Sensecap Open Api Document

Introduction

Overview

HTTP API

HTTP API Quickstart

HTTP API Access Guide

HTTP API Reference

Organization Management API

Group Management API

Device Management API

Device Data API

Data OpenStream API

Data OpenStream API Quickstart

Data OpenStream API Reference

SenseCAP SDK

Java SDK

Appendix

List of Sensor Types

List of Measurement IDs

List of Device Status IDs

List of Error Code

SenseCAP API Introduction

SenseCAP API is for users to manage IoT devices and data. It combines three types of API methods: HTTP protocol, MQTT protocol, and Websocket protocol.

- With HTTP API, users can manage LoRa and NB-IoT devices, to get RAW data or historical data.
- With MQTT API, users can subscribe to the sensor's real-time measurement data through the MQTT protocol.
- With Websocket API, users can get real-time measurement data of sensors through Websocket protocol.

HTTP API Quickstart

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

Prerequisite

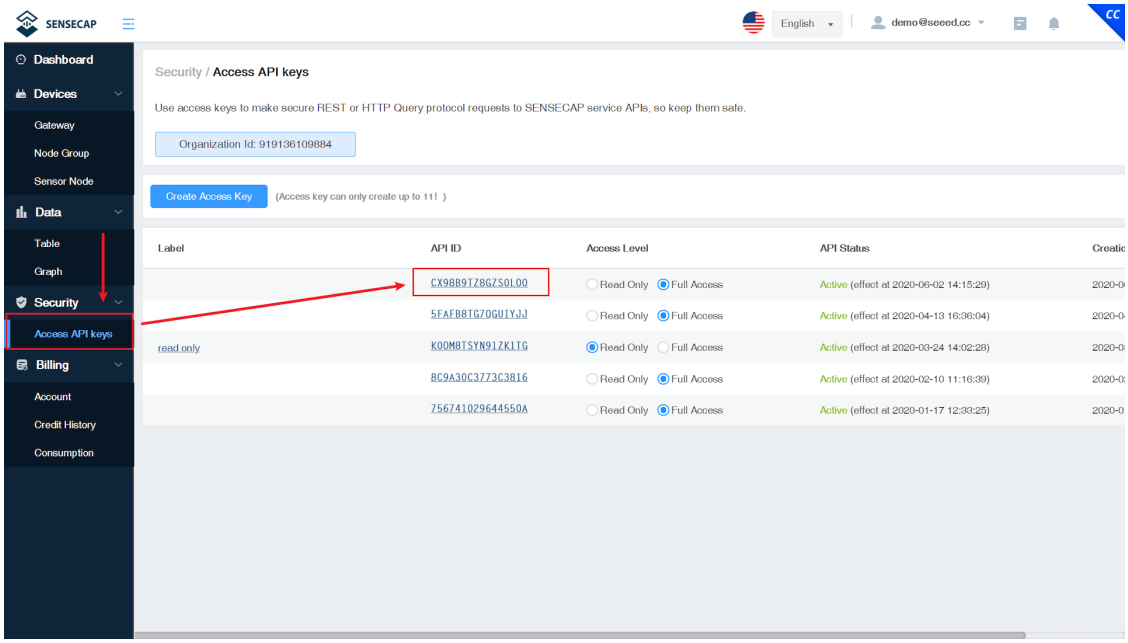
If you do not have an account, please register for the SenseCAP Portal.

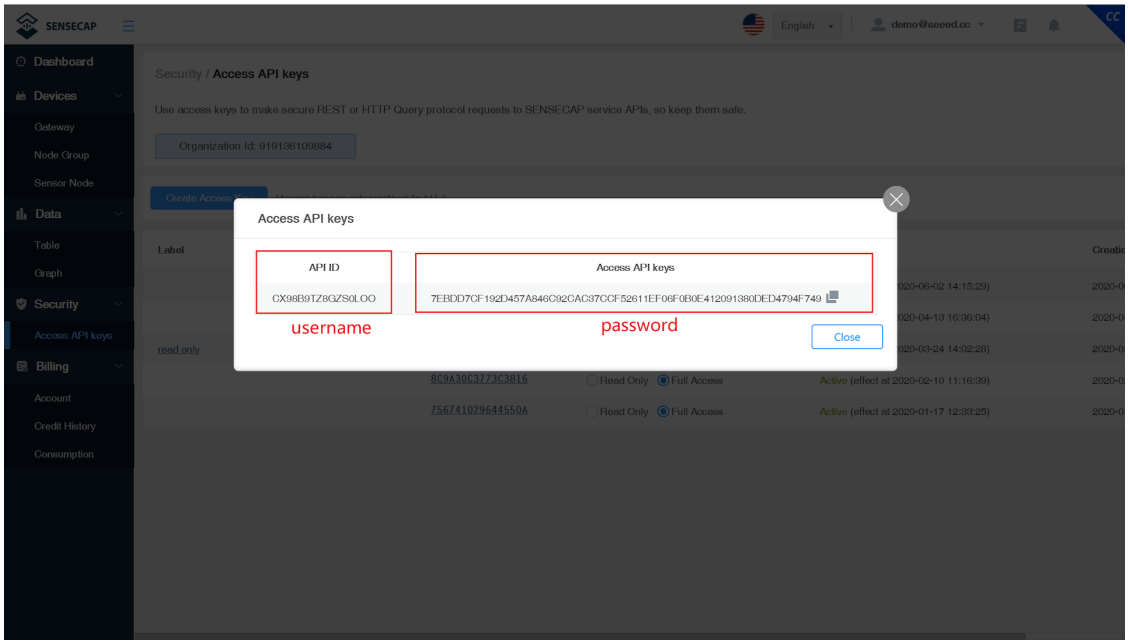
- China Stationhttps://sensecap.seeed.cn
- Global Stationhttps://sensecap.seeed.cc

> **Note:** LoRaWAN devices are used with Global Station

Get an Access Key

1. Login the SenseCAP Portal.
2. Navigate to "Security/Access API keys"
3. Click "Create Access Key"
4. Click "API ID", and get the "API ID" and "Access API keys" after entering the password.





Get all the Deivice Groups

Use curl to make an HTTP request.The following example calls the API to get all the Device Groups under the account.

- username = API ID
- password = Access API keys

```
curl --user "<username>:<password>" \
     https://sensecap.seeed.cc/openapi/list_groups
```

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

```
{
    "code": "0",
    "data": [
        {
            "group_name": "Default",
            "group_uuid": ""
        },
        {
            "group_name": "test group",
            "group_uuid": "80523B280630E611"
        },
        {
            "group_name": "demo",
            "group_uuid": "EBAD5387C4FC8711"
        }
    ]
}
```

HTTP API Access Guide

HTTP Request and Response

Requests are authenticated with the HTTP Basic Authentication.

HTTP HOST

- China Station: https://sensecap.seeed.cn/openapi
- Global Station: https://sensecap.seeed.cc/openapi

HTTP HEADER

Request

key	description
API-VERSION	api version

Response

key	description
api-gateway-excute-second	Time in seconds to execute the api
api-gateway-mpuo-consume	The quota consumed by executing the api

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.

