# IOT Platform Integration Api

Below you can find all the api methods IOT Platform offers to manage your Iot domain. Request and response samples are provided for all api methods with detailed parameter descriptions.

# 1. Introduction

Having REST principles in mind, IOT Platform api is built with easy-to-guess resource urls using standard http verbs. Payloads are provided in json.

Response codes for success and error conditions meet general patterns. ISO8601 date format is used for all dates. Daa values in api responses are provided in utc.

Platform also supports mqtt protocol in connectivity layer. Rest principles are applied in mqtt topics as in http urls.

# 2. Accessing Demo Platform

Platform supports two connection protocols namely, http(rest) and mqtt.

Your demo credentials will be provided, along with the document.

## 2.1. http access

You can use broker address given by email for rest login samples.

## 2.2. mqtt access

You can use given broker address by email for mqtt samples:

# 3. Versioning

All api methods are preceded with related version numbers. Obsolete methods will be removed in order to keep document up to date.

Older versions will be supported for backward compatibility until the announced date that they will be discontinued.

# 4. Authentication

IOT Platform uses tokens to authenticate requests. You can use these tokens to access IOT Platform services.

Tokens will be valid for a short amount of time (i.e. 30 min for apis general), so it will be a good practice to renew your token when it is about to expire to continue using services.

Token expiration date will be returned in response to let you know when your token will expire so that you will be able to renew your token prior to expiration.

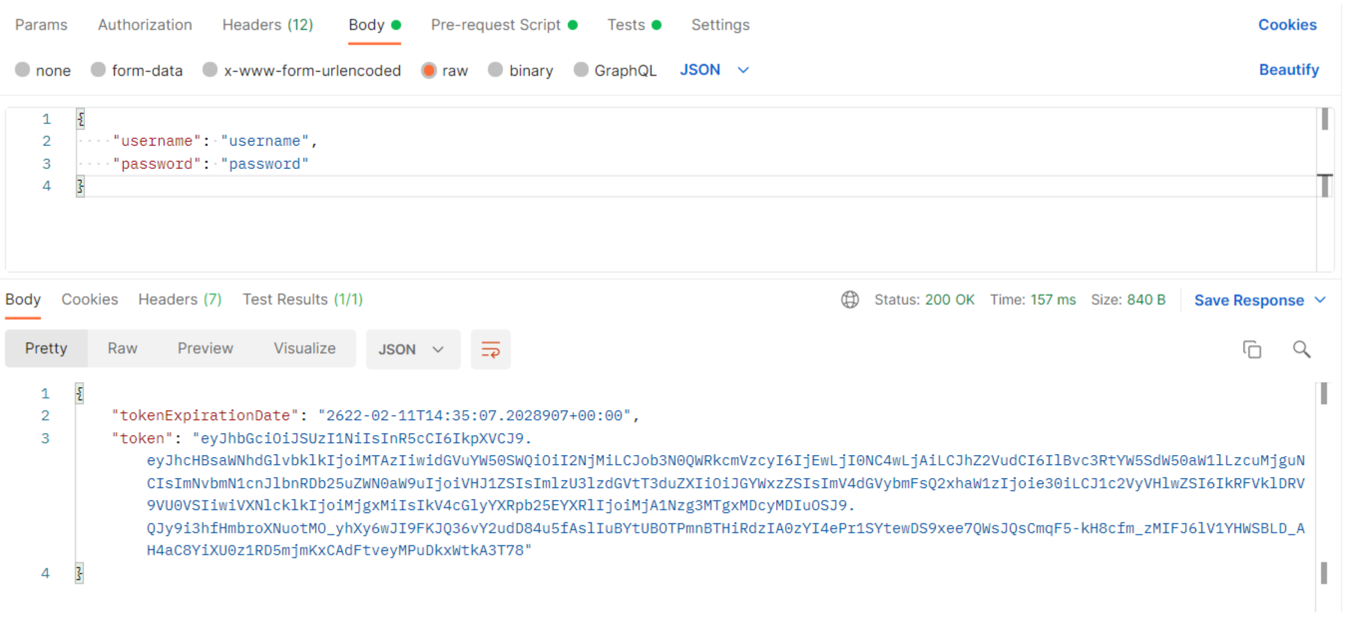
## 4.1. Retrieving Tokens

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>POST /v1/login/{tenantCode} | **tenantCode**<br><br>*string (required)* | platform-wise unique value identifying your tenant. You can have maximum number of users allowed for your tenant type. |
| **body**<br><br>`{ "username": "{username}", "password": "{password}" }` | **username**<br><br>*string (required)* | tenant-wise unique value identifying your user. Your access to resources will be determined by the privileges defined for your user. |
|  | **password**<br><br>*string (required)* | Your strong password for authentication. |

**Request Example**

```
POST /v1/login/myTenantCode  HTTP/1.1
Host: {platform url}
Content-Type: application/json
{
  "username": "superUser",
  "password": "superPasswort!"
}
```

**Sample screenshot for login is below**



```
{
    "username": "username",
    "password": "password"
}
```

```
{
    "tokenExpirationDate": "2622-02-11T14:35:07.2028907+00:00",
    "token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJhcHBsaWNhdGlvbklkIjoiMTAzIiwidGVuYW50SWQiOiI2NjMiLCJob3N0QWRkcmVzcyI6IjEwLjI0NC4wLjAiLCJhZ2VudCI6IlBvc3RtYW5SdW50aW1lLzcuMjguN
        CIsImNvbmN1cnJlbnRDb25uZWN0aW9uIjoiVHJ1ZSIsImlzU3lzdGVtT3duZXIiOiJGYWxzZSIsImV4dGVybmFsQ2xhaW1zIjoie30iLCJ1c2VyVHlwZSI6IkRFVklDRV
        9VU0VSIiwiVXNlcklkIjoiMjgxMiIsIkV4cGlyYXRpb25EYXRlIjoiMjA1Nzg3MTgxMDcyMDIuOSJ9.
        QJy9i3hfHmbroXNuotMO_yhXy6wJI9FKJQ36vY2udD84u5fAslIuBYtUBOTPmnBTHiRdzIA0zYI4ePr1SYtewDS9xee7QWsJQsCmqF5-kH8cfm_zMIFJ6lV1YHWSBLD_A
        H4aC8YiXU0z1RD5mjmKxCAdFtveyMPuDkxWtkA3T78"
}
```

| Response | Parameter | Notes: |
|---|---|---|
| **Success**<br>`{`<br>`  "tokenExpirationDate": "{tokenExpirationDate}",`<br>`  "token": "{token}"`<br>`}` | **tokenExpirationDate**<br><br>*date (required)* | You can use token expiration date to renew your token when it is about to expire. |
| | **token**<br><br>*string (required)* | You will use your token to access IOT Platform services. Token parameter usage will not be explained on every api request definition but you may find the token usage in every sample request. |
| **Error - 401**<br>`{`<br>`    "errors": [`<br>`        {`<br>`            "code": "User.UsernameOrPasswordInvalid",`<br>`            "description": "User name or password is invalid."`<br>`        }`<br>`    ]`<br>`}` | **errors**<br><br>*string array (required)* | "User name or password is invalid." error message is returned when wrong credentials are supplied. |
| **Error - 400**<br>`{`<br>`"errors": [`<br>`"Missing or invalid request parameters"`<br>`  ]`<br>`}` | **errors**<br><br>*string array (required)* | "Missing or invalid request parameters" error message is returned when request parameters are not correct.<br><br>This is the general response you will receive for malformed api requests. |

**Response Example**

```
{
    "tokenExpirationDate": "2020-05-11T22:32:16.3406555+00:00",
    "token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI"
}
```

## 4.2. Using Tokens

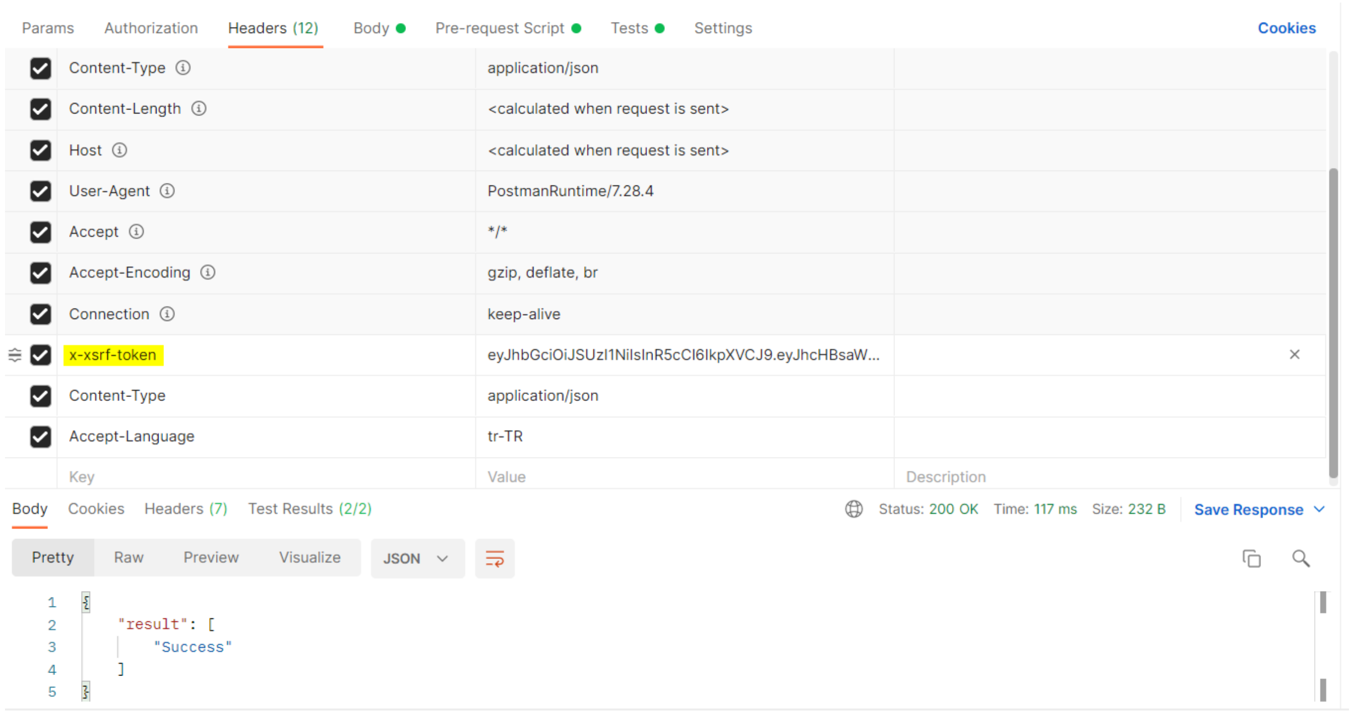
Token is the key to access  all the apis that require authentication based on the scope of your privileges.

