Sensecap Openapi Document

Introduction

HTTP API

HTTP API Reference

Data OpenStream API

Appendix

SenseCAP API Introduction

SenseCAP API is the interface to manage devices and data besides the SenseCAP Web Portal. SenseCAP API consists of HTTP API and Data OpenStream API.

With SenseCAP HTTP API, you can manage your LoRa and NB-IoT devices from your private cloud service, retrieve historical data in raw or segment format.

With Data OpenStream API, you can monitor the measurements from sensors in realtime.

HTTP API Quickstart

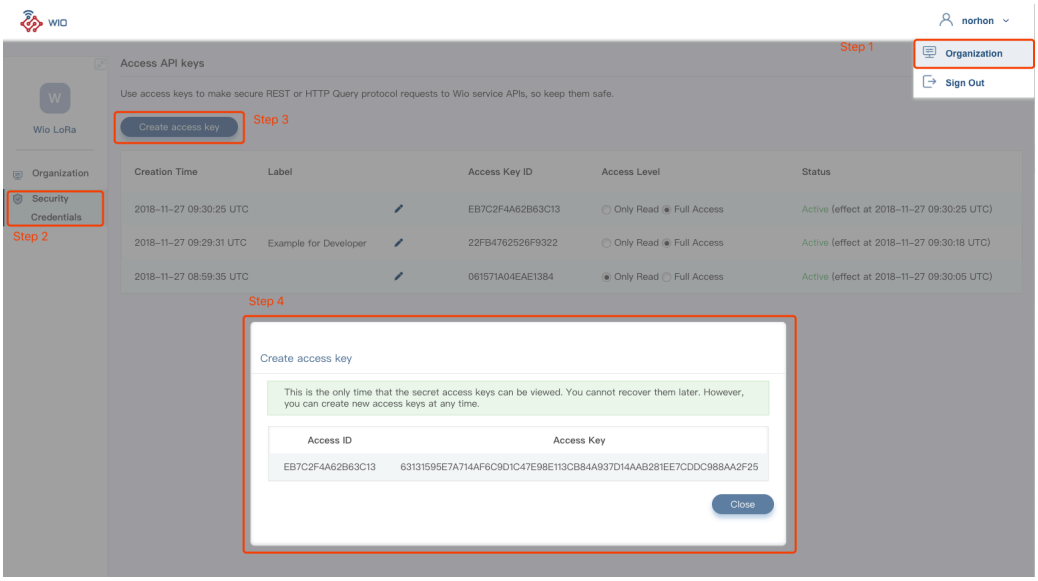
Summary:
In this quickstart we're gonna show you how to make your first HTTP API call to SenseCAP HTTP API backend.

Prerequisite

You should have your SenseCAP account from SenseCAP sales, and make sure you can login the SenseCAP Web Portal https://sensecap.seeed.cc .

Get an Access Key

1. Login the SenseCAP Web Portal https://sensecap.seeed.cc
2. Navigate to "Organization/Security Credentials"
3. Click "Create access key"
4. Set down the Access ID and Access Key



Get all the Deivice EUIs

In this quickstart, we will use curl to issue HTTP requests. The example below calls an API to retrieve all the device EUIs belonging to you.

```
curl --user "<access_id>:<access_key>" \
     https://sensecap-openapi.seeed.cc/1.0/lists/devices/eui
```

You should replace and with the one you got before. The command will output like the following

```
   {
     "code": "0",
     "data": {
       "gateway": [
         "2CF7F1121130003F",
         "2CF7F1100470001A"
       ],
       "node": [
         "2CF7F12210400031",
         "2CF7F12210400005",
         "2CF7F12210400023"
       ]
     },
     "time": 0.314
   }
```

HTTP API Access Guide

HTTP Methods

The SenseCAP API utilizes five HTTP methods for accessing resources:

- GET
    - Make a GET request to retrieve data. GET requests will never cause an update or change to your data because they're safe and idempotent.
- POST
    - Use a POST request to create new resources. For example, make a POST request to a device group where the body of your request JSON is a new group.
- PATCH
    - Make a PATCH request to update a resource. With PATCH request, you only need to provide the data you want to change.
- PUT
    - Use a PUT request to create or update a resource. This is most useful for syncing subscriber data.
- DELETE
    - Make a DELETE request to remove a resource.

