## **DATA FOR HACKERS**

**Tieto Intelligent Building sensors data**

**What kind of data?**

* A list of desks and meeting rooms on the 7th floor along with access to their utilization data as measurements, along with sample location data from a handful of employees.

Measurement format:

{  
 "time": "2016-10-22T18:00:34.837Z",  
 "id": "1234",  
 "self": "/measurement/measurements/1234",  
 "source": {  
 "id": "5678",  
 "self": "/inventory/managedObjects/5678”  
 },  
 "type": "desk",  
 "unit": "",  
 "value": 0  
 }

For desk and meeting room occupancy sensors “value” is 1 for occupied, 0 for unoccupied.

For location tags value is in format [x, y, z, floor, group], where xyz are meter based coordinates.

“time” is the timestamp of the last state change. Identical occupancy measurements might repeat numerous times.

A floor plan image of the 7th floor is available at <https://drive.google.com/file/d/0B-eIDMYN5nhYSUpaUGUzY1Z0ekE/view> The dimensions of the whole image area are 72.53 x 72.53 meters and the origin (0, 0 meters) is at the lower left corner. You can match the static object x, y coordinates and the dynamic employee location coordinates with the floor plan to do visualizations.

**How to access**

You can find a list of sensors in a text file called IB\_object\_list in the Drive folder. The data includes a “data\_source” that can be used for the “source” query string parameter when fetching measurements. Types are “desk” for desk occupancy sensors, “motion” for room occupancy sensors and “location” for employee location tags. The sensor’s name as well as the its relative location in the building are also included.

To fetch measurements, you can access the Cumulocity REST interface at <https://tieto.iottc.tieto.com/measurement> using a read-only account:

user: junction\_hacker

password: e\*@ND\_2foa

An example of a request to get the latest measurement for a sensor would be:

<https://tieto.iottc.tieto.com/measurement/measurements?pageSize=1&dateFrom=2016-10-21&dateTo=2016-11-30&revert=true&source=12191>

Documentation on how to use the REST interface available at <https://www.cumulocity.com/guides/reference/measurements/>

Fetch measurements using the sensor’s “data\_source” as “source”.

Additional info can be requested unless the F1 race is on.

EXTRA:

The object list now includes positions of lights in the building. They don’t have measurements that can be fetched, but their control\_id can be used to alter their state:

“Just make a GET call to https://9h21iuqi9g.execute-api.eu-central-1.amazonaws.com/v1/gateways/34/devices/8C07/setlevel/80 . The level (80 in the example) can be any value between 0 and 100. “

Replace the devices/XXXX value with the control id. You can check out the lights here at Junction next to Helvar’s table where they are mounted.

# **KONE API technical overview**

## **Introduction**

KONE Cloud APIs enable 3rd party developers to utilize and integrate KONE IoT ecosystem in internet connected applications. In the first phase, the APIs support three major use cases:

* Accessing building description (logical areas, elevators etc.)
* Making elevator calls
* Monitoring the state of an elevator

## **Hypermedia-driven API**

KONE Cloud APIs are fully hypermedia-driven REST APIs, utilizing HATEOAS (Hypertext As The Engine Of Application State) principles to decouple client and server so that the server is allowed to continue evolving without compromising backwards compatibility with existing clients. In this model, all endpoints of the API are resources that are manipulated with HTTP methods corresponding with classic CRUD (Create, Read, Update and Delete) functions. Additionally, each resource has a set of relations to other resources, effectively telling the client (and often the user) what can be done next.

For example, a client that wants to call an elevator, will create an elevator call resource using HTTP POST method. This will return a unique URL to the created resource. After this, the client may follow any changes to the call (such as assigned elevator) resource by GETting the URL. Additionally, the client may cancel the call with DELETE HTTP method.

The most important rule when creating applications that use HATEOAS REST API is that the client should never use hardcoded URLs and should never store any URLs returned by the server. The only exception to this rule is the index document, which should always be the starting point for any sequence.