When making api calls you pass your token on request header with x-xsrf-token attribute as below:

**Request Example**

```
GET /v1/vendor/apps HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

**The token taken from the login should be used in data sending.**



| Key | Value | Description |
|---|---|---|
| Content-Type | application/json | |
| Content-Length | <calculated when request is sent> | |
| Host | <calculated when request is sent> | |
| User-Agent | PostmanRuntime/7.28.4 | |
| Accept | */* | |
| Accept-Encoding | gzip, deflate, br | |
| Connection | keep-alive | |
| x-xsrf-token | eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBsaW... | |
| Content-Type | application/json | |
| Accept-Language | tr-TR | |

```
{
    "result": [
        "Success"
    ]
}
```

# 5. Authorization

By providing tenantcode at logon , you will access only resources assigned to you under that tenant, such as devices, device data etc.

# 6. Device Integration

Quite a lot protocols are involved for an Iot solution in a real world scenario. Sensors use protocols to connect to gateways, gateways use protocols both to connect to internet and to send data to platform through internet.

Sensor network connection protocols  are either wireless such as zigbee, zig-wave, bluetooth, wi-fi, gsm, lora or wired such as ethernet.

After internet connection has been established for sensors/gateways then devices need to send their data over internet using application protocols supported by iot platform.

## 6.1. Supported Protocols

Two mainstream protocols are supported for sending device data to IOT Platform platform, namely http (via rest api) and mqtt.

In mqtt protocol, devices are connected to mqtt broker and a persistent connection is established until connection is broken.

This approach follows a publish - subscribe pattern.

On the other hand since http is based on request  - response mechanism, it is a disconnected protocol by its nature.

Whichever protocol you prefer, you do not need physical devices to send data, you may send data by simulating devices. You may  develop an application and that application may authenticate and  send data  as long as data format and boundaries meet corresponding definitions on platform.

**Production**

On production environments http traffic should flow in a secure https channel. Mqtt traffic may also be required to flow in a secure mqtts channel depending on the business case.

### 6.1.1. Http

After retrieving token via authentication you are able to send data to platform using below format. But if device and its device profile(device type) has not been defined on the platform previously, device data will be rejected by the platform.

#### 6.1.1.1. Sending Data

You may send multiple datapoints in a single api call which may belong to multiple devices and multiple properties. Sending multiple datapoints in batches improves overall data transfer performance since processes like connection and authentication execute every api request.

Maximum number of datapoints allowed in a batch data is 100. If datapoint count exceeds maximum limit, payload will be rejected.  Api will return with http code 200 even faulty datapoints exist.  When multiple datapoints are sent, corresponding result messages are listed in the order of submitted datapoints.

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br>POST /v1/device/device-data | | Device identifier will be presented in the request body. |

**body**

```
[{
"propertyName":
"{propertyName}",
"value":{value},
"customId":
"{customId}",
"time": "{time}",
"geo": {
            "lat":
{lat},
            "lon":
{lon},
            "alt": {alt}
        }
}]
```

| | |
|---|---|
| **propertyName**<br><br>*string (required)* | unique value per device profile, identifying your data channel. Property names are predefined on the platform for corresponding device profiles. |
| **value**<br><br>*object (required)* | could be string, number or json depending on property data type and should stay within boundary limits specified on platform. |
| **customId**<br><br>*string (required)* | unique identifier value  for your device.<br><br>customid value should match "{identifer type}":"{identifier value}" pattern.<br><br>Allowed identifier type enumerations are: sn, mac, id, imei.<br><br>sn: Serial number<br><br>mac: Mac no<br><br>id: custom identifier<br><br>imei: IMEI<br><br>Custom id values should be unique per device - vendor. |
| **time**<br><br>*datetime (required)* | time is the original measurement time of the data. It should be in ISO8601 date format. |
| **lat**<br><br>*number (optional)* | latitude of your location. |
| **lon**<br><br>*number (optional)* | longitude of your location. |
| **alt**<br><br>*number (optional)* | altitude of your location. |

**Request Example**

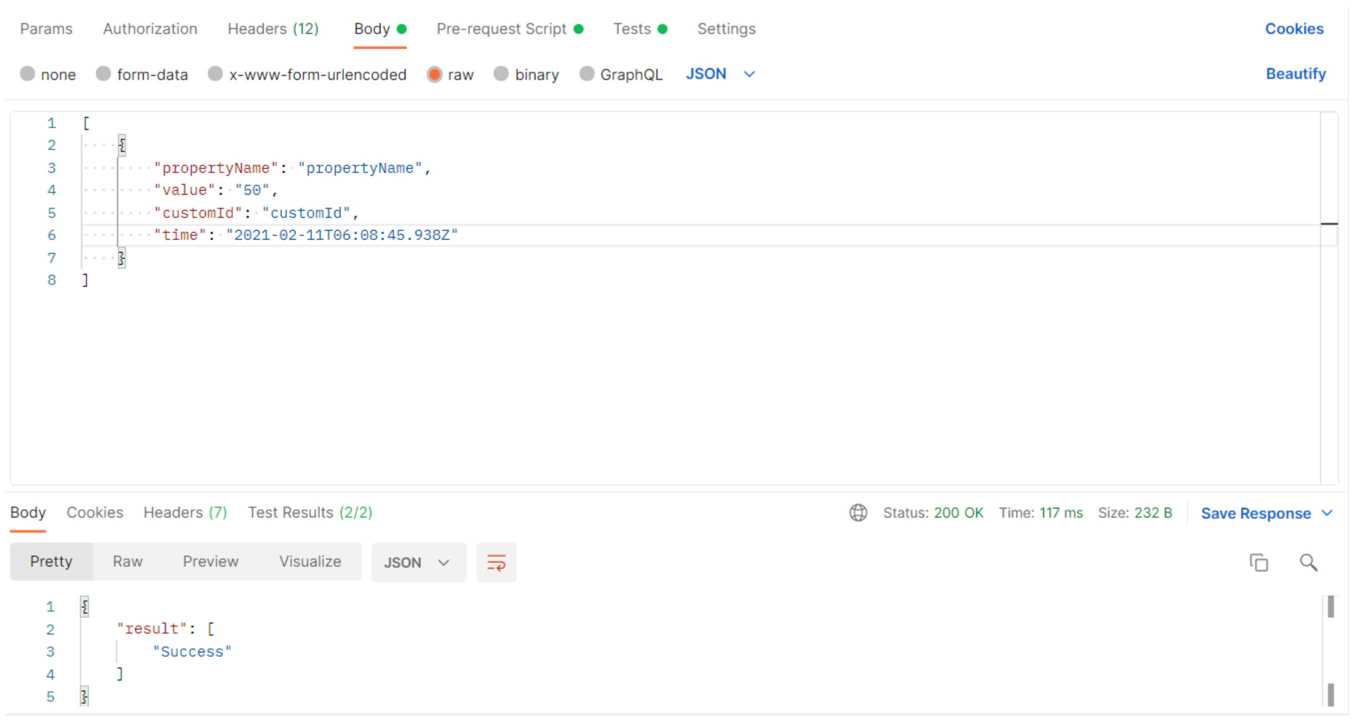
```
POST /v1/device/device-data HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
Content-Type: application/json

[
  {
    "propertyName": "battery",
    "value": 50,
    "customId": "sn:0123456789013",
    "time": "2019-10-13T15:55:09+03:00",
        "geo": {
            "lat": 40.3,
            "lon": 40.3,
            "alt": 100
        }
  },
  {
    "propertyName": "battery",
    "value": 49,
    "customId": "sn:0123456789013",
    "time": "2019-10-13T16:55:09+03:00",
        "geo": {
            "lat": 40.3,
            "lon": 40.3,
            "alt": 100
        }
  }
]
```

| Response | Parameter | Notes: |
|---|---|---|
| **Success or Error - 200**<br>`{ "result": [ {response message} ] }` | **result**<br>*string (required)* | Since resulting messages are returned for submitted  items in the same order, you will know which datapoints has been successful and/or failed.<br>As long as the request is valid it does not matter if some or all of submitted  datapoints' values are faulty, response always returns with 200 status code. |
| **Error - 500**<br>`{ "errors": [ "More than 100 datapoints are not allwed per batch"   ] }` | **errors**<br>*string array (required)* | No data is processed and "More than 100 datapoints are not allwed per batch" error message is returned if datapoint size exceeds allowed limit. |

**Response Example**