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

API Response

All response key 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"
}
```

Organization Management API

Get organization information

GET {host}/view_organization

Get organization information

Response

Name	Description
org_id	organization id

Example request

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

{
    "code": "0",
    "data": {
        "org_id": "919136109886",
    }
}
```

Group Management API

Create device group

POST {host}/create_group

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

Request

Body Parameters

name (required)	string	custom group name

Response

Name	Description
group_name	Group name
group_uuid	Device group unique identification

Example request

```
curl --request POST \
     --url '{host}/create_group' \
     --user '<username>:<password>' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --data '{"name":"device group name"}' \
     --include

{
    "code": "0",
    "data": {
        "group_name": "test group",
        "group_uuid": "478ED9E2A290F401"
    }
}
```

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Update device group information

POST {host}/update_group

update the device group name

Request

Body Parameters

group_uuid (required)	string	the group unique identification
name (required)	string	the new name of device group

Example request

```
curl --request POST \
     --url '{host}/update_group' \
     --user '<username>:<password>' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --data '{"group_uuid":"259B1906C3B38481", "name":"new name of the device group"}' \
     --include

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

Delete a device group

POST {host}/delete_group

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

Request

Query Parameters

group_uuid (required)	string	the group unique identification

Example request

```
curl --request POST \
     --url '{host}/delete_group?group_uuid=0C32119F38C89C31' \
     --user '<username>:<password>' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --include

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

List device group

GET {host}/list_groups

get the list of all groups

Response

Name	Description
group_name	Group name
group_uuid	Device group unique identification

Example request

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

{
    "code": "0",
    "data": [
        {
            "group_name": "Default",
            "group_uuid": ""
        },
        {
            "group_name": "test group",
            "group_uuid": "80523B280630E611"
        },
        {
            "group_name": "demo",
            "group_uuid": "EBAD5387C4FC8711"
        }
    ]
}
```

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Move device to other group

DELETE {host}/move_devices_to_group

Request

Path Parameters

group_uuid (required)	string	the group unique identification which the devices will be moved to

Body Parameters

devices (required)	array	device EUI

Example request

```
curl --request POST \
     --url '{host}/move_devices_to_group' \
     --user '<username>:<password>' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --data '{"devices":["2CF7F12004100001","2CF7F12004100002"], "group_uuid": "0BC724D9F32AD934"}' \
     --include

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

Device Management API

Device list

GET {host}/list_devices

get the list of devices

Request

Query Parameters

device_type	string	device type：1-gateway， 2-node(default)
group_uuid	string	group unique identification

Response

Name	Description
device_eui	Device unique identification
device_name	Device name

Example request

```
curl --request GET \
     --url {host}/list_devices?device_type=1&group_uuid=0C32119F38C89C31 \
     --user '<username>:<password>'

{
    "code": "0",
    "data": [
        {
            "device_eui": "2CF7F12010700088",
            "device_name": "device2CF7F12010700088"
        },
        {
            "device_eui": "2CF7F1201070001C",
            "device_name": "device2CF7F1201070001C"
        },
        {
            "device_eui": "2CF7F12104700010",
            "device_name": "US915-2CF7F12104700010"
        }
    ]
}
```

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Get device detail

POST {host}/view_devices

get the detail of devices

Request

Body Parameters

device_euis (required)	array	device eui, up to 50 devices at a time
device_type	string	device type：1-gateway， 2-node(default)

Response

Name	Description
frequency	Equipment communication frequency
device_eui	Device unique identification
device_name	Device name
device_network	Networking protocols, 1:LoRaWAN，2：NB-IoT，3：2G , 4:LoRaPP
position	Device GPS location
position_source	GPS position source, 0- manually set position, 1- position reported by the device
hardware_version	Device hardware version number
software_version	Device software version number
sim	Sim card information on the device
iccid	ICCID
msisdn	MSISDN
activateTime	The activation date
expiryDate	Billing end date
status	Status, 0- unknown, 1- normal, 2- single stop, 3- stop, 4- pre-sale number, 5- sale number, 6- transfer, 7- sleep, 8- to be activated
flow	Current month used flow
residueFlow	The remaining flow

Example request

```
curl --request POST \
     --url {host}/view_devices \
     --data '{"device_type":1, "device_euis":["2CF7F15000100122"]}' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --user '<username>:<password>'
```


```
{
    "code": "0",
    "data": [
        {
            "frequency": "470",
            "device_eui": "2CF7F15000100122",
            "device_name": "设备2CF7F15000100122",
            "device_network": 2,
            "position": {
                "latitude": 113.931225,
                "longitude": 22.569792
            },
            "position_source": 0,
            "hardware_version": "",
            "software_version": "23.0",
            "sim": {
                "iccid": "89860446091891237424",
                "msisdn": "1440467057424",
                "activateTime": "2019-12-03T00:00:00.000Z",
                "expiryDate": "2020-11-30T00:00:00.000Z",
                "status": 1,
                "flow": 0,
                "residueFlow": 1024
            }
        }
    ]
}
```


History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Get device channels

POST {host}/list_device_channels

Request

Body Parameters

device_euis (required)	array	device eui,up to 50 devices at a time

Response

Name	Description
device_eui	Device unique identification
channel_index	The channel number
sensor_id	The sensor id
sensor_status	Sensor status :0- idle 1- normal 2- abnormal

Name	Description
channel_type	Channel type,1: 485 Sensor; 2: Seeed Sensor; 3:485 Output; 4: Seeed Output
sensor_type	Sensor type
channel_name	The name of the channel
measurement_ids	Measured value id

Example request

```
curl --request POST \
     --url {host}/list_device_channels \
     --data '{"device_euis":["2CF7F15000100147","2CF7F16221200060"]}' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --user '<username>:<password>'


{
    "code": "0",
    "data": [
        {
            "device_eui": "2CF7F15000100147",
            "channels": [
                {
                    "channel_index": 1,
                    "sensor_id": "2CF7F13011900006",
                    "sensor_status": 1,
                    "channel_type": 2,
                    "sensor_type": "1005",
                    "channel_name": "",
                    "measurement_ids": [
                        "4101"
                    ]
                }
            ]
        },
        {
            "device_eui": "2CF7F16221200060",
            "channels": [
                {
                    "channel_index": 11,
                    "sensor_id": "0111006221200060",
                    "sensor_status": 1,
                    "channel_type": 1,
                    "sensor_type": "2001",
                    "channel_name": "",
                    "measurement_ids": [
                        "4097",
                        "4105"
                    ]
                }
            ]
        }
    ]
}
```

Get device running status

POST {host}/view_device_running_status

Request

Body Parameters

device_euis (required)	array	device eui,up to 50 devices at a time

Response

Name	Description
device_eui	Device unique identification
latest_message_time	The last time the device reported a message
online_status	Online status :0- offline, 1- online
battery_status	Battery state :0- low battery 1- good battery
report_frequency	If the device reports frequency per minute and returns -1, the device fails to report this information