> ℹ **Note:** If your ISP doesn't permit HTTP operations other than GET or POST, please use HTTP Method tunneling - make your call with POST, but include the method you intend to use in an X-HTTP-Method-Override header.

HTTP Request and Response

Requests are authenticated with the HTTP Basic Authentication.

HTTP Basic Authentication

HTTP Basic Authentication is one of the most common ways for RESTfull API authentication. We use Access ID as username and Access Key as password. Every HTTP client library should have its built-in support for Basic Authentication, in this documentation we use curl, which uses the –user option to specify Basic Authentication credential.

u can create access keys via SenseCAP Web Portal . Please refer to quickstart to see how to get an access key.

API Response

All response fields follow the lowercase and underscore convention

Successful Response with String

```
{
    "code":"0",
    "data":"
        // string
    "
}
```

Successful Response with Object

```
{
    "code":"0",
    "data":{
        // object
    }
}
```

Successful response with Array

```
{
    "code":"0",
    "data":[
        // Array
    ]
}
```

Error Response

```
{
    "code":"1001",
    "msg":"error message"
}
```

Group Management API

create a new device group

POST {host}/1.0/group

create a new device group, you can only create up to 50 groups.

Body Parameters

name (required)	string	custom group name

response

```
{
    "code": "0",
    "data": {
        "name": "custom group name",
        "unique_name": "39AE810FF660B42F"
    }
}
```

Example request

```
curl --request POST \
     --url '{host}/1.0/group' \
     --user '<username>:<password>' \
     --header 'content-type: application/x-www-form-urlencode
d' \
     --data '{"name":"device group name"}' \
     --include
```

update device group information

PATCH {host}/1.0/group/:unique_name

update the device group name

Path Parameters

unique_name (required)	string	the unique name of a device group

Body Parameters

name (required)	string	the new name of device group

response

```
{
  "code": "0",
  "data": ""
}
```

Example request

```
curl --request PATCH \
     --url '{host}/1.0/group/:unique_name' \
     --user '<username>:<password>' \
     --header 'content-type: application/x-www-form-urlencode
d' \
     --data '{"name":"new name of the device group"}' \
     --include
```

delete a device group

DELETE {host}/1.0/group/:unique_name

delete a device group and the devices will be moved to the Default group.

Path Parameters

unique_name (required)	string	the unique name of a device group

response

```
{
  "code": "0",
  "data": ""
}
```

Example request

```
curl --request DELETE \
     --url '{host}/1.0/group/:unique_name' \
     --user '<username>:<password>' \
     --header 'content-type: application/json' \
     --include
```

list device group

DELETE {host}/1.0/lists/group

get the list of all groups

response

```
{
    "code": "0",
    "data": [
        {
          "name": "group1",
          "group_unique_name": "220BD4AD87EB6B68",
          "dev_cnt": "3",
          "online_cnt": "1",
          "created": "1563947209000"
        }
    ]
}
```

Example request

```
curl --request GET \
     --url '{host}/1.0/lists/group' \
     --user '<username>:<password>' \
     --include
```

list device group with devices

DELETE {host}/1.0/lists/group/devices

List all groups and devices in each group

The Default group is a virtual group, its group_unique_name is empty. All devices will be claimed into the Default group, users could then reorganize them with different groups.

response

```
{
  "code": "0",
  "data": {
    "node": [
      {
        "group_name": "Default",
        "group_unique_name": "",
        "devices": []
      },
      {
        "group_name": "device group name",
        "group_unique_name": "DAF6EECDCDCA6BB2",
        "devices": [
          {
            "dev_eui": "2CF7F12004100005",
            "dev_name": "device name of LoRa node",
            "lon": "113.32134",
            "lat": "22.4523",
            "online_status": "1",  //  "0" offline, "1" online
            "battery_status": "1"  //  "0" battery low power,
"1" battery power OK
          }
        ]
      }
    ]
  }
}
```