```
{
  "result": [
    "Success",
    "DataPoint is not accepted, the same data cannot be sent",
    "DataPoint is not accepted, the same data cannot be sent",
    "DataPoint is not accepted, the same data cannot be sent",
    "Device not found, possibly wrong deviceId / customId ...",
  ]
}
```

**Sample screenshot for datafeed is below**



```
[
    {
        "propertyName": "propertyName",
        "value": "50",
        "customId": "customId",
        "time": "2021-02-11T06:08:45.938Z"
    }
]
```

```
{
    "result": [
        "Success"
    ]
}
```

## 6.1.2. Mqtt

Devices need to authenticate with broker before they could start sending data. Mqtt broker requires username, password and clientid for authentication.

Username, password and clientid will be created on the platform, connection credentials  should be configured on devices accordingly in order to connect to broker.

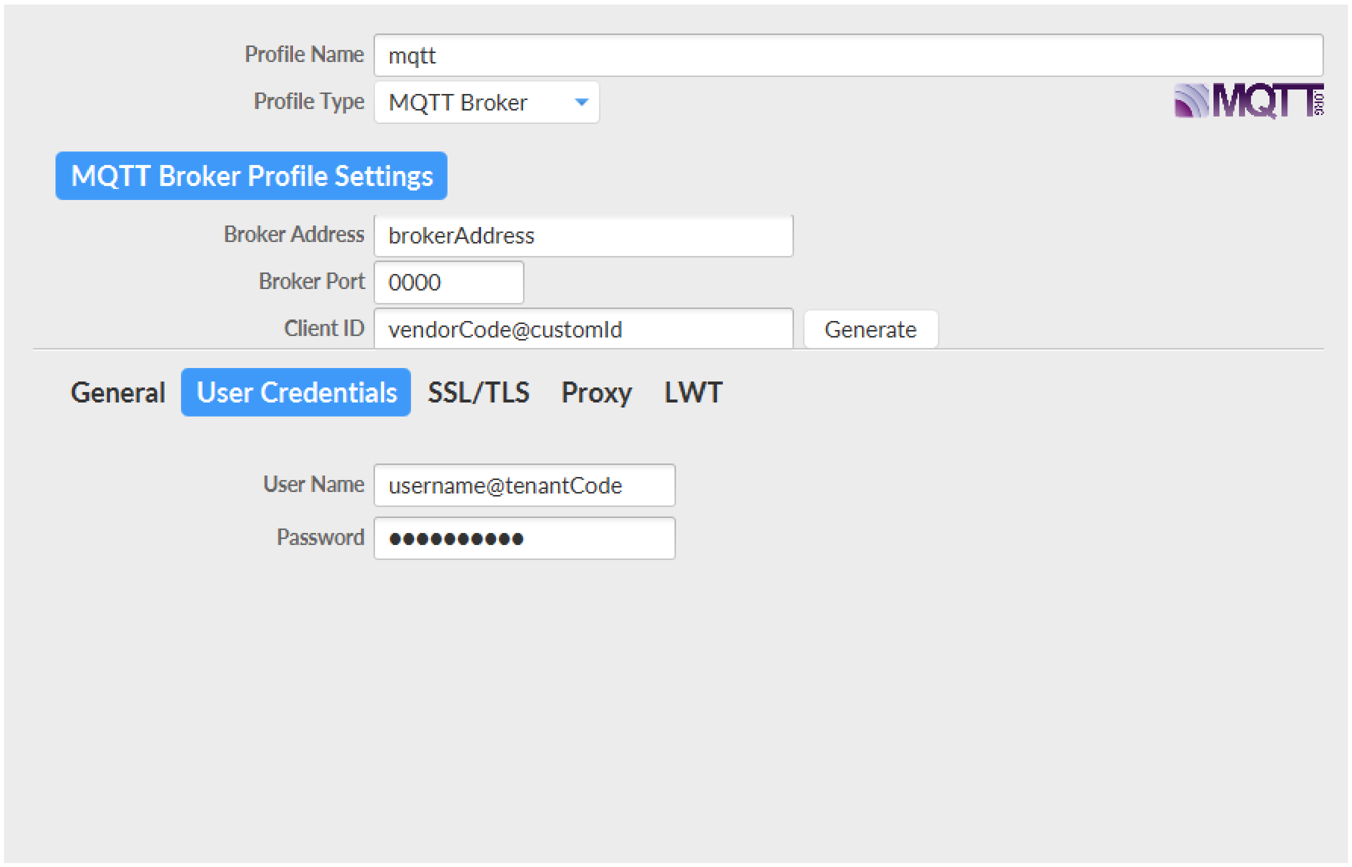
**About Broker Application**

Mqtt.fx is recommended for data sending. You can easily download it according to your operating system and system type. Sample datafeeds on Mqtt is below.

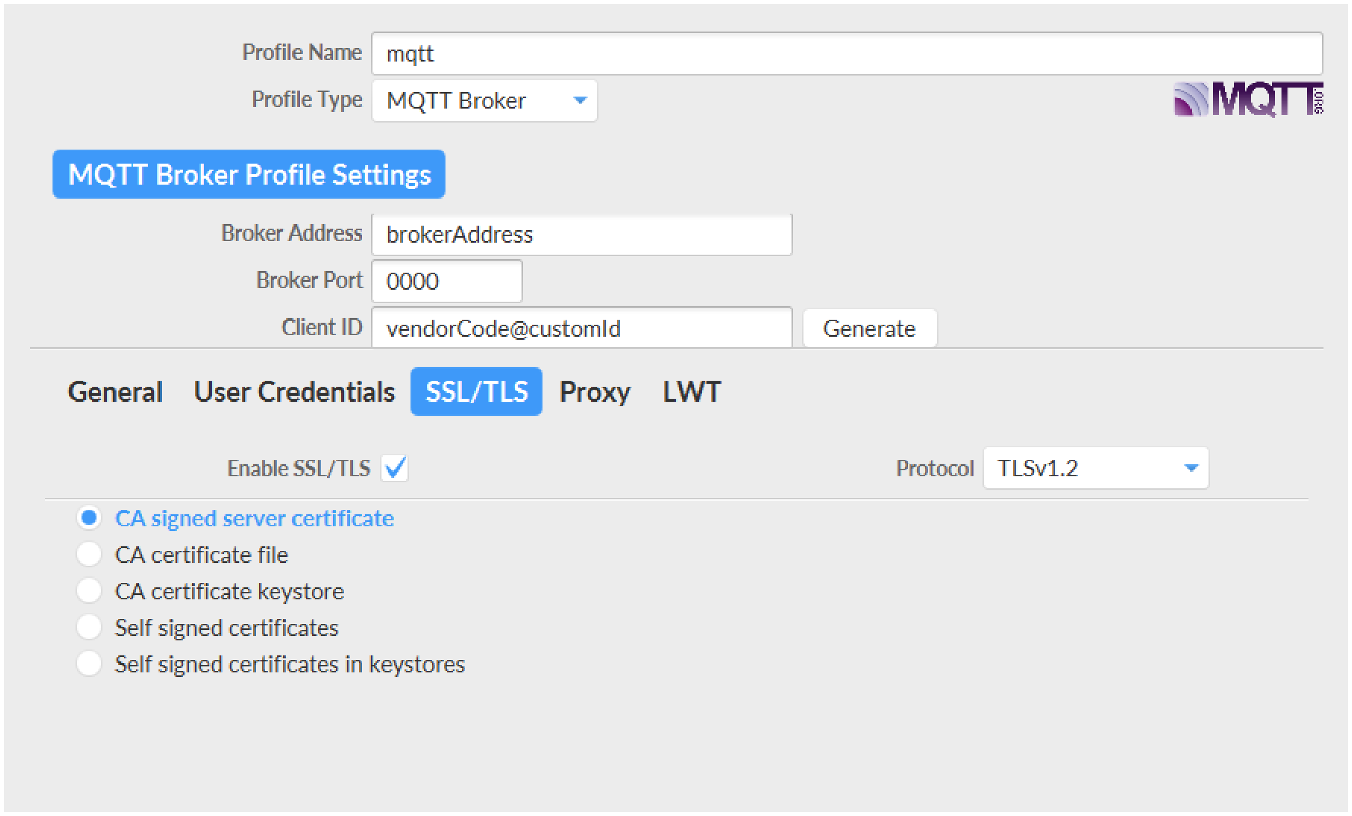
### 6.1.2.1. Authentication

| Broker Connection Parameter | Notes: |
|---|---|
| | Broker supports multiple connections with the same username and password but clientid information needs to be  unique . |

| | |
|---|---|
| **username**<br>*string(required)* | supplied by platform.<br>will follow the convention username@{tenantCode} |
| **password**<br>*string(required)* | provided by platform. |
| **clientid**<br>*string(required)* | provided by platform.<br>will be unique and follow the convention: vendorCode@customID |



| | |
|---|---|
| Profile Name | mqtt |
| Profile Type | MQTT Broker |
| Broker Address | brokerAddress |
| Broker Port | 0000 |
| Client ID | vendorCode@customId |
| User Name | username@tenantCode |
| Password | ●●●●●●●●●● |


Profile Name: mqtt

Profile Type: MQTT Broker

**MQTT Broker Profile Settings**

Broker Address: brokerAddress

Broker Port: 0000

Client ID: vendorCode@customId

Enable SSL/TLS: ✓

Protocol: TLSv1.2

- (●) CA signed server certificate
- ( ) CA certificate file
- ( ) CA certificate keystore
- ( ) Self signed certificates
- ( ) Self signed certificates in keystores

### 6.1.2.2. Sending Data

Mqtt protocol uses topics to receive and send data. Data is sent in batches but maximum datapoint count limit (100) will apply.

Devices are required to publish messages at least Qos1 quality of service level (or Qos2 if appropriate) since delivery of data cannot be guaranteed by mqtt protocol for Qos0.

As for the opposite direction, devices should subscribe to topics at Qos2 level not to miss commands that could be sent by the platform.

Devices may send coordinates information along with sensor data if they have built-in support so data could be categorized on platform by both geographic location and time.

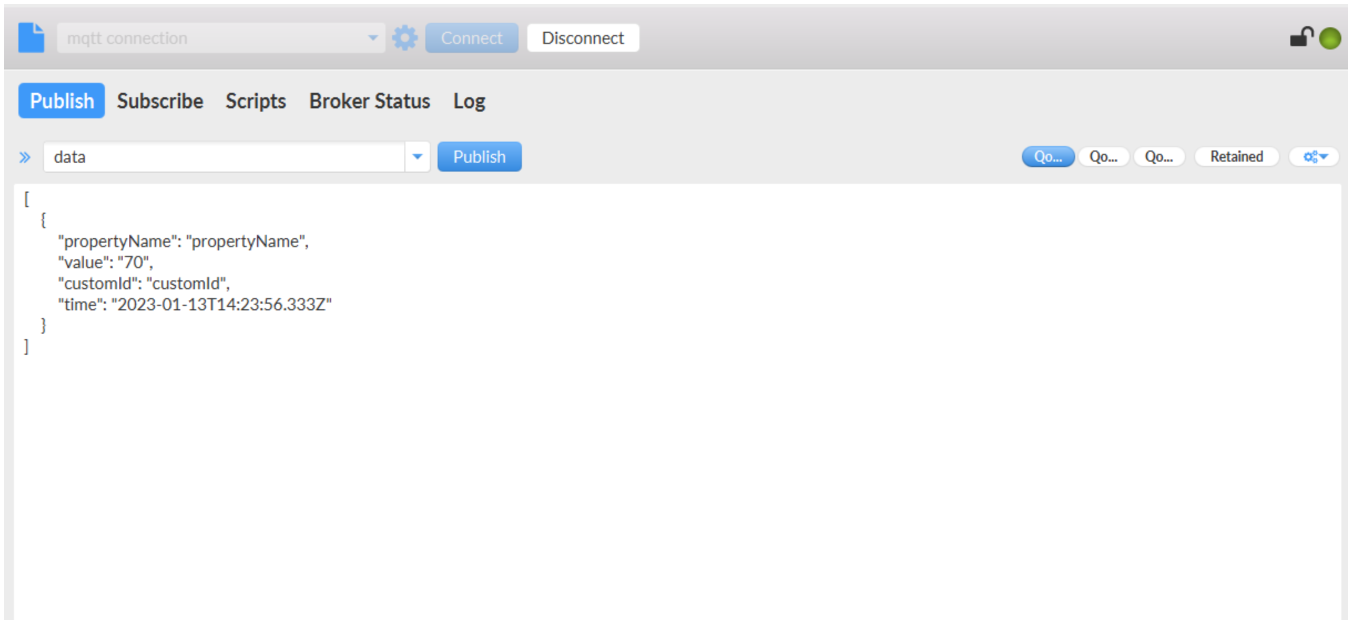
| Topic | Parameter | Notes: |
|---|---|---|
| **topic url**<br><br>**publish**<br><br>data | | No parameters used in topic url. Tenants will be differentiated in broker username.  Each user name is eligible for sending data for a single tenant. |
| **payload**<br><br>`[{`<br>`"propertyName":`<br>`"{propertyName}",`<br>`"value":{value},`<br>`"customId":`<br>`"{customId}",` | **propertyName**<br><br>*string (required)* | unique value per device profile, identifying your data channel. Property names are predefined on the platform for corresponding device profiles. |
| | **value**<br><br>*object (required)* | could be string, number or json depending on property data type and should stay within boundary limits specified on platform. |