Example request

```
curl --request POST \
     --url {host}/view_device_running_status \
     --data '{"device_euis":["2CF7F1101300001C","2CF7F16221200060"]}' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --user '<username>:<password>'

{
    "code": "0",
    "data": [
        {
            "device_eui": "2CF7F16221200060",
            "latest_message_time": "2020-04-20T07:06:32.944Z",
            "online_status": 0,
            "battery_status": 1,
            "report_frequency": 0
        },
        {
            "device_eui": "2CF7F1101300001C",
            "latest_message_time": "",
            "online_status": 0,
            "battery_status": 1,
            "report_frequency": -1
        }
    ]
}
```

Sensor measure list

GET https://sensecap-statics.seeed.cn/refer/def/sensor.json

Get a list of the physical measurements of all the sensors. The reference list of sensor measurements is stored in the cloud as a file that you can access directly

Response

Name	Description
zh-cn	Chinese description
en	English description

Name	Description
sm	The measurement type ID corresponding to each sensor type is recorded
rg	Value range of measured value
sensorType	Sensor type, recording the name of each sensor type
measurementId	Measurement id, which records the name and unit of each measurement

Example

```
{
 "zh-cn": {
  "sensorType": {
   "1001": "空气温湿度传感器",
   "1003": "光照强度传感器",
   ...
  },
  "measurementId": {
   "4097": ["空气温度", "℃"],
   "4098": ["空气湿度", "%RH"],
   ...
  }
 },
 "en": {
  "sensorType": {
   "1001": "Air Temperature and Humidity Sensor",
   "1003": "Light Intensity Sensor",
   ...
  },
  "measurementId": {
   "4097": ["Air Temperature", "℃"],
   "4098": ["Air Humidity", "%RH"],
   ...
  }
 },
 "sm": {
  "1001": ["4098", "4097"],
  "1003": ["4099"],
  ...
 },
 "rg": {
  "4097": "-40~90",
  "4098": "0~100",
  ...
 }
}
```

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Bind device

POST {host}/device/bind

bind device to account

Request

Body Parameters

eui (required)	string	device eui
code (required)	string	device code
device_name	string	device name
group_uuid	string	group uuid,available through the group list interface
longitude	string	device positon, longitude
latitude	string	device postion,latitude

Example request

```
curl --request POST \
     --url '{host}/bind_device' \
     --user '<username>:<password>' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --data '{"code":"device code","eui":"device eui"}' \
     --include


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

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Unbind device

POST {host}/delete_devices

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

Request

Body Parameters

device_euis (required)	array	device eui,up to 50 devices at a time

Example request

```
curl --request POST \
     --url {host}/delete_devices \
     --data '{"device_euis":["2CF7F15000100122"]}' \
     --header 'Content-Type: application/json; charset=utf-8' \
     --user '<username>:<password>'

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

Device Data API

Get the latest data of the device

GET {host}/view_latest_telemetry_data

Returns the latest telemetry data for the device within a year.If channel_index is not specified,return the latest data point under each channel of the device.

Request

Query Parameters

device_eui (required)	string	device eui
channel_index	string	query data from this channel
measurement_id	string	sensor measurement ID

Response

Name	Description
channel_index	The channel number
measurement_value	Measured value
measurement_id	Measured value id
time	Measured time

Example request

```
curl --request GET \
     --url {host}/view_latest_telemetry_data?device_eui=2CF7F12010700001&measurement_id=4104&channel_ind
ex=32 \
     --user '<username>:<password>' \
     --include

{
    "code": "0",
    "data": [
        {
            "channel_index": 32,
            "points": [
                {
                    "measurement_value": 185,
                    "measurement_id": "4104",
                    "time": "2020-03-26T07:28:28.575Z"
                }
            ]
        }
    ]
}
```

History version

version	description

version	description
Sensecap V1	Maintenance stopped, not recommended

Get the history data of the device

GET {host}/list_telemetry_data

To obtain the historical data of the specified sensor node device, the data returned for a maximum of one month can only be queried for the last three months

Request

Query Parameters

device_eui (required)	string	device eui
channel_index	string	query data from this channel
measurement_id	string	sensor measurement ID
limit	number	the number of records you want to query
time_start	number	timestamp, unit millisecond, the default date is one day ago
time_end	number	timestamp, unit millisecond,Default current time

Response

Returns the representation of the format sampled column data

Name	Description
list[0]	The channel number and measured value id
list[1]	Measured value，contains the measured value and the measured time

Example request

```
curl --request GET \
  --url {host}/list_telemetry_data?device_eui=2CF7F12210400082 \
  --user '<username>:<password>' \
  --include


{
    "code": "0",
    "data": {
        "list": [
            [[1, "4097"], [1, "4098"], [2, "4097"]],
            [
                [ [22, "2020-03-06T23:48:00Z"], [23, "2020-03-06T23:48:00Z"], [24, "2020-03-06T23:48:00
Z"]]
                [ [22, "2020-03-06T23:48:00Z"], [23, "2020-03-06T23:48:00Z"] ]
                [ [22, "2020-03-06T23:48:00Z"], [23, "2020-03-06T23:48:00Z"] ]
            ]

        ]
    }
}
```

The following example returns the following:

- list[0][0]：Represents the channel and the measured value id.In this example, the channel is 1 and the measured value id is 4097
- list[1][0]：Corresponding to the measurement results of list[0][0]
- list[1][1]：Corresponding to the measurement results of list[0][1]
- list[1][0][0]：In this example, the first measurement result corresponding to the measurement value id of channel 1 is 4097, the measurement value is 22, and the measurement time is 2020-03-06 t23:48:00Z

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Get the segment data of the device

GET {host}/aggregate_chart_points

Divide the large data segment into small data segments, then output the average value of each segment. The return data is up to one year and 250 points each measurement at most.If more than 250 points, it will automatically re-divide the time period to return 250 points.

Request

Query Parameters

device_eui (required)	string	device eui
channel_index	string	sensor measurement ID
measurement_id	string	query data from this channel
interval	string	The length of the time period to get, unit minute. 60 minutes for default

time_start	number	timestamp, unit millisecond, the default date is one day ago
time_end	number	timestamp, unit millisecond, Default current time

Response

Returns the representation of the format sampled column data

Name	Description
channel	The channel number
average_value	Measured average value
measurement_id	Measured value id
time	Measured time

Example request

```
curl --request GET \
  --url {host}/aggregate_chart_points?device_eui=2CF7F12210400082 \
  --user '<username>:<password>' \
  --include

{
    "code": "0",
    "data": [
        {
            "channel": 1,
            "lists": [
                {
                    "average_value": 7.27,
                    "measurement_id": "4106",
                    "time": "2019-05-25T23:42:00.000Z"
                },
                {
                    "average_value": 6.85,
                    "measurement_id": "4106",
                    "time": "2019-05-27T10:45:00.000Z"
                }
            ]
        }
    ]
}
```