Example request

```
curl --request GET \
     --url '{host}/1.0/lists/group/devices' \
     --user '<username>:<poassword>' \
     --header 'content-type: application/json' \
     --include
```

move device to other group

DELETE {host}/1.0/group/:unique_name/devices

Path Parameters

unique_name (required) string
unique name which the group will be moved to

Body Parameters

eui (required)	string	device EUI

response

```
{
  "code": "0",
  "data": ""
}
```

Device Management API

gateway list

GET {host}/1.0/lists/gateway

get the list of gateways

Query Parameters

offset number

current page number, start from 1, default = 1

length string

the number of rows in page, default = 20

```
//response
{
    "code": "0",
    "data": {
        "page": {
            "count": "8"
        },
        "list": [
            {
                "dev_eui": "2CF7F1120410001D",
// gateway EUI, equipment unique identity
                "dev_name": "gtw seeed 2018csc",
// gateway name
                "dev_area": "Asia",
// gateway area, Europe America Asia
                "dev_country": "CHN",
// name country, defined by customer
                "online_status": "0",
// 1 online, 0 offline
                "description": "organization first gateway",
// desc something
                "activate_time": "1543990216000",
// when the customer activates the device, unit milliseconds
                "latestmsg_time": "1543990216000",
// the latest message time, unit milliseconds
                "production_time": "1539747273000",
// the gateway production time, unit milliseconds
                "frequency": "CN_470_510",
// frequency
                "position": {
// geographic, set by customer
                    "lat": "22.2235",
// geographic.latitude
                    "lon": "113.21",
// geographic.longitude
                "ver_hardware": "1.0.3",
// hardware version
                "ver_software": "1.0.2",
// software version
                "antenna_placement": "indoor",
// antenna placement, set by customer
            }
        ]
    }
}
```

Example request

```
curl --request GET \
     --url {host}/1.0/lists/gateway?offset=1&length=2 \
     --user '<username>:<password>'
```

get gateway detail

GET {host}/1.0/gateway/:gateway_eui

get the detail of a specific gateway

Path Parameters

gateway_eui (required)	string	device EUI

```
//response
{
    "code": "0",
    "data": {
        "dev_eui": "2CF7F1120410001D",     // gateway EUI, equi
pment unique identity
        "dev_name": "gtw seeed 2018csc",   // gateway name
        "dev_area": "Asia",                // gateway area, Eur
ope America Asia
        "dev_country": "CHN",              // name country, def
ined by customer
        "group_name": "Group for Dev",     // the group name, i
f customer set a group for the device
        "access_read": "68B04B2BD1AE8367A4B501BF1C478024DC901A1
8657AAEBEB4C07AE9B10A0B15",    // device access code for read
        "access_write": "D978F7DF08B3C933CB6C0A23F303F1460572E6
0348E240C8432FB482FC11A7A3",   // device access code for write
        "online_status": "0",                          // 1 onli
ne, 0 offline
        "description": "organization first gateway",  // desc s
omething
        "activate_time": "1543990216000",              // when t
he customer activates the device, unit milliseconds
        "latestmsg_time": "1543990216000",             // the la
test message time, unit milliseconds
        "production_time": "1539747273000",            // the ga
teway production time, unit milliseconds
        "frequency": "CN_470_510",                     // freque
ncy
        "position": {                                  // geogra
phic, set by customer
            "lat": "22.2235",                          // geogra
phic.latitude
            "lon": "113.21"                            // geogra
phic.longitude
        "ver_hardware": "1.0.3",                       // hardwa
re version
        "ver_software": "1.0.2",                       // softwa
re version
        "antenna_placement": "indoor",                 // antenn
a placement, set by customer
    }
}
```

Example request

```
curl --request GET \
     --url {host}/1.0/gateway/:gateway_eui \
     --user '<username>:<password>'
```

update gateway

PATCH {host}/1.0/gateway/:gateway_eui

update gateway information，the following Body Parameters need to be filled in one or more

Path Parameters

gateway_eui (required)	string	device EUI

Body Parameters

group_unique_name	string	created from groups management api
dev_name	string	device name
dev_country	string	ISO 3166-1 country codes, e.g. country code "CHN", means China
position	object	{"lon":"longitude", "lat":"latitude", "alt":"altitude"}
description	object	description for this gateway

antenna_placement object enum:indoor|outdoor

```
//response
{
    "code":"0",
    "data":""
}
```

Example request

```
curl --request PATCH \
     --url {host}/1.0/gateway/:gateway_eui \
     --user '<username>:<password>' \
     --header 'content-type: application/x-www-form-urlencode
d' \
     --data '{"dev_name":"Example for Developer"}' \
     --include
```

delete gateway from organization

DELETE {host}/1.0/gateway/:gateway_eui

remove the binding relationship of this gateway and the organization of API caller, but user can bind it back with SenseCAP App.