```
"time": "{time}",
"geo": {
            "lat": {lat}
,
            "lon": {lon}
,
            "alt": {alt}
        }
}]
```

| | |
|---|---|
| **customId**<br><br>*string (required)* | unique identifier value  for your device.<br><br>customid value should match "{identifer type}":"{identifier value}" pattern.<br><br>Allowed identifier type enumerations are: sn, mac, id, imei.<br><br>sn: Serial number<br><br>mac: Mac no<br><br>id: custom identifier<br><br>imei: IMEI<br><br>Custom id values should be unique per device - vendor. |
| **time**<br><br>*datetime (required)* | original measurement time of the data. It should be in ISO8601 date format. |
| **lat**<br><br>*number (optional)* | latitude of your location. |
| **lon**<br><br>*number (optional)* | longitude of your location. |
| **alt**<br><br>*number (optional)* | altitude of your location. |

**Broker Publish Example**

```
topic        : data
payload        :
[
  {
    "propertyName": "battery",
    "value": 50,
    "customId": "sn:0123456789013",
    "time": "2019-10-12T16:02:08+03:00",
        "geo": {
            "lat": 40.3,
            "lon": 40.3,
            "alt": 100
        }
  },
  {
    "propertyName": "battery",
    "value": 49,
    "customId": "sn:0123456789013",
    "time": "2019-10-12T17:02:08+03:00",
        "geo": {
            "lat": 40.3,
            "lon": 40.3,
            "alt": 100
        }
  }
]
```