History version

version	description
Sensecap V1	Maintenance stopped, not recommended

Data OpenStream API Quickstart

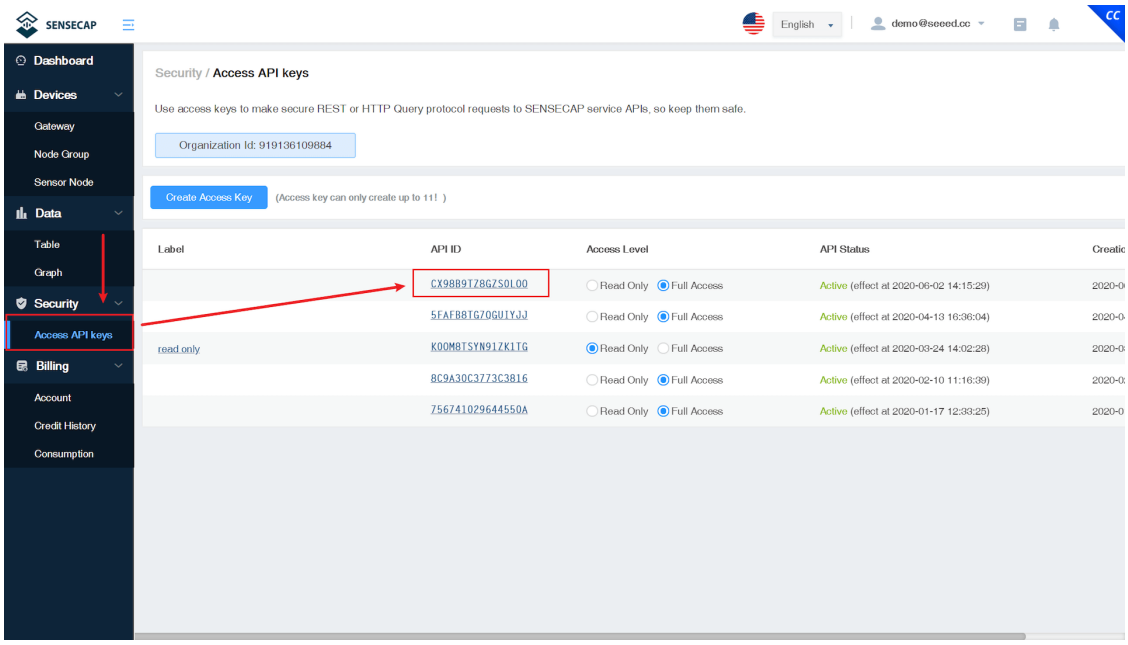
Summary:
This guide will walk you through how to subscribe your devices' messages as well as how to send a command to a specific device, using Eclipse Mosquitto's CLIs to subscribe or publish messages.

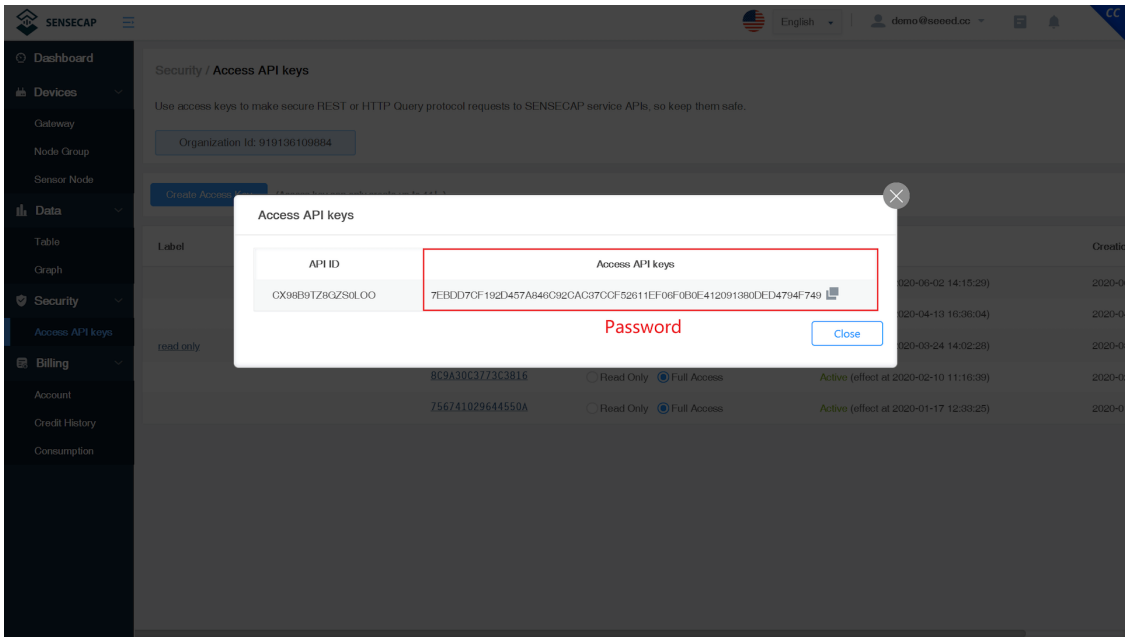
Setup

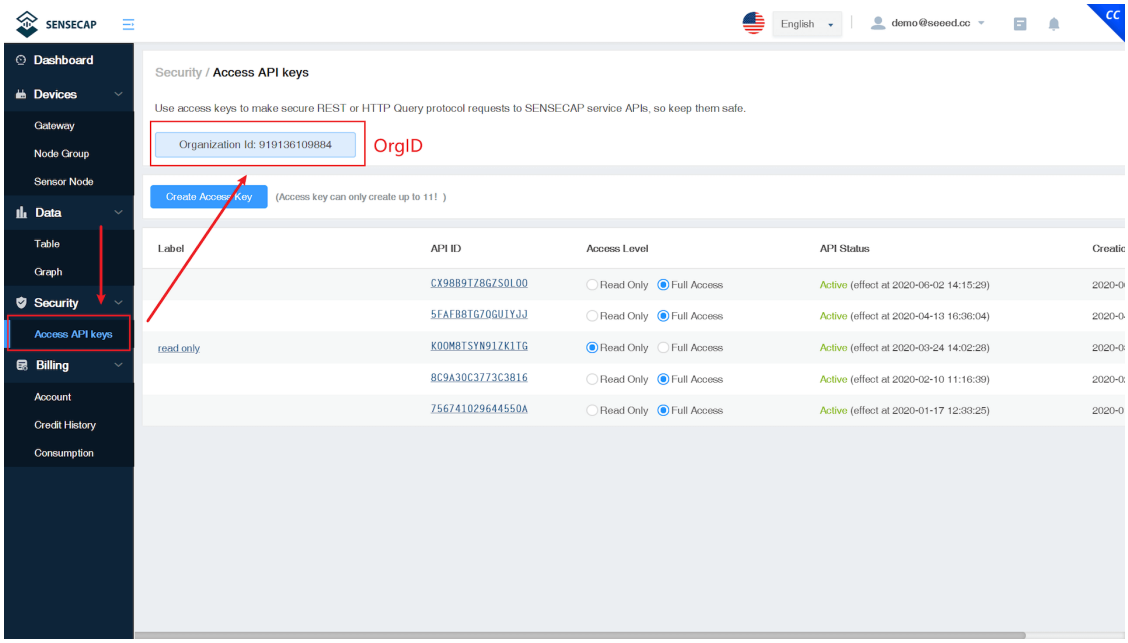
- Install or download Mosquitto.