Path Parameters

gateway_eui (required) string device EUI

```
//response
{
    "code":"0",
    "data":""
}
```

Example request

```
curl --request DELETE \
     --url {host}/1.0/gateway/:gateway_eui \
     --user '<username>:<password>'
```

node list

GET {host}/1.0/lists/node

get the list of Node devices

Query Parameters

offset number

current page number, start from 1, default = 1

length string

the number of rows in page, default = 20

```
//response
{
    "code": "0",
    "data": {
        "page": {
            "count": "28"                              // total nu
mber
        },
        "list": [
            {
                "dev_eui": "2CF7F12004100005",      // node EU
I, equipment unique identity
                "dev_name": "Example for Dev",      // node nam
e
                "dev_area": "Europe",               // node are
a, Europe America Asia
                "dev_country": "CHN",               // name cou
ntry, defined by customer
                "online_status": "1",               // 1 onlin
e, 0 offline
                "battery_status": "1",              // 1 node b
attery for good, 0 node battery for bad
                "description": "describe",          // desc som
ething
                "activate_time": "1543990216000",   // when th
e customer activates the device, unit milliseconds
                "latestmsg_time": "1543990216000",  // the late
st message time, unit milliseconds
                "production_time": "1539747273000", // the nod
e production time, unit milliseconds
                "frequency": "EU_863_870",          // frequenc
y
                "msg_count": "12",                  // the tota
l number of message that have been sent by the device
                "ver_hardware": "1.3",              // hardwar
e version
                "ver_software": "1.2"              // software
version
            }
        ]
    }
}
```

Example request

```
curl --request GET \
     --url {host}/1.0/lists/node?offset=1&length=2 \
     --user '<username>:<password>'
```

get node detail

GET {host}/1.0/node/:node_eui

get the detail of a specific node

Path Parameters

node_eui (required)	string	device EUI

```
//response
{
    "code": "0",
    "data": {
        "dev_eui": "2CF7F12004100005",      // node EUI, equipm
ent unique identity
        "dev_name": "Example for Dev",      // node name
        "dev_area": "Europe",               // node area, Europ
e America Asia
        "dev_country": "CHN",               // name country, de
fined by customer
        "group_name": "Group for Dev",      // the group name,
if customer set a group for the device
        "access_read": "9872A8956A444FEF6A2B08625534D311F444714
FE5887AF338E42456EEDD8E16",    // device access code for read
        "access_write": "3D4474E08EDC039EB3EC2C3827666446D8D6C2
D31741F8FA271AD9AFA6B4A12D",   // device access code for write
        "online_status": "1",               // 1 online, 0 offl
ine
        "battery_status": "1",              // 1 node battery f
or good, 0 node battery for bad
        "description": "describe",          // desc something
        "activate_time": "1543990216000",   // when the custome
r activates the device, unit milliseconds
        "latestmsg_time": "1543990216000",  // the latest messa
ge time, unit milliseconds
        "production_time": "1539747273000", // the node product
ion time, unit milliseconds
        "frequency": "EU_863_870",          // frequency
        "msg_count": "12",                  // the total numbe
r of message that have been sent by the device
        "ver_hardware": "1.3",              // hardware version
        "ver_software": "1.2",              // software version
        "sensors": [                    // the sensors mounted o
n the device, some devices can mounted more than one sensor
            {
                "sensor_eui": "2CF7F13004200016",           //
sensor EUI
                "sensor_name": "Temperature and Humidity",  //
sensor name
                "sensor_channel": "1",                      //
the channel name or channel tag on which the sensor is mounted
on the device
                "sensor_measure": [                         //
some sensor have more than one measure
```

```
                    {
                        "id": "4097",                       //
measure id
                        "name": "Air Temperature"           //
measure name
                    },
                    {
                        "id": "4098",
                        "name": "Air Humidity"
                    }
                ]
            }
        ]
    }
}
```

Example request

```
curl --request GET \
     --url {host}/1.0/node/:node_eui \
     --user '<username>:<password>'
```

update node

PATCH {host}/1.0/node/:node_eui

update node information, the following Body Parameters need to be filled in one or more