```
[
  {
    "propertyName": "propertyName",
    "value": "70",
    "customId": "customId",
    "time": "2023-01-13T14:23:56.333Z"
  }
]
```

## 6.2. Common Sensors

There are some basic information most (if not all) devices have in common such as firmware version, ip no, lifetime etc. Devices are required to send these information either periodically or when triggered based on the corresponding use case.

Data for *non-periodic* built-in sensor topics is expected to be sent as soon as it is created.

Data for some *non-periodic* sensor topics also needs to be sent once a day to avoid unintentional data loss due to networking conditions etc.

Below you can find common sensor topics.

| Property Name | Data Type | Frequency | Values/Format | Remarks |
|---|---|---|---|---|
| status | string | periodically | *valid values:*<br><br>*online*<br><br>*device_error*<br><br>*(sample for device_error*<br><br>[<br>{<br>"propertyName": "status",<br>"value": "DEVICE_ERROR",<br>"customId": "sn:456789765465633443",<br>"code":"502",<br>"message":"device disconnected",<br>"time": "2021-02-24T11:09:47.506Z"<br>}<br>]<br><br>) | It is required to send its status information (status=online) periodically.<br><br>The period depends on device characteristics and related business cases. Expected maximum period for a device to send data will be specified on its corresponding device profile.<br><br>online : device is operating normally<br><br>device_error : device is in faulty condition (devices can also send code and message)<br><br>Although device may not send its status as *device_error* when it is in a faulty condition, if integration is provided via a device backend system, it may send *device_error* status for that device via http.<br><br>Similarly in mqtt, a gateway may send connected sensor's error state on behalf of it. |
| firmwareversion | string | after any restart and once a day | no specific value format | Device firmware versions are necessary for tracking expected device functionality and planning upgrades. |
| ip | string | after any restart and once a day | based on ip version<br><br>(ipv4, ipv6 etc.) | ip numbers are necessary to enable remote management of a device via other protocols like ssh etc. |

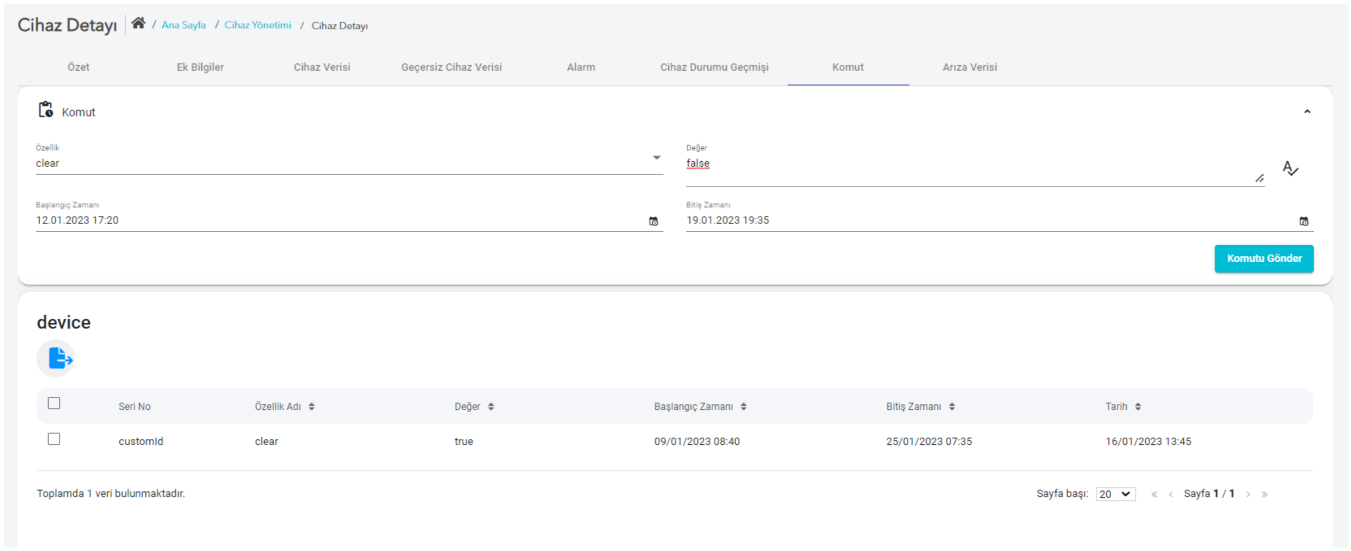
| | | | | |
|---|---|---|---|---|
| devicecommandresult | string | when device respondes to a command | <code>:<uuid> | Devices need to log the command results  (either success or fail) using datafeed to devicecommandresult adding uuid of corresponding command.<br><br>Parts of log expressions are:<br><br>code: Log code and its standard description needs to be defined on platform to create more meaningful reports.<br><br>uuid: If log created due to a platform command, corresponding command transaction id needs to be provided.<br><br>See Table 6.3.1 |
| log | string | when created | free format | free format |
| gatewaylog | string | when gateway sends log | <code>:<info> | Gateway logs provide useful information about state changes on devices and provide information about network status.<br><br>code: Log code and its standard description needs to be defined on platform to create more meaningful reports.<br><br>info: If gateway has extra information about the problem, please send us this data at info field<br><br>See log codes in Table 6.5.1 |
| snapshot | json | after any restart and in response to getsnaphot command | device specific<br><br><br>*sample:*<br><br>*{"dim-level": 10, "failuretoconnect":5, "uptimehours": 12}* | Provides general operational information specific to device. |
| remaininglife | number | once a day if needed | days left for replacement | If device's lifetime is affected by usage characteristics or environmental conditions etc. then device should provide its remaining lifetime  once a day.<br><br>If a device's lifetime linearly decreases  every day then device does not need to send this value. Lifetime needs to be defined as linearly descresing on platform. |
| battery | number | periodically | percentage of battery left | Battery needs to be replaced before it is fully discharged to let device operate continuously.<br><br>1 hour interval may be sufficient for most cases. |
| coordinates | json | periodically | *sample:*<br><br>*{"lat": 40.3, "lon": 40.3, "alt": 100},* | Device with a GPS sensor could send its coordinates periodically for tracking on a map.<br><br><br>Attributes explanations as below:<br><br>lat : lattitude<br><br>lon : longitude<br><br>alt  : altitude |
| gateway | string | when gateway's sensors have changed and once a day | customid of gateway with identifier<br><br>*sample:*<br><br>*"sn:0123456789014"* | Gateways are required to send gateway information for each sensor device if a gateway's sensor configuration is changed by adding or removing sensors.<br><br>For removed sensors gateway can send sensor's gateway information as "null" |
| mode | string | when *setmode* command is sent by device | *valid values:*<br><br>ACTIVE<br><br>PASSIVE | Mode information |

## 6.3. Commands

Some devices can respond to certain commands and act accordingly. You can send commands to these devices by *command* topic.

Devices need to subscribe to command topic at Qos2 (for exactly once delivery as an mqtt protocol standard).

Commands can be sent to device/devices on SkywaveIOT platform. Sample command post is below.


Cihaz Detayı / Ana Sayfa / Cihaz Yönetimi / Cihaz Detayı

Özet | Ek Bilgiler | Cihaz Verisi | Geçersiz Cihaz Verisi | Alarm | Cihaz Durumu Geçmişi | Komut | Arıza Verisi

Komut

Özellik: clear

Değer: false

Başlangıç Zamanı: 12.01.2023 17:20

Bitiş Zamanı: 19.01.2023 19:35

Komutu Gönder

device

| | Seri No | Özellik Adı | Değer | Başlangıç Zamanı | Bitiş Zamanı | Tarih |
|---|---|---|---|---|---|---|
| ☐ | customId | clear | true | 09/01/2023 08:40 | 25/01/2023 07:35 | 16/01/2023 13:45 |

Toplamda 1 veri bulunmaktadır.

Sayfa başı: 20 Sayfa 1 / 1

Devices need to log the command results  (either success or fail) using datafeed to devicecommandresult adding the uuid of corresponding command.

| Severity | Code | Description | Remarks |
|---|---|---|---|
| info | 101 | command saved | need to be sent when command is saved successfully. |
| error | 102 | command not saved | needs to be sent if command could not be saved.<br><br>Additional error info is preferable. |
| info | 103 | command executed | needs to be sent when command is executed at execution date. |
| error | 104 | command not executed | needs to be sent when command could not be executed at execution date.<br><br>Additional error info is preferable. |

Table 6.3.1

If device has no gateway it can subscribe to a command topic url belonging to its own customid. If device has gateway, we will send the data to its gateway. And the topic will be command/tenantCode/vendorCode/gatewayCustomId.

When a command is intended for a sensor of a gateway, gateway will execute the command on behalf of its sensor and log the result using datafeed as if the sensor is sending the log response (sensor's customid will be in log topic payload).

| Topic | Parameter | Notes: |
|---|---|---|
| **topic url**<br><br>command/tenantCode/vendorCode/customId<br><br>or<br><br>command/tenantCode/vendorCode/gatewayCustomId | | |
| **message**<br><br>`{`<br>`  "vendorCode": {vendorCode},`<br>`  "customId": {customId},`<br>`  "propertyName": {propertyName},`<br>`  "command": {command},`<br>`  "uuid": {uuid},`<br>`  "commandStartDate": {commandStartDate},`<br>`  "commandEndDate": {commandEndDate},`<br>`  "gatewayCustomId": {gatewayCustomId},`<br>`  "tenantCode": {tenantCode}`<br>`}` | **propertyName**<br><br>*string(required)* | name of command to be executed by device. |
| | **value**<br><br>*object(required)* | could be string, number or json depending on command type. |
| | **startDate**<br><br>*string(optional)* | the command will be executable after this date |
| | **endDate**<br><br>*datetime(optional)* | the command will be executable until this date |

|  |  |  |
|---|---|---|
|  | **customId**<br><br>*string(required)* | customid of device to execute the command. |
|  | **uuid**<br><br>*string(required)* | transaction id |

**Execution of Overlapping Dates for State Setting Commands**

Start - end dates of a state setting command may intersect preceding commands' start - end dates. In this case proceeding command's value overrides only preceding start - end dates' intersecting part. Command values for non-overridden date intervals stay the same.

If device has gateway, we will send the data to its gateway. And the topic will be vendorCode/gatewayCustomId.

To better illustrate, let's give an example:

**Broker Command Example**

```
topic        : vendorCode/customId
payload        :
[

   {   "customId":"customId", "propertyName": "setmode",     "value":  "passive",  "startDate":"2022-11-06T18:
00:00+00:00",   "endDate":"2022-11-06T19:00:00+00:00", "uuid":"c3bde9a2-75e4-11eb-9439-0242ac130002" },
// Expected behavior: Device will set to its mode to passive on 2022-11-06 between 18:00 and 19:00 by UTC.

   {   "customId":"customId", "propertyName": "setmode",     "value":  "active",    "startDate":"2022-11-06T18:
30:00+00:00",  "endDate":"2022-11-06T21:00:00+00:00", "uuid":"c3bde9a2-75e4-11eb-9439-0242ac130002" },
// Expected behavior:  Device will set to its mode to passive on 2022-11-06 between 18:00 and 18:30 and active
between 18:30 -21:00 by UTC.

   {   "customId":"customId", "propertyName": "setmode",     "value":  "passive",  "startDate":
null,                          "endDate":"2022-11-06T19:30:00+00:00", "uuid":"c3bde9a2-75e4-11eb-9439-
0242ac130002" },
// Expected behavior:  Device will set to its mode to passive starting from now until 2022-11-06 19:30 and
active between 19.30-21:00 by UTC.

   {   "customId":"customId", "propertyName": "setmode",     "value":  "active",    "startDate":"2022-11-06T19:
00:00+00:00",  "endDate": null, "uuid":"c3bde9a2-75e4-11eb-9439-0242ac130002"}
// Expected behavior:  Device will set to its mode to passive starting from now until 2022-11-06 19:00 and
active forever after 2022-11-06 :19.00 by UTC.