Credentials

Browse SenseCAP Portal, navigate to "Security/Access API keys", click the "Create Access Key", and you can get the "Access API keys", set down it as <Password>, and also "Organization ID" as <OrgID>.







Receive Devices' Messages

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

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

- OrgID = Organization ID
- Password = Access API keys

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

Please replace the Organization ID and Access API Key you just obtained with the <OrgID> and <Password> above.

2.Power up devices, while devices keep sending messages, you should receive the data like:

```
/device_sensor_data/1234/2CF7F12000000001/1/vs/4105 {"value":2,"timestamp":1544151824139}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/vs/4097 {"value":23,"timestamp":1544151900992}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/vs/4101 {"value":101629,"timestamp":1544151901112}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/vs/4098 {"value":71,"timestamp":1544151900992}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/vs/4099 {"value":69.12,"timestamp":1544151902224}
/device_sensor_data/xxxx/2CF7F12XXXXXXXXX/1/vs/4100 {"value":437,"timestamp":1544151922137}
```

example	field	description
1234	OrgId	Organization ID
2CF7F12000000001	DeviceEUI	Unique identification of device
1	Channel	A physical socket on the device for a sensor to be connected
vs	Reserved	The reserved field
4105	MeasureID	The type of measurement, 4105 is the Wind Speed
2	value	Collected measurements, the Wind Speed is 2m/s
1544151824139	timestamp	The collection timestamp of the data

Subscribe a Specific Key

Specifying a specific key enables you to subscribe to data for a particular device or channel.

Example:
Subscribe to the temperature value collected by the Air Temperature and Humidity Sensor (DeviceEUI: 2CF7F12210400083;Channel: 1;).The temperature measurement ID is 4097.
Replace <OrgID> as Organization ID, <Password> as Access API Key, execute the command:

```
mosquitto_sub \
    -h sensecap-openstream.seeed.cn \
    -t '/device_sensor_data/<OrgID>/2CF7F12210400083/1/vs/4097' \
    -u 'org-<OrgID>' \
    -P '<Password>' \
    -I 'org-<OrgID>-quickstart' \
    -v
```

Received the data:

```
/device_sensor_data/521853156991/2CF7F12210400083/1/vs/4097 {"value":28,"timestamp":1561373812474}
```

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

Data OpenStream API Reference

The Connection Information

- Host: China Station: sensecap-openstream.seeed.cn Global Station: 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 SenseCAP Portal, and replace <Random ID> with you randomly generated Numbers and lowercase letters.
- Username: org-<Organization ID>, replace <Organization ID> with you got from dashboard (refer to the quickstart).
- Password: Get Access API keys on your SenseCAP Portal “security /API Access Key” (refer to the quickstart).

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 “subscribe” to receive messages. “subscribe” is the most common way to continuously monitor the telemetry data from devices.

Message Topic

Receive Device’s Telemeasuring Data

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

Field	Description
OrgID	Your “Organization ID”, you can find this on SenseCAP Portal. You own a unique Organization ID, and all the topics will need it.
DeviceEUI	Unique identification of device
Channel	A physical socket on the device for a sensor to be connected
Reserved	Reserved
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 measurement 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

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

This is a sensor measurement data 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	The collection timestamp of the data, unit millisecond

Receive Device's Status Data

Topic Format: /device_status_event/<OrgID>/<DeviceEUI>/<Reserved>/<StatusID>

Field	Description
OrgID	Your "Organization ID", you can find this on SenseCAP Portal. You own a unique Organization ID, and all the topics will need it.
DeviceEUI	Unique identification of device
Reserved	Reserved
StatusID	Please refer to "List of Device Status IDs" in this documentation

Subscribe to the required StatusID according to the list of device state IDs to avoid subscribing to unexpected IDs

Message Body

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

Field	Description
value	Sensor's Status Value
timestamp	The collection timestamp of the data, unit millisecond

Sensecap Java SDK Quickstart

Summary:
This guide will guide you how to use Sensecap Java SDK

Overview

Prerequisite

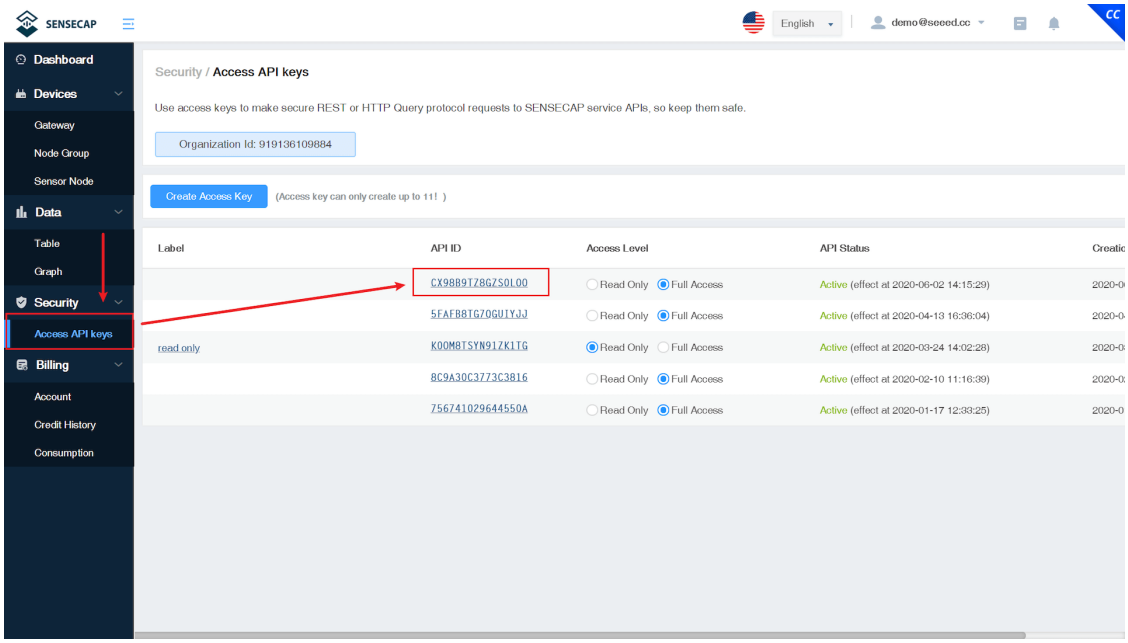
If you do not have an account, please register for the SenseCAP Portal.