Path Parameters

node_eui (required)	string	device EUI

Body Parameters

dev_name	string	device name
group_unique_name	string	new group ID if you want to move this node

description	string	description of this node

response

```
{
    "code":"0",
    "data":""
}
```

Example request

```
curl --request PATCH \
     --url {host}/1.0/gateway/:gateway_eui \
     --user '<username>:<password>' \
     --header 'content-type: application/x-www-form-urlencode
d' \
     --data '{"dev_name":"Example for Developer"}' \
     --include
```

delete node from organization

DELETE {host}/1.0/node/:node_eui

remove the binding relationship of this node and the organization of API caller, but user can bind it back with SenseCAP App.

Path Parameters

node_eui (required)	string	device EUI

response

```
{
    "code":"0",
    "data":""
}
```

Example request

```
curl --request DELETE \
     --url {host}/1.0/node/:node_eui \
     --user '<username>:<password>' \
```

sensor measure list

GET {host}/1.0/lists/sensor/measure

get the list of all physical measurements of all sensor types. also see Appendix - List of Sensor Types.

```
//response
{
    "code": "0",
    "data": [
        {
            "sensor_id": "1009",
            "sensor_name": "Wind Speed",
            "sensor_measure": [
                {
                    "measure_id": "4112",
                    "measure_name": "Wind Direction",
                    "measure_unit": "°"
                },
                {
                    "measure_id": "4105",
                    "measure_name": "Wind Speed",
                    "measure_unit": "m/s"
                }
            ]
        },
        {
            "sensor_id": "1008",
            "sensor_name": "Wind Direction",
            "sensor_measure": [
                {
                    "measure_id": "4104",
                    "measure_name": "Wind Direction",
                    "measure_unit": "°"
                }
            ]
        },
        {
            "sensor_id": "1006",
            "sensor_name": "Soil temperature and humidity",
            "sensor_measure": [
                {
                    "measure_id": "4103",
                    "measure_name": "Soil Humidity",
                    "measure_unit": "%RH"
                },
                {
                    "measure_id": "4102",
                    "measure_name": "Soil Temperature",
                    "measure_unit": "°C"
                }
```

```
            ]
        },
        {
            "sensor_id": "1005",
            "sensor_name": "Atmospheric Pressure",
            "sensor_measure": [
                {
                    "measure_id": "4101",
                    "measure_name": "Atmospheric Pressure",
                    "measure_unit": "Pa"
                }
            ]
        },
        {
            "sensor_id": "1004",
            "sensor_name": "Carbon Dioxide",
            "sensor_measure": [
                {
                    "measure_id": "4100",
                    "measure_name": "Carbon Dioxide",
                    "measure_unit": "Vol"
                }
            ]
        },
        {
            "sensor_id": "1003",
            "sensor_name": "Light Intensity",
            "sensor_measure": [
                {
                    "measure_id": "4099",
                    "measure_name": "Light Intensity",
                    "measure_unit": "Lux"
                }
            ]
        },
        {
            "sensor_id": "1001",
            "sensor_name": "Air Temperature and Humidity",
            "sensor_measure": [
                {
                    "measure_id": "4098",
                    "measure_name": "Air Humidity",
                    "measure_unit": "%RH"
                },
                {
```

```
                    "measure_id": "4097",
                    "measure_name": "Air Temperature",
                    "measure_unit": "℃"
                }
            ]
        }
    ],
    "time": 1.958
}
```


Example request

```
curl --request GET \
     --url {host}/1.0/lists/sensor/measure \
     --user '<username>:<password>' \
```

Device Data API

list device EUIs

GET {host}/1.0/lists/devices/eui

this is a quick method to get all device EUIs belonging to the organization

Query Parameters

offset number

current page number, start from 1, default = 1

length string

the number of rows in page, default = 20

response


```
{
    "code": "0",
    "data": {
        "gateway": [
            "2CF7F1........",
            "2CF7F1........",
            "2CF7F1........"
        ],
        "node": [
            "2CF7F1........",
            "2CF7F1........",
            "2CF7F1........",
            "2CF7F1........",
            "2CF7F1........"
        ]
    }
}
```

Example request

```
curl --request GET \
  --url {host}/1.0/lists/devices/eui \
  --user '<username>:<password>'
```

get the latest data of the device

GET {host}/1.0/devices/data/:node_eui/latest

A node may have multiple channels, each channel is a physical socket for a sensor to be connected. A sensor may produce multiple measurements. e.g. A temperature and humidity sensor will output temperature and humidity value. We call this temperature value a measurement, similarly we call humidity value another measurement.