## **Hypermedia relations**

Hypermedia relations are defined as a string that consists of a prioritized, space-separated list of keywords that describe the relation the requested resource has to another resource. For example, a building resource may have a link to a collection of elevators within the building with relation *“elevator device collection child”*. The exact means to discover links and their relations is specified by the hypermedia format.

To find the correct resource, a client application should iterate through all the links and find the best match against its own keywords. The keywords are not globally unique. For example, a keyword “parent” may point to an entirely different resource depending on the nature of the resource containing the relation.

## **Collection+JSON**

The primary hypermedia format utilized by the KONE Cloud APIs is Collection+JSON (<http://amundsen.com/media-types/collection/>), which is an open JSON based standard that organizes all resources into structured *collections*. A collection is the main level attribute of all Collection+JSON resources, consisting of a number of child items, links, queries (links constructed using given variables), templated write instructions and error records. It is highly recommended to read the format specification carefully through.

Collection+JSON is designed to be an extensible framework for hypermedia. As such, the core specification does not cover all the needs of a modern API. KONE may use existing extensions and other customization where beneficial. At the moment, the URI Templates extension (<https://github.com/collection-json/extensions/blob/master/uri-templates.md>) is widely used in the API.

KONE Cloud API uses a concept of *assorted collections* to embed resources for convenience access. In practice this is done simply by adding the external resources as additional entities under the items array. Each of the embedded resources is always included as a normal link and the client should not trust that the embedded resource is always available (i.e. a client following a relation may check if an item with given href is available as an item, but should not trust that it is always available even if at a given time it is). Furthermore, a client must never poll a resource when it is only expecting a change in the embedded resource.

**720 air quality data**

**What kind of data**

A list of indoor air-quality sensors (“nodes”) on the 7th floor along with access to aggregates of their recorded measurements for temperature, relative humidity, CO2 and VOC (Volatile Organic Compounds).

Node list format:

|  |
| --- |
| {  "data": {  "nodes": [  {  "city": "Espoo",  "node\_id": "0e34909f-f07f-417c-a667-bf7b12757eef",  "street": "Keilaniemi, Keilalahdentie",  "sensortypes": [  "co2",  "relative\_humidity\_percent",  "temperature\_celsius",  "voc\_ch2o\_equiv"  ],  "country": "Finland",  "timezone": "Europe/Helsinki",  "area\_name": "7. krs Tieto Oyj, Keilaniemi",  "node\_name": "Avotila It\u00e4"  },  **<... more ...>**  },  "status": 200  } |

Measurement list format:

|  |
| --- |
| {  "data": {  "measurements": [  {  "record\_time": "2016-11-23T00:00:00",  "sensors": {  "co2": {  "value\_avg": 570.0,  "value\_max": 570.0,  "value\_min": 570.0,  "value\_stddev": 0.0  },  "relative\_humidity\_percent": {  "value\_avg": 31.0,  "value\_max": 31.0,  "value\_min": 31.0,  "value\_stddev": 0.0  },  "temperature\_celsius": {  "value\_avg": 22.4461536407471,  "value\_max": 22.6,  "value\_min": 22.4,  "value\_stddev": 0.0877061365017857  },  "voc\_ch2o\_equiv": {  "value\_avg": 957.5,  "value\_max": 960.0,  "value\_min": 950.0,  "value\_stddev": 4.52267016866645  }  }  },  **<... more …>**  ]  },  "status": 200  } |