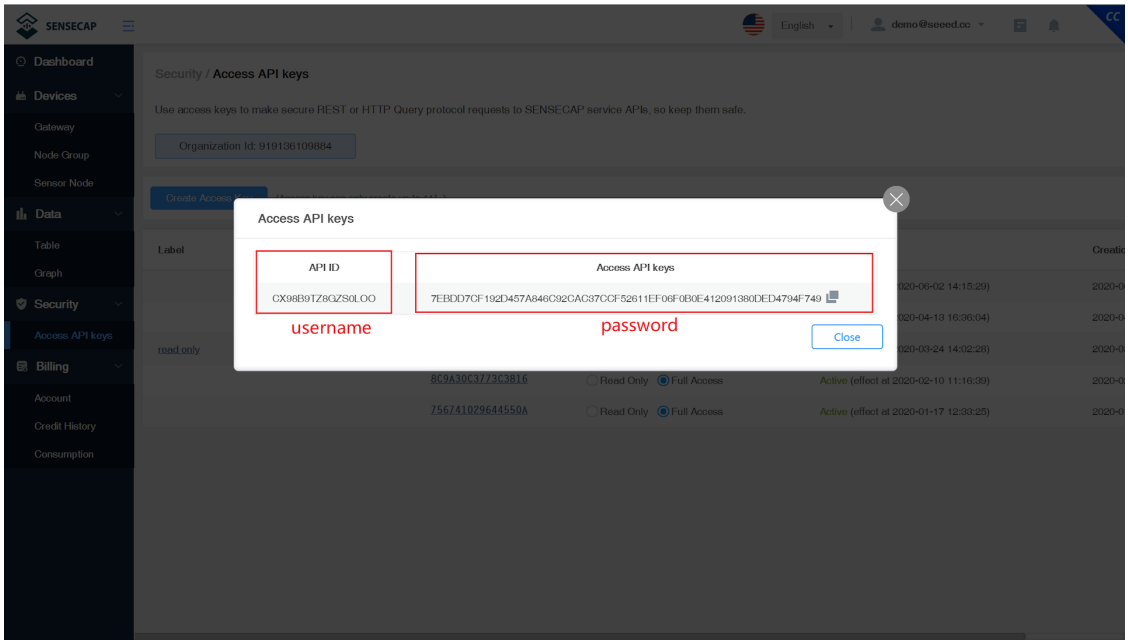
- China Stationhttps://sensecap.seeed.cn
- Global Stationhttps://sensecap.seeed.cc

> **Note:** LoRaWAN devices are used with Global Station

Get an Access Key

1. Login the SenseCAP Portal.
2. Navigate to “Security/Access API keys”
3. Click “Create Access Key”
4. Click “API ID”, and get the “API ID” and “Access API keys” after entering the password.





Maven coordinates

```
<dependency>
    <groupId>cc.seeed.sensecap-sdk</groupId>
    <artifactId>sensecap-java-sdk</artifactId>
    <version>1.0-RELEASE</version>
</dependency>
```

SenseCAPClient initialization

```
//client Configure
{
    String accessId = " ";
    String accessKey = "";
    int region = RegionType.SENSECAP_CC.getRegion();
  //SenseCAPClient senseCAPClient = new SenseCAPClientBuilder().buildConfig(accessId, accessKey, region)
;
    OpenApiConfig openApiConfig = new OpenApiConfig(accessId, accessKey, region);
    SenseCAPClient senseCAPClient = new SenseCAPClientBuilder().buildConfig(openApiConfig);
}
```

Organization

Get organizationId

```
public long getOrganizationId() throws BaseException;
```

Request parameters

None

Return values

Type	Description
long	organizationId

Request Example

```
public static void getOrganizationId() throws BaseException{
    long organizationId = senseCAPClient.getOrganizationManager()
            .getOrganizationId();
    System.out.println(organizationId);
}
```

Group

Create Group

```
public GroupInfo create(String groupName) throws BaseException;
```

Request parameters

Name	Type	Description
groupName	String	Group name

Return values

Type	Description
GroupInfo	Group information

Request Example

```
public static void createGroup() throws BaseException {
    GroupInfo result = senseCAPClient.getGroupManager().create("SDk-TEST");
    System.out.println(result);
}
```

Rename group

```
public void rename(String groupUUID, String groupName) throws BaseException;
```

Request parameters

Name	Type	Description
groupUUID	String	GroupUUID

Name	Type	Description
groupName	String	Group name

Return values

None

Request Example

```
public static void renameGroup() throws BaseException {
        senseCAPClient.getGroupManager().rename("groupUUID", "SDk-TEST");
    }
```

Remove group

```
public void remove(String groupUUID) throws BaseException;
```

Request parameters

Name	Type	Description
groupUUID	String	GroupUUID

Return values

None

Request Example

```
public static void renameGroup() throws BaseException {
        senseCAPClient.getGroupManager().remove("GroupUUID");
    }
```

Get all group

```
public List<GroupInfo> getGroupList() throws BaseException;
```

Request parameters

None

Return values

Type	Description
list	Group information list

Request Example

```
public static void getGroupList() throws BaseException {
    GroupResult groupResult = senseCAPClient.getGroupManager().createGroupQuery()
            .build()
            .execute();
    List<GroupInfo> groupInfos = groupResult.toList();
    System.out.println(groupInfos.toString());
}
```

Device

Move device

```
public  void moveDevices(String groupUUID, List<String> deviceEuis) throws BaseException;
```

Request parameters

Name	Type	Description
groupUUID	String	GroupUUID
deviceEuis	array	Device eui list

Return values

None

Request Example

```
public static void moveDeivces() throws BaseException {
    List<String> deviceEuis = Lists.newArrayList();
    deviceEuis.add(deviceEui);
    senseCAPClient.getDeviceManager().moveDevices("groupUUID", deviceEuis);
}
```

Get all devices under the group

```
public List<DeviceBaseInfo> getDeviceList(int deviceType, String groupUUID) throws BaseException;
```

Request parameters

Name	Type	Description
groupUUID	String	GroupUUID
deviceType	int	Device type 1：gateway ,2：node(default)

Return values

Type	Description
list	device base information list

Request Example

```
public static void getDeviceList() throws BaseException {
    DeviceResult deviceResult = senseCAPClient.getDeviceManager().createDeviceQuery()
            .deviceType(2)
            .groupUUID("")
            .build()
            .execute();
    List list = deviceResult.deviceList();
    System.out.println(list.toString());
}
```

Get device information

```
public List<DeviceInfo> getDeviceInfo(List<String> deviceEuis, int deviceType)  throws BaseException;
```

Request parameters

Name	Type	Description
deviceEuis	String	Device eui list
deviceType	int	Device type 1：gateway ,2：node(default)

Return values

Type	Description
list	Device information list

Request Example

```
public static void getDeviceInfoList() throws BaseException {
    List<String> deviceEuis = Lists.newArrayList();
    deviceEuis.add(deviceEui);
    DeviceResult deviceResult = senseCAPClient.getDeviceManager().createDeviceQuery()
            .deviceEuis(deviceEuis)
            .deviceType(2)
            .build()
            .execute();
    List<DeviceInfo> deviceInfos = deviceResult.deviceInfos();
    System.out.println(deviceInfos.toString());

}
```

Get device channel information

```
public List<DeviceChannelInfo> getDeviceChannelList(List<String> deviceEuis)  throws BaseException;
```

Request parameters

Name	Type	Description

Name	Type	Description
deviceEuis	String	device eui list

Return values

Type	Description
list	Device channel information list

Request Example

```
public static void getDeviceChannelList() throws BaseException {
    List<String> deviceEuis = Lists.newArrayList();
    deviceEuis.add(deviceEui);
    DeviceResult deviceResult = senseCAPClient.getDeviceManager().createDeviceQuery()
            .deviceEuis(deviceEuis)
            .build()
            .execute();
    List<DeviceChannelInfo> deviceChannelInfos = deviceResult.channelList();
    System.out.println(deviceChannelInfos.toString());
}
```

