\":{\"height\":\"64\", \"signal\":\"25\", \"battery\":\"4182\", \"temp\":\"28\", \"humidity\":\"14\", \"err\":\"347\"}}, \"deviceId\":\"Proto06V\", \"deviceType\":\"LevelMeter\", \"eventType\":\"status\", \"format\":\"format:14br69:LevelMeter:Proto06V\", \"timestamp\":\"2017-11-08T12:00:20.857Z\"}
]}
```

Python Application Using HTTP API

The attached Python application will achieve the same as above.

You will need to have python and pip installed on your machine and connected to the internet.

1. Run the python application.

Command: *python cloudantTutorial.py*

2. Once the above command is run, you will get a prompt to enter the number of hours into the past that you would like to view the data for and press enter. E.g. 3 hours as below

```
[root@zaz-on3 Strathmore]#  
[root@zaz-on3 Strathmore]# python cloudantTutorial.py  
Fetch data for the past ...hrs ? : 3
```

3. You will then get a prompt to enter the limit of the data records to fetch and press enter. E.g. 10 as below

```
[root@zaz-on3 Strathmore]# python cloudantTutorial.py  
Fetch data for the past ...hrs ? : 3  
Enter the limit of the documents to fetch ? : 10
```

4. You should get an output showing the request body object and response, which is the calculated height of the water in the tank in cm and the timestamp in UTC. E.g.

```
Fetch data for the past ...hrs ? : 3  
Enter the limit of the documents to fetch ? : 10  
Fetching 10 latest documents for the past 3 hours  
Request Body Object...  
{'sort': [{'timestamp': 'desc'}], 'limit': 10, 'selector': {'timestamp': {'$gt': '2017-11-08T11:46:47.000Z', '$lt': '2017-11-08T14:46:47.000Z'}, 'deviceId': 'Proto06V'}}  
Response status code.. 200  
Calculated Height in cm  
57 cm --- 2017-11-08T14:41:45.935Z  
57 cm --- 2017-11-08T14:36:43.530Z  
58 cm --- 2017-11-08T14:31:41.093Z  
58 cm --- 2017-11-08T14:26:39.527Z  
59 cm --- 2017-11-08T14:21:36.009Z  
59 cm --- 2017-11-08T14:16:34.086Z  
60 cm --- 2017-11-08T14:11:31.044Z  
60 cm --- 2017-11-08T14:06:28.642Z  
60 cm --- 2017-11-08T14:01:26.089Z  
61 cm --- 2017-11-08T13:56:23.579Z
```