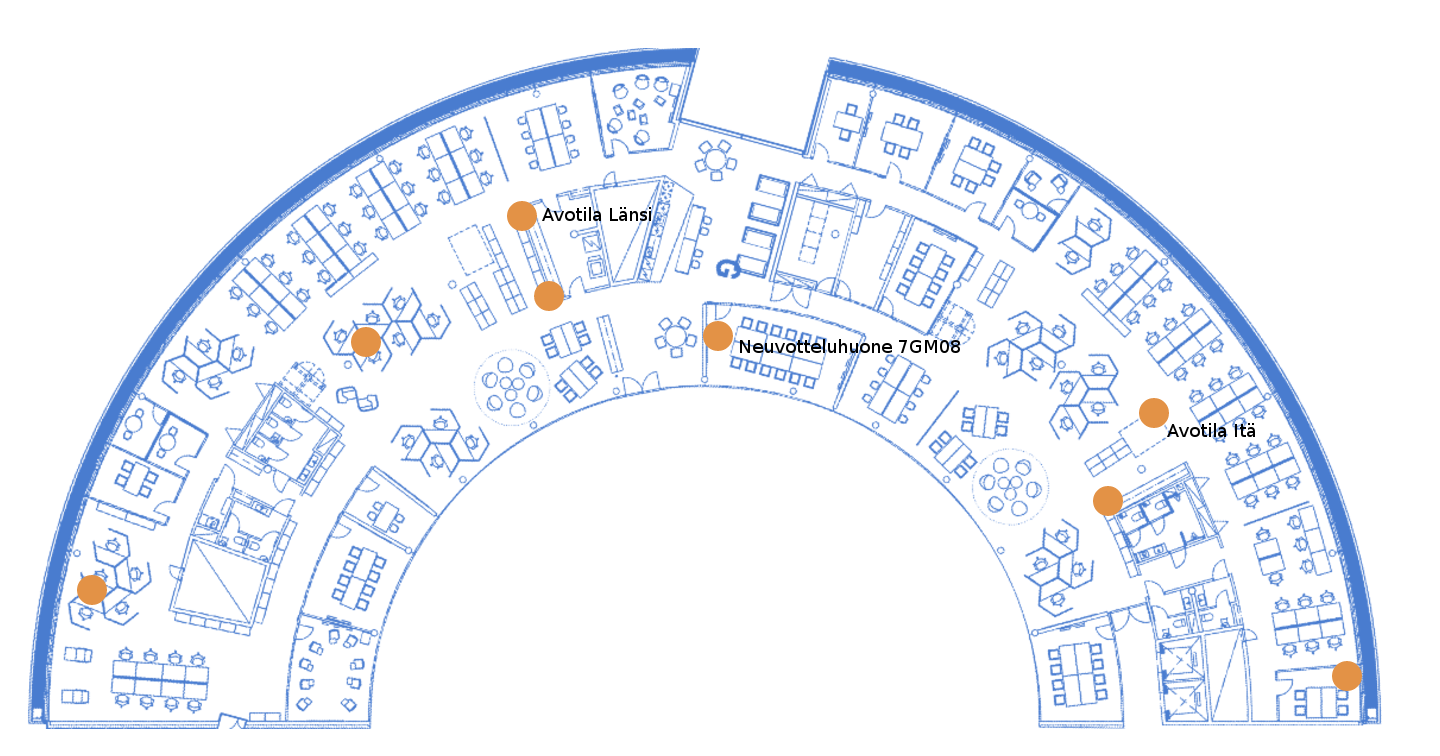
Temperature unit: °C

Relative Humidity unit is: %

CO2 unit is: ppm (parts per million)

VOC unit is: ppb (parts per billion)

**Sensor Locations:**



**How to access**

The REST API prepared for the Hackathon uses Basic HTTP authentication (<https://en.wikipedia.org/wiki/Basic_access_authentication>). Therefore, each API endpoint requires a valid “Authorization” header to be sent along with the HTTP request.