This API will output the latest value of every measurement, from the sensor installed at specified channel. If the channel parameter is omitted, all sensors at all channels will be included. If a channel had installed different sensors before, the latest measurement of previous sensors will be included as well.

Path Parameters

node_eui (required)	string	Node EUI

Query Parameters

measure_id	string	sensor measurement ID, eg. measure_id=4097
channel	string	query data from this channel, if omitted, measurements of all channels will be returned

response

```
{
  "code": "0",
  "data": [
    {
      "channel": "1",                    // channel number
      "points": [                        // latest data points
        {
          "name": "Soil Temperature",    // the measurement name
          "unit": "℃",                   // the measurement unit
          "value": "13.75",              // the measurement valu
e
          "created": "1545013573"        // when the record is c
reated (timestamp, unit second)
        },
        {
          "name": "Soil Humidity",
          "unit": "%RH",
          "value": "19",
          "created": "1545013573"
        }
      ]
    }
  ]
}
```

Example request

```
curl --request GET \
     --url {host}/1.0/devices/data/:node_eui/latest?measure_i
d:measure_id&channel:channel \
     --user '<username>:<password>' \
     --include
```

get the history data of the device

GET {host}/1.0/devices/data/:node_eui/raw

Get the history data of a specified node.

The maximum number of data points returned per channel is 400. If the number of data points of a channel in the specified time range is larger than 400, the latest 400 data points will be returned. If the time range is omitted, the latest data will be returned.

Path Parameters

node_eui (required)	string	Node EUI

Query Parameters

measure_id	string	sensor measurement ID, eg. measure_id=4097
channel	string	query data from this channel, if omitted, measurements of all channels will be returned
limit	number	the number of records you want to query, max is 400
time_start	number	timestamp, unit millisecond
time_end	number	timestamp, unit millisecond

```
//response
/*
A data set is identified by channel and measurement ID, which m
eans that the same channel may have multiple data sets if it ha
d installed different sensors with different measurement ID.
e.g. Channel 1 had installed light sensor then carbon dioxide s
ensor, then there will be two data sets related with channel 1
- {"channel": "1", "name": "Light", "points": [...], ...} and
{"channel": "1", "name": "CO2", "points": [...], ...}.
*/
{
    "code": "0",
    "data": [
        {                                   // data set
            "channel": "1",                 // channale number
            "channel_name": "channel alias name",
            "name": "Air Temperature",      // measurement name
            "unit": "℃",                    // measurement unit
            "points": [                     // measurement data poi
nts
                {
                    "value": "8",                   // measuremen
t value
                    "created": "1545990300348"      // timestamp,
unit millisecond
                },
                {
                    "value": "7",
                    "created": "1545989996049"
                }
            ]
        },
        {
            "channel": "2",
            "channel_name": "the 2nd channel",
            "name": "Air Humidity",
            "unit": "%RH",
            "points": [
                {
                    "value": "59",
                    "created": "1545990300348"
                },
                {
                    "value": "68",
                    "created": "1545989996049"
```

```
                },
                {
                    "value": "68",
                    "created": "1545989996060"
                }
            ]
        }
    ],
    "time": 0.297
}
```


Example request

```
curl --request GET \
  --url {host}/1.0/devices/data/:node_eui/raw?limit=:limit&time_start=:time_start&time_end=:time_end \
  --user '<username>:<password>' \
  --include
```

get the segment data of the device

GET {host}/1.0/devices/data/:node_eui/segment

Segment a huge data set into small ones, then output the average value and peak value of each small segment. The small segment is described with time length, e.g. 60 minutes.