]
```

## 6.4. Common Commands

Like common sensor topics, there are common commands applicable to most type of devices like restart, firmware upgrade etc.

Below you can find common command topics.

| Property Name | Command Type | Data Type | Value/Format | Remarks |
|---|---|---|---|---|
| setlifetime | operation only | number | days until replacement | Sets device lifetime. If lifetime does not decrease linearly, device needs to send its remaining life time everyday by decreasing this value with corresponding usage characteristics. |
| configure | operation only | json | *sample:*<br><br>*{"lightmode":"summer"}* | Configures device. Each type of device may have a different configuration.<br><br>If configuration could not be updated current configuration is not changed. |
| setconnection | operation only | json | *sample:*<br><br>*{"username":"x", "password":"y", "broker":"mqtt broker address"}* | Sets new broker connection for device.<br><br>If new connection could not be established, connection is reverted back to previous connection. |

| | | | | |
|---|---|---|---|---|
| firmwareupgrade | operation only | json | *sample:*<br><br>*{"path":"abb", "version":"1.2.1"}* | Triggers device to download firmware from ftp (address embedded in firmware) and upgrade.<br><br>path : ftp folder<br><br>version : version to upgrade<br><br>If device cannot connect to broker after firmware upgrade, it should revert to previous version.<br><br>If device starts having connection problems to broker it may retry to download latest firmware. |
| resend | operation only | json | *sample:*<br><br>*{"property":"temperature", "startdate": null, "enddate":null}* | Triggers device to send data stored in device for specified property and date interval. |
| setfrequency | operation only | json | *sample:*<br><br>*{"property":"temperature", "value":5}* | Sets sensor data frequency (unit is minutes). |
| execute | operation only | string | *sample:*<br><br>*"format sd"* | Sends device specific command line commands. |
| setmode | state setting | string | *valid values:*<br><br>*active*<br><br>*passive* | Sets device mode.<br><br>Device should not send or store data when mode is set to passive ; except status , firmwareversion and mode information if device is still connected.<br><br>Passive mode is mainly used for maintenance purposes.<br><br>Device is expected to send and store data when its mode is set to active. |
| factoryreset | operation only | N/A | *no value (null)* | Reverts device configuration to factory settings.<br><br>Broker connection information should not be changed not to lose device connectivity. |
| restart | operation only | N/A | *no value (null)* | Restarts device.<br><br>Mainly used for operational purposes to fix temporary issues on device. |
| getsnapshot | operation only | N/A | *no value (null)* | Triggers device to send snapshot data.<br><br>Mainly used for operational purposes to get an overall operational status of the device. |
| clear | operation only | json | *sample:*<br><br>*{"property":"log", "startdate": null, "enddate":null}* | Deletes a topic's or all topics' stored data on device for a given date interval.<br><br>if property is not specified, all topics' stored data on device will be deleted for specified date interval.<br><br>If all data on the device should be deleted, the value will be like {}. |

## 6.5. Gateway Log Codes

Below you can find gateway log codes for common scenarios. Device specific logs need to be defined on the platform when a new type of device will be added.

This is an evolving list.  New codes are likely to be added especially with the feedbacks of device vendors.

Log topic data format is free format.

| Severity | Code | Description |
|---|---|---|
| error | 201 | Unhandled error |
| error | 202 | Unknown topic |
| error | 203 | RTC Clock could not be synchronized |
| error | 204 | RAM table error |
| error | 205 | Device restarting |
| info | 206 | Fallback to default firmware |
| info | 207 | Offline data sending started |
| info | 208 | Offline data sending end |
| info | 209 | Reconnect tried for n times before connection established |
| info | 301 | GSM Soft Reset |
| info | 302 | GSM Hard Reset |

| | | |
|---|---|---|
| warning | 303 | GSM Low Signal |
| error | 304 | GPRS Error encountered n times before connection established |
| info | 401 | Sd Card Ready |
| error | 402 | Sd Card Not Found |
| error | 403 | Sd Card Memory Full |
| warning | 404 | Sd Card Busy |
| info | 501 | Sensor x connected |
| error | 502 | Sensor x not found |
| error | 503 | Sensor x has error |
| error | 601 | GPS No Data Received |
| warning | 602 | GPS Insufficient Satellites Count |
| info | 603 | GPS Ready |

## 6.6. Sending Data Stored When Offline

Devices may exist at environments where network bandwidth is limited or unstable. When a telemetry data is measured but could not be sent to platform, device needs to store these datapoints with related measurement timestamps (specified by time attribute in payload)  so it may re-send them when network conditions are available.

device_error status value is an exception.  Device may go offline due to an error condition but when it recovers and becomes online then it only needs to send online status, previous device_error status messages need not to be sent.

Device may send the root cause of failure in another agreed upon topic, but repeating error logs need not to be sent, only first error log that caused the failure with its timestamp may be sent.

# 7. Filtering Device Data

Simple data filtering is available with basic parameters such as start date, end date and device custom id. When no parameters are used all devices' all data is returned. Data is being paged to avoid timeouts.

Parameters could be combined to narrow down filtering. There are two different apis, one for filtering valid another for filtering invalid data.

## 7.1. Filtering Valid Device Data

Valid data matching correct data type and within defined boundaries can be filtered as below.

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>GET /v1/vendor/valid-device-data | | No required parameters exist in url. Below optional parameters may be used for narrowing your search. |
| | startDate<br><br>*datetime (optional)* | data with measurement time starting from this date are returned when specified. If not specified, data starting from oldest one is returned.<br>ISO8601 date format is used. |
| | endDate<br><br>*datetime (optional)* | data with measurement time until this date are returned. If not specified, data until current time is returned.<br>ISO8601 date format is used. |
| | | |

|  |  |  |
|---|---|---|
|  | **customid**<br><br>*string (optional)* | unique identifier value  for your device.<br><br>customid value should match "{identifer type}":"{identifier value}" pattern.<br><br>Allowed identifier type enumerations are: sn, mac, id, imei.<br><br>sn: Serial number<br><br>mac: Mac no<br><br>id: custom identifier<br><br>imei: IMEI<br><br>Custom id values should be unique per device - vendor. |
|  | **propertyName**<br><br>*string (optional)* | propertyName information |
|  |  | Below are ***standard paging parameters*** for listing platform objects. |
|  | **size**<br><br>*number (optional)* | number of items ro return in each page.  If not specified, default value (100) is used. When a number exceeding 100 is specified,  value is ignored and default value is used. |
|  | **page**<br><br>*number (optional)* | index of page to return. If not specified, first page is returned. |

## Request Example

```
GET /v1/vendor/valid-device-data?startDate=2019-10-12T12:38:06+03:00&endDate=2019-10-14T12:38:06+03:
00&size=2&page=0&customId=sn:0123456789013 HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

| Response | Parameter | Notes: |
|---|---|---|
|  |  | below are common wrappers for list type responses. |
| **Success**<br><br>`{`<br>`    "result": {`<br>`        "contents": [`<br>`            {`<br>`                "propertyName": "{propertyName}",`<br>`                "customId": "{customId}",`<br>`                "deviceReadTime": "{deviceReadTime}",`<br>`                "actionTime": "{actionTime}",`<br>`                "value": "{value}"`<br>`            }`<br>`        ],`<br>`        "totalPages": {totalPages},`<br>`        "totalElements": {totalElements},`<br>`        "pageNumber": {pageNumber}`<br>`    }`<br>`}` | **result**<br><br>*object (required)* | container for response |
|  | **contents**<br><br>*array (required)* | container for data array. |
|  | **totalPages**<br><br>*number (required)* | count of pages. |
|  | **totalElements**<br><br>*number (required)* | count of items in all pages. |
|  | **pageNumber**<br><br>*number (required)* | current page index starting from zero. |
|  |  | Response specific schema is as below: |

| | | |
|---|---|---|
| | **propertyName**<br><br>*string (required)* | unique value per device profile, identifying your data channel. Property names are predefined on the platform for corresponding device profiles. |
| | **customId**<br><br>*string (required)* | unique identifier value  for your device.<br><br>customid value should match "{identifer type}":"{identifier value}" pattern.<br><br>Allowed identifier type enumerations are: sn, mac, id, imei.<br><br>sn: Serial number<br><br>mac: Mac no<br><br>id: custom identifier<br><br>imei: IMEI<br><br>Custom id values should be unique per device - vendor. |
| | **deviceReadTime**<br><br>*datetime (required)* | original measurement time of the data in ISO8601 format as UTC. |
| | **actionTime**<br><br>*datetime (required)* | time when data is received by platform in ISO8601 format as UTC. |
| | **value**<br><br>*string (required)* | measurement value device has sent. |

**Response Example**

```
{
    "result": {
        "contents": [
            {
                "propertyName": "battery",
                "customId": "sn:0123456789013",
                "deviceReadTime": "2019-10-12T13:02:08.000+0000",
                "actionTime": "2020-05-30T11:11:14.000+0000",
                "value": "50"
            },
            {
                "propertyName": "battery",
                "customId": "sn:0123456789013",
                "deviceReadTime": "2019-10-12T13:01:10.000+0000",
                "actionTime": "2020-05-28T12:03:15.000+0000",
                "value": "50"
            }
        ],
        "totalPages": 1,
        "totalElements": 2,
        "pageNumber": 0
    }
}
```

## 7.2. Filtering Invalid Device Data

Invalid data unmatching the correct data type and out of defined boundaries can be filtered as below.

| Request | Parameter | Notes: |
|---|---|---|
| **url** | | No required parameters exist in url. Below optional parameters may be used for narrowing your search. |
| GET /v1/vendor/invalid-device-data | **startDate**<br><br>*datetime (optional)* | data with platform received date starting from this date are returned when specified. If not specified, data starting from oldest one is returned.<br>ISO8601 date format is used. |
| | **endDate**<br><br>*datetime (optional)* | data with platform received date until this date are returned. If not specified, data until current time is returned.<br>ISO8601 date format is used. |
| | | Standard paging parameters can be used in url. If not provided, default values will be used for these parameters. |

**Request Example**

```
GET /v1/vendor/invalid-device-data?startDate=2019-10-12T12:38:06+03:00&endDate=2019-10-14T12:38:06+03:
00&size=2&page=0 HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

| Response | Parameter | Notes: |
|---|---|---|
| **Success** | | common response wrappers for lists are described here. |
| `{ "result": { "contents": [ { "id": {id}, "userId": {userId}, "userIp": {userIp}, "readingTime": "{readingTime}", "requestBody": "{requestBody}", "reason": "{reason}", "type": "{type}" } ], "totalPages": {totalPages}, "totalElements": {totalElements}, "pageNumber": {pageNumber} } }` | **id**<br><br>*number(required)* | used for tracking invalid data on platform. |
| | **userId**<br><br>*string(required)* | id of user who has sent the invalid data. |
| | **userIp**<br><br>*string(required)* | ip number of user who has sent the invalid data. |
| | **readingTime**<br><br>*datetime(required)* | time when data is received by platform in ISO8601 date format as UTC. |
| | **requestBody**<br><br>*string(required)* | payload the device has sent. |
| | **reason**<br><br>*string(required)* | explains why data is rejected. |
| | **type**<br><br>*string(required)* | category of reject message. |