You may use the following credentials to make API calls:

Username: junction\_hackathon@720.fi

Password: i<3python

You can query the list of accessible nodes using following REST API endpoint:

|  |  |
| --- | --- |
| HTTP Method | API Endpoint |
| GET | https://hackathon.720.fi/nodes |

You can then use the node\_id from the API response to query for measurements (see the following paragraph)

You can query a list of measurements for a given node using following REST API endpoint:

|  |  |
| --- | --- |
| HTTP Method | API Endpoint |
| GET | https://hackathon.720.fi/nodes/<node\_id>/measurements?from=<timestamp>&until=<timestamp>&aggregate=<granularity> |

Where:

* <node\_id> is a UUID, which can be retrieved using the node API (see previous paragraph)
* <timestamp> is a timestamp in the format [YYYY]-[mm]-[dd]T[HH]:[MM]:[SS] (example: 2016-11-25T18:45:00)
* <granularity> defines the level of detail for the retrieved measurement aggregates. Possible values are:
  + 1d (for 1-day aggregates)
  + 6h (for 6-hour aggregates)
  + 1h (for 1-hour aggregates)
  + 5m (for 5-minute aggregates)

Please note that you can only request a limited time-span of measurements per API call (depending on the selected aggregate granularity):

|  |  |
| --- | --- |
| granularity | Maximum time-span |
| 5m | 2 hours |
| 1h | 8 days |
| 6h | 32 days |
| 1d | 93 days |

Contact Details:

[*simon@720.fi*](mailto:simon@720.fi) *(Tech Specialist) /* [*rick@720.fi*](mailto:rick@720.fi) *(Mentor)*

**Video surveillance data**

(Partner Mirasys Oy)

360 video feed from Keilalahti 7th floor

**How to access**

MKV-files are in the folder. 360 video feed from Keilalahti 7th floor<https://drive.google.com/open?id=0B8xVjU8wE8A8RG1ObzdVYmRwMGM>

Contact: Jouni Raiha

Mirasys Ltd.

Atomitie 5C, 00370 Helsinki, Finland

Mobile +358 50 594 1517

Email: jouni.raiha@mirasys.com

Web: [www.mirasys.com](http://www.mirasys.com/)

**Electricity consumption data (ISS)**

WHAT KIND OF DATA

Data is electricity consumptions from floors that are used by Tieto. Data is in CSV-file (dot (.) as decimal separator). Time is UTC time zone (Local time is UTC+2. 27.3.2016 4 AM - 30.10.2016 4 AM was daylight saving (summer time) and local time was UTC+3.). Data consists meter readings that are read at 5 minutes interval. To get consumption you’ll need to make subtraction. Note that the data contains only electricity used in these floors. It means lightning and user consumption like laptops, copy machines, coffee automats etc normal office devices.

Note 16.11.2016 at 13-15 pm (local time) there was maintenance of metering hub and this caused some displacement of the data so that time stamps are not accurate but cumulative consumption during that period is accurate.

Current transformer factor’s were not set up in the meters until 28.10.2016 9.00 UTC. This means that before that time the consumption calculated needs to be multiplied with current transformer factors. Factors are: 50 for meters