Get device running status

```
public List<DeviceStatusInfo> getDeviceRunningStatusList(List<String> deviceEuis)  throws BaseException;
```

Request parameters

Name	Type	Description
deviceEuis	String	device eui list

Return values

Type	Description
list	device running status list

Request Example

```
public static void getDeviceRunningStatusList() throws BaseException {
    List<String> deviceEuis = Lists.newArrayList();
    deviceEuis.add(deviceEui);
    DeviceResult deviceResult = senseCAPClient.getDeviceManager().createDeviceQuery()
            .deviceEuis(deviceEuis)
            .build()
            .execute();
    List<DeviceStatusInfo> deviceStatusInfos = deviceResult.runningStatusList();
    System.out.println(deviceStatusInfos);
}
```

Bind device

```
public boolean bindDevice(String eui, String code, String deviceName, String groupUUID, String longitude
, String latitude) throws BaseException;
```

Request parameters

Name	Type	Description
eui	String	device eui
code	String	device code
deviceName	String	device name
groupUUID	String	groupUUID
longitude	String	longitude
latitude	String	latitude

Return values

None

Request Example

```
public static void bindDevice() throws BaseException {
    senseCAPClient.getDeviceManager().createBinder()
            .eui("")
            .code("")
            .deviceName("")
            .groupUUID("")
            .longitude("")
            .latitude("")
            .build()
            .bind();
}
```

Delete device

```
public boolean deleteDevices(List<String> deviceEuis) throws BaseException;
```

Request parameters

Name	Type	Description
deviceEuis	String	device eui list

Return values

Type	Description
boolean	success or not

Request Example

Request Example

```
public static void deleteDevices() throws BaseException {
    List<String> deviceEuis = Lists.newArrayList();
    deviceEuis.add(deviceEui);
    boolean b = senseCAPClient.getDeviceManager().deleteDevices(deviceEuis);
}
```

Telemetry data

Get latest telemetry data

```
List<LatestTelemetryDataInfo> getLatestTelemetryData(String deviceEui, int channelIndex, int measurement
Id) throws BaseException;
```

Request parameters

Name	Type	Description
deviceEui	String	device eui
channelIndex	int	channel
measurementId	int	measurement id

Return values

Type	Description
list	latest telemetry data list

Request Example

```
public static void getLatestTelemetryData() throws BaseException {
    DataResult dataResult = senseCAPClient.getDataManager().createDataQuery()
            .deviceEui(deviceEui)
            .channelIndex(1)
            .measurementId(4099)
            .build()
            .execute();
    List<LatestTelemetryDataInfo> latestTelemetryDataInfos = dataResult.latestData();
    System.out.println(latestTelemetryDataInfos.toString());
}
```

Get historical telemetry data

```
List<TelemetryDataInfo> getTelemetryDataListCallback(String deviceEui, int channelIndex, int measurement
Id, int limit, long startTime, long endTime, TelemetryDataCallback var) throws BaseException;
```

Request parameters

Name	Type	Description
deviceEui	String	device eui
channelIndex	int	channel
measurementId	int	measurement id
limit	int	Number of queries
startTime	long	start time
endTime	long	end time
var	TelemetryDataCallback	Callback function

Parameter description

- [startTime>0,endTime>0] Get the historical data of the specified time range [-30d,now]
- (startTime=0,endTime>0] Get the default one day old to endTime historical data [-1d,endTime]
- [startTime>0,endTime=0) Get the historical data of the start time and subscribe to the latest news [startTime,–]
- (startTime=0,endTime=0) Subscribe to the latest news only [now,–]
- Service restart requires user to handle reconnection subscription by self

Return values

Type	Description
list	historical telemetry data list

Request Example

```
public static void getTelemetryDataListCallBack() throws BaseException {
        DataResult dataResult = senseCAPClient.getDataManager().createDataQuery()
            .deviceEui(deviceEui)
            .channelIndex(1)
            //.measurementId(4100)
            .startTime(1605385871000L)
            //.endTime(1605496271000L)
            .limit(100)
            .callback(new TelemetryDataCallback() {
                @Override
                public void messageArrived(TelemetryDataResult var) throws BaseException {
                    System.out.println("Test get mqtt messages:  " + var.toString());
                }
            })
            .build()
            .execute();
    List<TelemetryDataInfo> list = dataResult.dataList();
    System.out.println(JSON.toJSONString(list));


}
```

Get device Line chart data(openAPI)

```
List<ChartPointDataInfo> getChartPointData(String deviceEui, int channelIndex, int measurementId, int in
terval, long startTime, long endTime) throws BaseException;
```

Explain

- Divide the large data segment into small data segments, then output the average value of each segment.
- The return data is up to one year and 250 points each measurement at most.If more than 250 points, it will automatically re-divide the time period to return 250 points.

Request parameters

Name	Type	Description
deviceEui	String	device eui
channelIndex	int	channel
measurementId	int	measurement id
interval	int	Interval minutes
startTime	long	start time Timestamp MS
endTime	long	end time Timestamp MS

Return values

Type	Description
list	device Line chart data list

Request Example

```
public static void getLatestTelemetryData() throws BaseException {
    DataResult dataResult = senseCAPClient.getDataManager().createDataQuery()
            .startTime(1605600000000L)
            .deviceEui(deviceEui)
            .build()
            .execute();
    List<ChartPointDataInfo> chartPointDataInfos = dataResult.chartPointData();
    System.out.println(JSON.toJSONString(chartPointDataInfos));
}
```

Get device Line chart data

```
List<ChartDataListInfo> getChartDataList(String deviceEui, int channelIndex, int measurementId, int inte
rval, long startTime, long endTime) throws BaseException;
```

Explain

- Return all data points within one day (inclusive)
- If the time is greater than one day, the average value of data points in the **interval** range is returned
- **interval** should not be less than the minimum interval of data reporting

Request parameters

Name	Type	Description
deviceEui	String	device eui
channelIndex	int	channel

Name	Type	Description
measurementId	int	measurement id
interval	int	Interval minutes
startTime	long	start time Timestamp MS
endTime	long	end time Timestamp MS

Return values

Type	Description
list	device Line chart data list

Request Example

```
public static void getLatestTelemetryData() throws BaseException {
    DataResult dataResult1 = senseCAPClient.getDataManager().createDataQuery()
            .startTime(1605600000000L)
            .deviceEui(deviceEui)
            .interval(30)
            .build()
            .execute();
    List<ChartDataListInfo> listInfoList = dataResult1.chartData();
    System.out.println(JSON.toJSONString(listInfoList));
}
```