```
{
    "result": {
        "contents": [
            {
                "id": {id},
                "userId": {userId},
                "userIp": {userIp},
                "readingTime": "{readingTime}",
                "requestBody": "{requestBody}",
                "reason": "{reason}",
                "type": "{type}"
            }
        ],
        "totalPages": {totalPages},
        "totalElements": {totalElements},
        "pageNumber": {pageNumber}
    }
}
```

**Response Example**

```
{
    "result": {
        "contents": [
            {
                "id": 387,
                "userId": null,
                "userIp": null,
                "readingTime": "2020-06-04T11:53:05.000+0000",
                "requestBody": "{\r\n  \"customId\" : \"sn:0123456789013\",\r\n  \"propertyName\" : \"battery\",
\r\n  \"value\" : \"50\",\r\n  \"time\" : \"2019-10-12T16:02:11+03:00\"\r\n}",
                "reason": "DataPoint is not accepted, the same data cannot be sent",
                "type": "DUPLICATE_SENDING_ERROR"
            },
            {
                "id": 388,
                "userId": null,
                "userIp": null,
                "readingTime": "2020-06-04T11:53:08.000+0000",
                "requestBody": "{\r\n  \"customId\" : \"sn:0123456789013\",\r\n  \"propertyName\" : \"battery\",
\r\n  \"value\" : \"50\",\r\n  \"time\" : \"2019-10-12T16:02:11+03:00\"\r\n}",
                "reason": "DataPoint is not accepted, the same data cannot be sent",
                "type": "DUPLICATE_SENDING_ERROR"
            }
        ],
        "totalPages": 1,
        "totalElements": 2,
        "pageNumber": 0
    }
}
```

# 8. Setting Up Your Iot Solution

## 8.1. Device

Device is the virtual representation of a physical thing in your environment that can send data and/or receive input to perform some kind of action. It may have builtin capability to connect to internet directly or connect to a gateway that will act as a proxy to send and receive data on behalf of it. Physical sensors/gateways need to be configured to access to platform using supported protocols.

### 8.1.1. Creating Device

You can create your device on platform using either mqtt or rest api.

#### 8.1.1.1. Http

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br>POST /v1/vendor/device | | |

| | | |
|---|---|---|
| **body**<br>`{`<br>`  "brand":"{brand}",`<br>`  "customId":"`<br>`{customId}",`<br>`  "deviceProfileId":`<br>`{deviceProfileId},`<br>` "deviceProfileCode":`<br>`"{deviceProfileCode}",`<br>`  "locationId":`<br>`{locationId},`<br>`  "locationCode":`<br>`"{locationCode}",`<br>`  "lat": {lat},`<br>`"lon": {lon},`<br>`  "model":"{model}",`<br>`  "name":"{name}",`<br>`  "servedAppId":`<br>`{servedAppId},`<br>`  "status":"{status}"`<br>`}` | **brand**<br><br>*string(required)* | brand of your device |
| | **customid**<br><br>*string(required)* | unique identifier value  for your device.<br><br>customid value should match "{identifer type}":"{identifier value}" pattern.<br><br>Allowed identifier type enumerations are: sn, mac, id, imei.<br><br>sn: Serial number<br><br>mac: Mac no<br><br>id: custom identifier<br><br>imei: IMEI<br><br>Custom id values should be unique per device - vendor. |
| | **deviceProfileId**<br><br>*number (either deviceProfileId or deviceProfileCode is required)* | id representing type of your device. |
| | **deviceProfileCode**<br><br>*string (either deviceProfileId or deviceProfileCode is required)* | code representing type of your device. |
| | **locationId**<br><br>*number(optional)* | id representing location of your device.<br><br>You may provide locationId or locationCode of your device if you have already obtained from platform.<br><br>Inferring device location from geographical coordinates(latitude, longitude) is also possible. Location could be determined based on accuracy of coordinates. |
| | **locationCode**<br><br>*string(optional)* | code representing location of your device. |
| | **lat**<br><br>*number(optional)* | lattitude of your device's location. |
| | **lon**<br><br>*number(optional)* | longitude of your device's location. |
| | **model**<br><br>*string(optional)* | model of your device |
| | **name**<br><br>*string(optional)* | user friendly name of your device. |
| | **servedAppId**<br><br>*number(optional)* | application that your device used for. A device may belong to a single application. |
| | **status**<br><br>*string(required)* | could be either **active** or **passive**.<br><br>Sensor data received for passive devices are discarded. |

**Request Example**

```
POST /v1/vendor/device HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
Content-Type: application/json

{
    "brand": "my brand",
    "customId": "sn:012345678901255668722",
    "deviceProfileId": 1,
    "locationId": 1,
    "model": "Heat-01",
    "name": "Backyard Heat Sensor",
    "servedAppId": 21,
        "vendorId": 1,
    "status": "active"
}
```

| Response | Parameter | Notes: |
|---|---|---|
| **Success - 200**<br>`{ "result": {"message":"Success","id":{id}} }` | **result**<br>*object (required)* | When your device is created successfully you will receive a success message with your device id. |
| **Error - 406**<br>`{"code":"{code}","message":"{message}"," traceId":"{traceId}"}` | **code**<br>*string (required)*<br>**message**<br>*string (required)*<br>**traceId**<br>*string (required)* | http status code 406 is returned when parameter values submitted are not allowed on platform.<br>code may be considered as error category.<br>message provides user friendly description of error.<br>traceId is used for tracking/debugging the problem in further detail in platform logs. |

**Response Example**

```
{"result":{"message":"Success","id":2331}}
```

### 8.1.1.2. Mqtt

Devices themselves can also create other devices on platform via mqtt protocol.  In order to create a device using mqtt, all the required information should be provided with the same schema provided as in http protocol.

Since mqtt is not a request-response protocol, no response will be received in return.

Devices capable of connecting to broker can auto register themselves on platform before sending data. It is a preferred solution for reducing operational costs avoiding creation of devices manually.

Creating a device by mqtt protocol can also be useful when a gateway is sending data on behalf of connected sensors. Since gateway detects connected sensors, it may create these sensors on platform before  sending their data , but this time using their customids instead of its own customid.

**Broker Publish Example**

```
topic        : save-device
payload        :  {
  "brand": "brand",
  "customId": "customId",
  "deviceProfileId": 483,
  "establishmentDate": null,
  "expiryDate": null,
  "gatewayCustomId": "",
  "locationId": 3338,
  "setupLat": "",
  "setupLon": "",
  "setupAlt": "",
  "maintenanceEndDate": null,
  "model": "model",
  "name": "device",
  "servedAppId": 442,
  "status": "ACTIVE",
  "vendorId": 363,
  "warrantyEndDate": null
}
```

## 8.1.2. Listing Devices

You may need to list your devices for managing them in bulk or just for displaying.

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br>GET /v1/vendor/devices | | No required parameters exist in url. |
| | | Standard paging parameters can be used in url. If not provided, default values will be used. |

**Request Example**

```
GET /v1/vendor/devices?size=2&page=0 HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

| Response | Parameter | Notes: |
|---|---|---|

**Success**

```
{
    "result": {
        "contents": [
            {
                "name": "{name}",
                "deviceStatus":
"{deviceStatus}",
                "brand": "{brand}",
                "model": "{model}",
                "customId":
"{customId}",
                "deviceProfileId":
{deviceProfileId},
                "locationId":
{locationId},
                "servedAppId":
{servedAppId}
            }
        ],
        "totalPages": {totalPages},
        "totalElements":
{totalElements},
        "pageNumber": {pageNumber}
    }
}
```

| | |
|---|---|
| **totalPages**<br>*number(required)*<br>count of pages.<br>**totalElements**<br>*number(required)*<br>count of items in all pages.<br>**pageNumber**<br>*number(required)*<br>current page index starting from zero. | common response wrappers for lists |
| **name**<br>*string(required)* | Device Name |
| **deviceStatus**<br>*string(required)* | device status may be online, offline or error inferred by platform from device's connection or data it is sending. |
| **brand**<br>*string(required)* | Device Brand |
| **model**<br>*string(required)* | Device Model |
| **customId**<br>*string(required)* | unique identifier value  for your device |
| **deviceProfileId**<br>*number(required)* | Device profile information |
| **locationId**<br>*number(required)* | Device location information |
| **servedAppId**<br>*number(optional)* | Device served App information |

**Response Example**

```
{
    "result": {
        "contents": [
            {
                "name": "Backyard Heat Sensor",
                "deviceStatus": "offline",
                "brand": "my brand",
                "model": "HEAT-01",
                "customId": "sn:01234567890125566871З",
                "deviceProfileId": 1,
                "locationId": 1,
                "servedAppId": 21
            },
            {
                "name": "Backyard Heat Sensor 2",
                "deviceStatus": "online",
                "brand": "my new brand",
                "model": "HEAT-01X",
                "customId": "sn:012345678901255668714",
                "deviceProfileId": 1,
                "locationId": 1,
                "servedAppId": 21
            }
        ],
        "totalPages": 184,
        "totalElements": 367,
        "pageNumber": 0
    }
}
```

## 8.2. Device Profile

Term "property" defines what type of information could be measured (temperature, humidity, etc.) or received by a device (commands such as reset, firmware upgrade, etc.) . Device profile is formed by grouping common properties for the same kind of devices.

Every device is created using a device profile as mentioned previously. Device profile can have one or many properties depending on the types of data that a device may send or receive. Each property defines metada of data; type of data, boundaries, whether it is device command or sensor data.

### 8.2.1. Listing Device Profiles

You may need to list your device profiles for managing them in bulk or just for displaying.