8UG01, 5F01, 4F01, 3F01, 2F01, 2G03

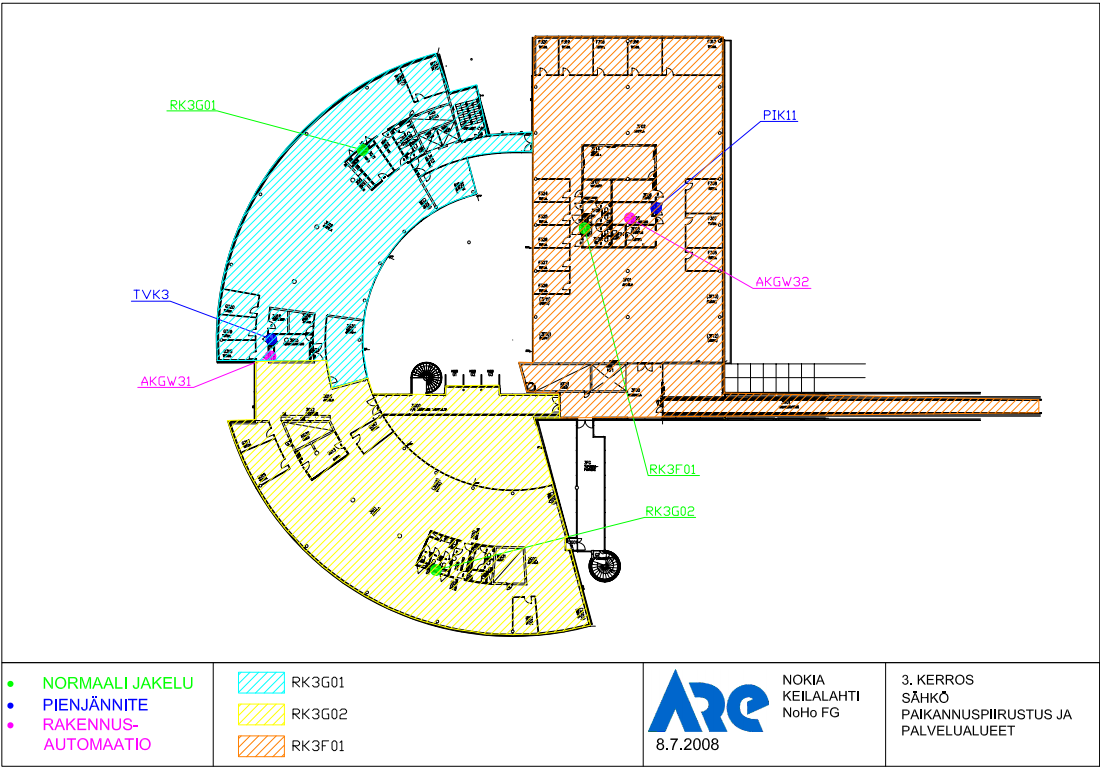
and 25 for the rest.

There is also another file (keilalahti\_datalogdump\_241116\_corrected.csv ) where meter factors are already included. Data is csv-file (field separator ; and decimal separator ,) This file contains consumptions for every 5 minutes.

If you’ll need more information about this. Contact Vesa Ruusunen / ISS Palvelut Oy

Meter names show the floor and wing of the building as in the picture below.

3G01 = third floor, G-wing, part 1



HOW TO ACCESS

Data (meter readings) is available in file named keilalahti\_datalogdump\_241116.csv (you’ll need to use meter factors in this file and make subtraction to get consumption)

Data (consumption) is available in file named keilalahti\_datalogdump\_241116\_corrected.csv . (You don’t need to worry about meter factors when operating with this file)

**Water, heating, water cooler electricity data (ISS)**

WHAT KIND OF DATA

Data contains Water consumption, building heating energy consumption, total electricity consumption and electricity consumption of water coolers (cooled water is used for building cooling). There are also other users in the building than just Tieto. For example there is a lunch kitchen in the building. Data is hourly interval data (hourly consumptions). Time zone is local time. Data is CSV-file.

Fields are

DomesticWater\_m3 = Total water usage in the building. Unit cubic meter

WarmTapWater\_m3 = Usage of the warm water that comes from tap, Unit cubic meter

DistrictHeating\_kWh = Heating energy of the building. Unit kWh

Electricity\_of\_cooling\_machines\_kWh = Electricity that cooling system uses. Unit kWh

Total\_Electricity\_kWh = Total electricity consumption in the building. Unit kWh

HOW TO ACCESS

Data is available in file named

Keila\_utilities\_semicolon.csv (in this file decimal separator is , and field separator is ; )

Keila\_utilities\_comma.cvs (in this file decimal separator is . and field separator is , )

The data in these files is identical.

**Lunch data (ISS)**

WHAT KIND OF DATA

Number or lunch portitions sold per hour and revenue for that.

HOW TO ACCESS

Data is available in file named

Tieto\_Lunch.xlxs and Tieto\_Lunch.csv (in this file decimal separator is , and field separator is ;)

Feel free to format a copy of the file for your needs.