List of Sensor Types

Sensor ID	Sensor Name	Measurement IDs	Measurement Name
1001	Air Temperature and Humidity Sensor	4098, 4097	Air Humidity, Air Temperature
1003	Light Intensity Sensor	4099	Light Intensity
1004	CO2 Sensor	4100	CO2
1005	Barometric Pressure Sensor	4101	Barometric Pressure
1006	Soil Moisture and Temperature Sensor	4102, 4103	Soil Temperature, Soil Moisture
1008	Wind Direction Sensor	4104	Wind Direction
1009	Wind Speed Sensor	4105	Wind Speed
1011	Rain Gauge	4113	Rainfall Hourly
1013	Ultrasonic Distance Sensor	4115	Distance
1014	Water Leak Detector	4116	Water Leak
1015	Liguid Level Sensor	4117	Liguid Level
2001	Compact Weather Station 5 in 1 (S500)	4097, 4098, 4101, 4104, 4105	Air Temperature, Air Humidity, Barometric Pressure, Wind Direction, Wind Speed
2002	Compact Weather Station 3 in 1 (S300)	4097, 4101, 4098	Air Temperature, Barometric Pressure, Air Humidity
2003	Compact Weather Station 4 in 1 (S400)	4097, 4098, 4101, 4099	Air Temperature, Air Humidity, Barometric Pressure, Light Intensity
2004	NH3, Temperature & Humidity Sensor	4097, 4098, 4118	Air Temperature, Air Humidity, NH3
2005	H2S, Temperature & Humidity Sensor	4097, 4098, 4119	Air Temperature, Air Humidity, H2S
2007	Soil Temperature and VWC Sensor	4110, 4102	Soil Volumetric Water Content, Soil Temperature
2009	Turbine Flowmeter Sensor	4121, 4120	Total Flow, Flow Rate
2010	Sunshine Duration Sensor	4126	Sunshine Duration
2011	Total Solar Radiation Sensor	4127	Total Solar Radiation
2012	Evaporation Sensor	4128	Water Surface Evaporation
2013	PAR Sensor	4129	Photosynthetically Active Radiation(PAR)
2015	Soil Heat Flux Sensor	4125	Soil Heat Flux

2016	Soil Tension Meter	4133	Soil Tension
2017	Liquid EC and TDS Sensor	4135, 4134, 4124, 4123	TDS, Salinity, Water Temperature, Water Eletrical Conductivity
2018	Leaf Wetness and Temperature Sensor	4137, 4136	Leaf Wetness, Leaf Temperature
2019	Multilayer Soil Moisture and Temperature Sensor	4144, 4138, 4139, 4140, 4141, 4142, 4143, 4145	Soil Temperature-30cm, Soil Moisture-10cm, Soil Moisture-20cm, Soil Moisture-30cm, Soil Moisture-40cm, Soil Temperature-10cm, Soil Temperature-20cm, Soil Temperature-40cm
201A	Compact Weather Station 9 in 1 (S900)	4147, 4146, 4097, 4098, 4099, 4101, 4104, 4105, 4113	PM10, PM2.5, Air Temperature, Air Humidity, Light Intensity, Barometric Pressure, Wind Direction, Wind Speed, Rainfall Hourly
100A	PH Sensor	4106	pH
100B	PAR Sensor	4107	Light Quantum
100C	EC Sensor	4108	Electrical Conductivity
100D	Dissolved Oxygen Sensor	4109	Dissolved Oxygen
100E	Soil Temperature, VWC & EC Sensor	4108, 4102, 4110	Electrical Conductivity, Soil Temperature, Soil Volumetric Water Content
200A	Compact Weather Station 7 in 1 (S700)	4098, 4099, 4101, 4104, 4105, 4113, 4097	Air Humidity, Light Intensity, Barometric Pressure, Wind Direction, Wind Speed, Rainfall Hourly, Air Temperature
200D	Oxygen Sensor	4122	Oxygen Concentration
200E	Water EC Sensor	4123	Water Eletrical Conductivity

List of Measurement IDs

Measurement ID	Measurement Name	Value Range	Unit
4097	Air Temperature	-40~90	°C
4098	Air Humidity	0~100	%RH
4099	Light Intensity	0~188000	Lux
4100	CO2	0~10000	ppm
4101	Barometric Pressure	300~1100000	Pa
4102	Soil Temperature	-30~70	°C
4103	Soil Moisture	0~100	%RH
4104	Wind Direction	0~360	°
4105	Wind Speed	0~60	m/s
4106	pH	0~14	PH
4107	Light Quantum	0~5000	umol/m²s
4108	Electrical Conductivity	0~23	dS/m
4109	Dissolved Oxygen	0~20	mg/L
4110	Soil Volumetric Water Content	0~100	%
4111	Soil Electrical Conductivity	0~23	dS/m
4112	Soil Temperature(Soil Temperature, VWC & EC Sensor)	-40~60	°C
4113	Rainfall Hourly	0~240	mm/hour
4115	Distance	28~250	cm
4116	Water Leak	true / false	
4117	Liguid Level	0~500	cm
4118	NH3	0~100	ppm
4119	H2S	0~100	ppm
4120	Flow Rate	0~65535	m3/h
4121	Total Flow	0~6553599	m3
4122	Oxygen Concentration	0~25	%vol
4123	Water Eletrical Conductivity	0~20000	us/cm
4124	Water Temperature	-40~80	°C
4125	Soil Heat Flux	-500~500	W/m²
4126	Sunshine Duration	0~24	h
4127	Total Solar Radiation	0~5000	W/m²
4128	Water Surface Evaporation	0~10000	mm
4129	Photosynthetically Active Radiation(PAR)	0～5000	umol/m²s

4130	Accelerometer	0,0,0~x.xx,y.yy,z.zz	m/s
4131	Volume	0~100	dB
4133	Soil Tension	-100~0	KPA
4134	Salinity	0~20000	mg/L
4135	TDS	0~20000	mg/L
4136	Leaf Temperature	0~100	°C
4137	Leaf Wetness	-40~85	%
4138	Soil Moisture-10cm	0~100	%
4139	Soil Moisture-20cm	0~100	%
4140	Soil Moisture-30cm	0~100	%
4141	Soil Moisture-40cm	0~100	%
4142	Soil Temperature-10cm	-30~70	°C
4143	Soil Temperature-20cm	-30~70	°C
4144	Soil Temperature-30cm	-30~70	°C
4145	Soil Temperature-40cm	-30~70	°C
4146	PM2.5	0~1000	μg/m3
4147	PM10	0~2000	μg/m3
4150	AccelerometerX	-49.99~49.99	m/s
4151	AccelerometerY	-49.99~49.99	m/s
4152	AccelerometerZ	-49.99~49.99	m/s
5100	Switch	100~200	

设备状态ID表

SensorHub

Status ID	Status name
3000	battery
3009	Signal strength
3013	The onboard temperature
3017	Battery charging state(1:In the charging，2:Full charged,3:No full charged)
3900	Data collection interval

LoraPP Node

Status ID	Status name
3000	battery,Once a day

LoraWan Node

Status ID	Status name
3000	battery,Once a day

List of Error Code

code	msg	description
10000	Framework analysis error	An unknown error has occurred in the system
11201	You have no options permission	No operation permission
11202	Request parameters are invalid	Invalid request parameter
11203	Invalid device EUI	EUI format error
11317	Group not exist	Group does not exist
11327	Group is exists	Group already exists
11330	Check eui error	Device EUI does not exist
11334	Eui is binded	The device has been bound