Path Parameters

node_eui (required)	string	Node EUI

Query Parameters

measure_id	string	sensor measurement ID, eg. measure_id=4097
channel	string	

query data from this channel, if omitted,
measurements of all channels will be returned

segment	string	
time length of segment, unit minute		
time_start	string	timestamp, unit millisecond
time_end	string	timestamp, unit millisecond

response

```
{
  "code": "0",
  "data": [
    {
      "channel": "1",           // channale number
      "channel_name": "channel alias name",
      "name": "Wind Direction", // measurement name
      "unit": "°",              // measurement unit
      "list": [                 // segment list
        {
          "created": "1545181200",  // timestamp, unit second
          "avg": "30",              // avgrage value of the seg
ment
          "peak": "90"              // peak value of the segmen
t
        },
        {
          "created": "1545182200",
          "avg": "31",
          "peak": "93"
        }
      ]
    }
  ]
}
```

Example request

```
curl --request GET \
  --url {host}/1.0/devices/data/:node_eui/segment?measure_id:me
asure_id&channel:channel&segment:segment&time_start:time_star
t&time_end:time_end \
  --user '<username>:<password>' \
  --include
```

Data OpenStream API Quickstart

Summary:
This guide will walk you through how to subscribe your devices' messages as well as how to send a command to a specific device, using Eclipse Mosquitto's CLIs to subscribe or publish messages.

Setup

- Install or download Mosquitto.

Credentials

Browse https://sensecap.seeed.cc , navigate to "User -> Organization -> Secret Credentials", create a full access "Access Key", set down it as , and also "Organization ID" as .

Receive Devices' Messages

Let's listen for all of your devices' messages.

1.Open a terminal window and execute the following command.

```
mosquitto_sub \
    -h openstream.api.sensecap.seeed.cc \
    -t '/device_sensor_data/<OrgID>/+/+/+/+' \
    -u 'org-<OrgID>' \
    -P '<Password>' \
    -I 'org-<OrgID>-quickstart' \
    -v
```

You should replace and with the ones you set down from "Credentials".

2.Power up devices, while devices keep sending messages, you should see outputs like:

```
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/2CF7F13XXXXXXXXX/41
05 {"value":0,"timestamp":1544151824139}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/2CF7F13XXXXXXXXX/40
97 {"value":23,"timestamp":1544151900992}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/2CF7F13XXXXXXXXX/41
01 {"value":101629,"timestamp":1544151901112}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/2CF7F13XXXXXXXXX/40
98 {"value":71,"timestamp":1544151900992}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/2CF7F13XXXXXXXXX/40
99 {"value":69.12,"timestamp":1544151902224}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/2CF7F13XXXXXXXXX/41
00 {"value":437,"timestamp":1544151922137}
```

It shows each data collected by sensors, with Device EUI , Device Channel, Sensor EUI, Measurement ID, Measurement Value, and timestamp, see the reference for more details.

Subscribe a Specific Field

You can also subscribe a specific field in the topic, so that you could get the data from a single device, or a single channel.

```
mosquitto_sub \
    -h openstream.api.sensecap.seeed.cc \
    -t '/device_sensor_data/<OrgID>/2CF7F12XXXXXXXXX/#' \
    -u 'org-<OrgID>' \
    -P '<Password>' \
    -I 'org-<OrgID>-quickstart' \
    -v
```

You should replace and with what you got from "Credentials", and replace 2CF7F12XXXXXXXXX with the Device EUI you own.

Congratulations! Now you know how to monitor and receive messages via MQTT. Go build something awesome!

Data OpenStream API Reference

The Connection Information

- Host: sensecap-openstream.seeed.cc
- Port: 1883 for MQTT, or 8083 for MQTT Over WebSocket
- ClientID: org-<Organization ID>-<Random ID>, replace <Orgnization ID> with you got from dashboard, and replace <Random ID> with you randomly generated Numbers and lowercase letters.
- Username: org-<Organization ID>, replace <Organization ID> with you got from dashboard.
- Password: Is your organization access key.

Publish And Subscribe Model

SenseCAP OpenStream API implements “Publish And Subscribe Model”, as the MQTT protocol does. You can connect your server to SenseCAP OpenStream API through MQTT or MQTT over WebSocket to communicate with the standard pub-sub protocol.

You can “publish” commands to platforms or devices and “subscribe” to receive messages. “subscribe” is the most common way to continuously monitor the telemetry data from devices.