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>GET /v1/vendor/device-profiles | | No required parameters exist in url. |

**Request Example**

```
GET /v1/vendor/device-profiles?size=2&page=0 HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

| Response | Parameter | Notes: |
|---|---|---|
| **Success**<br>`"result": {`<br>`    "contents": [`<br>`        {`<br>`        "id": {id},`<br>`        "name": "{name}",`<br>`"maxInactiveTime": {maxInactiveTime},`<br>`        "description": "{description}"`<br>`        }`<br>`    ]`<br>`}` | **id**<br>*number(required)* | unique id specifying your device profile. |
| | **name**<br>*string(required)* | your device profile name. |
| | **maxInactiveTime**<br>*number(required)* | when no data is sent from a device belonging to this device profile within maxInactiveTime interval (in seconds), device's status is marked as offline on platform. |
| | **description**<br>*string(optional)* | description of your device profile. |

**Response Example**

```
{
    "result": {
        "contents": [
            {
            "id": 28,
            "name": "Is Sensörü",
            "maxInactiveTime": 1,
            "description": null
        },
        {
            "id": 30,
            "name": "Is Sensörü 2",
            "maxInactiveTime": 1,
            "description": null
        }
        ],
        "totalPages": 2,
                   "totalElements": 4,
            "pageNumber": 0
    }
}
```

### 8.2.2. Listing Device Profile Properties

Querying a device profile's  property list allows you to know types of sensor data your device may send or commands it may receive belonging to that device profile.

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>GET /v1/vendor/device-profile/property?profileId=1 | | You need to specify profileId parameter to retrieve properties belonging to that device profile. |

**Request Example**

```
GET /v1/vendor/device-profile/property?profileId=1 HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

**Response**

```
{
    "result": [
        {
                        "id": "{id}",
                        "name": "{name}",
            "label": "{label}",
                        "dataType": "{dataType}",
            "type": "{type}",
                        "default":
{default},
            "status": "{status}",
            "unit": "{unit}",
                        "validationRegex":
"{validationRegex}",
            "minThreshold": {minThreshold},
            "maxThreshold": {maxThreshold}
        }
        ]
}
```

| Parameter | Notes: |
|---|---|
| **id**<br><br>*number (required)* | system-wide unique id specifying your device profile property. |
| **name**<br><br>*string (required)* | device profile-wise unique name specifying your device profile property. |
| **label**<br><br>*string (required)* | used for labeling your device profile property. |
| **dataType**<br><br>*string (required)* | data type used to validate data sent to your property.<br><br>supported data types are **number** and **text**. |
| **type**<br><br>*string (required)* | type of property.<br><br>could be either **in, out** or **calculated**. |
| **default**<br><br>*boolean (required)* | default properties are built-in properties added to each profile automatically by the platform. |
| **status**<br><br>*string (required)* | could be either **active** or **passive**.<br><br>Sensor data received for passive properties are discarded. |
| **unit**<br><br>*string (optional)* | unit of sensor measurement. |
| **validationRegex**<br><br>*string (optional)* | regular expressions used to validate string data sent to your property. |

|  |  |  |
|---|---|---|
|  | **minThreshold**<br><br>*string (optional)* | an alarm will be created whenever a sensor value below this value is received. |
|  | **maxThreshold**<br><br>*string (optional)* | an alarm will be created whenever a sensor value above this value is received. |

**Response Example**

```
{
  "result": [
    {
      "id": 302,
      "name": "myproperty",
      "label": "mylabel",
      "dataType": "number",
      "type": "in",
      "default": false
      "status": "active",
      "validationRegex": "^\\d+(\\.\\d+)?",
      "unit": "12",
      "minThreshold": "10",
      "maxThreshold": "70"
    }
  ]
}
```

## 8.3. Location

Devices let you you both monitor and manage your remote location. You may monitor near real-time data on screens or trigger alerts on live data for predetermined thresholds for issues requiring immediate attention. Past records on your locations provide business insight for operational purposes, reducing costs or increasing customer satisfaction. You may also react to changes in your location directly based on capabilities of your devices, such as turning on/off lights.

Your  location is a modelling your environment of interest which may contain one or many devices and it could be virtually anything such as a building, field, room, factory or a moving car.

Locations could be structured in a hierarchical way so you can group them on various levels.  By using your location hierarchy, you can manage your devices under the same hierarchy in bulk and generate reports based on levels of your choice. As an example you may create school locations, you may place faculties under schools, you may add buildings under faculties (classrooms, laboratories etc.) and rooms under buildings, etc. So you can easily compare data of your locations on same levels.

### 8.3.1. Listing Locations

You can list your locations, child locations or parent locations(roots) each by different api methods, both returns data with same schema.

**Locations**

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>GET /v1/vendor/locations |  | No required parameters exist in url. |

| | | |
|---|---|---|
| | **name**<br><br>*string(optional)* | name of your location. Since location name is unique per tenant, only one location returns if name matches. |

**Request Example**

```
GET /v1/vendor/locations?name=mylocation HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

### Root-Locations

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>GET /v1/vendor/root-locations | | No required or optional parameters exist in url.<br><br>Only root locations are listed. |

**Request Example**

```
GET /v1/vendor/root-locations HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

### Child-Locations

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br><br>GET /v1/vendor/location/children | **id**<br><br>*number(required)* | id of your location whose child locations you need to list. |
| | **direct-children**<br><br>*boolean(optional)* | if provided only first-level children are returned. |

**Request Example**

```
GET /v1/vendor/location/children?id=1&direct-children=true HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

### Location Queries' Response

| Response | Parameter | Notes: |
|---|---|---|
| **Success**<br><br>`"result": {`<br>`    "contents": [` | **id**<br><br>*number (required)* | unique id specifying your location. |

```
            "id": {id},
            "name": "{name}",
            "description":
"{description}",
                         "status":
"{status}",
                        "code":
"{code}",
            "type": "{type}",
                        "lat":
{lat},
            "lon": {lon},
            "zoneJson": {zoneJson},
            "zoneRadius":
{zoneRadius},

"placeTypeId":
{placeTypeId},

"placeTypeName":
{placeTypeName},

            "parentLocationId":
{parentLocationId}
        ]
    }
```

| | |
|---|---|
| **name**<br><br>*string (required)* | your location name. |
| **description**<br><br>*string (optional)* | description of your location. |
| **status**<br><br>*string (required)* | could be either **active** or **passive**. |
| **code**<br><br>*string (optional)* | tenant-wise unique code identifying your location.<br><br>You can use this feld for your own internal location id to help reporting. |
| **type**<br><br>*string (optional)* | type of your location. |
| **lat**<br><br>*number (optional)* | latitude of your location. |
| **lon**<br><br>*number (optional)* | longitude of your location. |
| **zoneJson**<br><br>*object (optional)* | polygon coordinates of your location.<br><br>A sample will be provided when format is finalized. |
| **zoneRadius**<br><br>*number (optional)* | radius specified for your location in meters, indicating effective area of your location. Based on this information, some alarms could be set or unassigned devices detected within this radius could be attached to this location. |
| **placeTypeId**<br><br>*number (required)* | type id of your location. |
| **placeTypeName**<br><br>*string (required)* | type name of your location (building, house, car, etc.). |
| **parentLocationId**<br><br>*number (optional)* | id representing parent location. |

**Response Example**

```
{
    "result": [
        {
                        "id": 33,
            "name": "my location",
                        "description": "this is my location",
            "status": "active",
            "code": "myLocationCode",
            "type": "geo",
            "lat": 14.5,
            "lon": 15.5,
            "zoneJson": null,
            "zoneRadius": null,
            "placeTypeId": 1,
                         "placeTypeName": "myPlaceType"
            "parentLocationId": 32
        }
  }
}
```

## 8.4. App

App describes the vertical Iot solution for your Iot environment which consists of devices, rules, reports and everything needed to operate your specific business area. A tenant may have more than one app. Apps are defined on platform by platform administrators and assigned to tenants with related privileges.

App examples could be smart irrigation, smart parking, smart lighting etc.

### 8.4.1. Listing Apps

You can list your apps belonging to your tenant as below:

| Request | Parameter | Notes: |
|---|---|---|
| **url**<br>GET /v1/vendor/apps | | No required parameters exist in url. |

**Request Example**

```
GET /v1/vendor/apps HTTP/1.1
Host: {platform url}
x-xsrf-token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhcHBsaWNhdGlvbklkIjoiMSIsInRlbmFudElkIjoiMSIsImhvcCoXwBcoYiaHcEQ8ngzI
```

| Response | Parameter | Notes: |
|---|---|---|

| Success | | |
|---|---|---|
| <pre>{<br>  "result": [<br>    {<br>      "id": {id},<br>      "name": "{name}",<br>      "deviceProfiles": [<br>        {deviceProfiles}<br>      ]<br>    }<br>  ]<br>}</pre> | **id**<br><br>*number(required)* | unique id specifying your app. |
| | **name**<br><br>*string(required)* | your app name. |
| | **deviceProfiles**<br><br>*object array(optional)* | device profiles assigned to your app.<br><br>Below device profile parameters are returned:<br><br>"id" , "name" , "maxInactiveTime", "description" |

**Response Example**

```
{
    "result": [
        {
            "id": 196,
            "name": "Smart App1",
            "deviceProfiles": []
        },
        {
            "id": 29,
            "name": "Smart App2",
            "deviceProfiles": [
                {
                    "id": 1,
                    "name": "temperature sensor",
                    "maxInactiveTime": 3,
                    "description": null
                },
                {
                    "id": 6,
                    "name": "humidity sensor",
                    "maxInactiveTime": 1,
                    "description": null
                }
            ]
        }

    ]
}
```