Message Topic

Receive All Devices’ Telemeasuring Data

Topic: /device_sensor_data/<OrgID>/+/+/+/+

Receive Specified Device’s Telemeasuring Data

Topic Format:
/device_sensor_data/<OrgID>/<DeviceEUI>/<Channel>/<SensorEUI>/<MeasurementID>

Field	Description
OrgID	Your “Organization ID”, you can find this on https://sense-cap.seeed.cc. You own a unique Organization ID, and all the topics will need it.
DeviceEUI	The device EUI
Channel	A physical socket on the device for a sensor to be connected
SensorEUI	The sensor’s ID
MeasurementID	Please refer to “List of Measurement IDs” in this documentation

Note: “+” means that there is no filtering condition for this field, matching all possible configurations. So, “/+/+/+/+” means to listen for all “<DeviceEUI>”, “<Channel>”, “<SensorEUI>”, “<MeasurementID>”

Topic can specify filtering conditions to implement listening on specified devices, channels and telemeasuring data types. For example, you can only listen for Device whose device ID is “2F000000000000”, then you can replace the <DeviceEUI> field with 2F000000000000

The “2F000000000000” in this example must be a device that you have already bound to your account. And you should always remember to replace <OrgID> with your own “Organization ID”.

Message Body

New Telemeasuring Data

```
{
    "value": 437,
    "timestamp": 1544151922137
}
```

This is a sensor telemeasuring data message uploaded by a device, which conforms to the JSON format and can be parsed by JSON parser. In general, for most functional requirements, a body needs to be used in conjunction with some fields in the topic.

Field	Description
value	Sensor’s Measurement Value
timestamp	Time to measure data, unit millisecond

List of Sensor Types

Sensor ID	Sensor Name	Measurement IDs	Measurement Name
1011	SenseCap Rainfall Recorder Sensor	0x1011(4113)	Rainfall(hour)
100E	SenseCAP Soil VWC&EC&Temp Sensor	0x1010(4112), 0x100F(4111), 0x100E(4110)	Soil Temperature, Soil Electrical Conductivity, Soil Volumetric Water Content
100D	SenseCAP Dissolved Oxygen Sensor	0x100D(4109)	Dissolved Oxygen
100C	SenseCAP Electrical Conductivity Sensor	0x100C(4108)	Eletrical Conductivity
100B	SenseCAP Light quantum Sensor	0x100B(4107)	Light quantum
100A	SenseCAP Water PH Sensor	0x100A(4106)	Water PH
1009	SenseCAP Wind Speed Sensor	0x1009(4105)	Wind Speed
1008	SenseCAP Wind Direction Sensor	0x1008(4104)	Wind Direction
1006	SenseCAP Soil Temp&Humi Sensor	0x1007(4103), 0x1006(4102)	Soil Humidity, Soil Temperature
1005	SenseCAP Air Pressure Sensor	0x1005(4101)	Air Pressure
1004	SenseCAP CO2 Sensor	0x1004(4100)	CO2
1003	SenseCAP Light Sensor	0x1003(4099)	Light
1001	SenseCAP Air Temp&Humi Sensor	0x1002(4098), 0x1001(4097)	Air Humidity, Air Temperature

List of Measurement IDs

Measurement ID	Measurement Name	Value type	Value Range	Unit
0x1011(4113)	Rainfall(hour)	number	0~240	mm/hour
0x1010(4112)	Soil Temperaturex	number	-40~60	°C
0x100F(4111)	Soil Electrical Conductivity	number	0~23	ds/m
0x100E(4110)	Soil Volumetric Water Content	number	0~100	%
0x100D(4109)	Dissolved Oxygen	number	0~20	mg/L
0x100C(4108)	Eletrical Conductivity	number	0~23	dS/m²
0x100B(4107)	Light quantum	number	0~2000	umol/㎡s
0x100A(4106)	Water PH	number	0~14	PH
0x1009(4105)	Wind Speed	number	0~60	m/s
0x1008(4104)	Wind Direction	Enum	0,45,90,135,180,225,270,315	°
0x1007(4103)	Soil Humidity	number	0~100	%RH
0x1006(4102)	Air Pressure	number	-30~70	°C
0x1005(4101)	Soil Temperature	number	300~1100000	Pa
0x1004(4100)	CO2	number	400~10000	ppm
0x1003(4099)	Light	number	0~188000	Lux

Measurement ID	Measurement Name	Value type	Value Range	Unit
0x1002(4098)	Air Humidity	number	0~100	%RH
0x1001(4097)	Air Tempera-ture	number	-40~90	°C