OGC SensorThings API
Rest API

Dr. Hylke van der Schaaf
Reinhard Herzog

Do It Yourself

It‘s not as scary as you think

- Docker Quick-Start:
  https://fraunhoferiosb.github.io/FROST-Server/deployment/docker.html
- Full SensorThings API Tutorial
  https://fraunhoferiosb.github.io/FROST-Server/sensorthingsapi/1_Home.html
- Demo service:
  https://ogc-demo.k8s.ilt-dmz.iosb.fraunhofer.de/v1.1
  (As long as nobody deletes everything)
  (Or the server catches fire)

Getting to your data

Quick Overview

- Based on OASIS OData
- Base URL: http://server.de/FROST-Server/v1.1
- Read: GET
  - v1.1 → Get index
  - v1.1/Collection → Get all in a set
  - v1.1/Collection(id) → Get one from a set
- Create: POST
  - v1.1/Collection → Create a new entity
- Update: PATCH
  - v1.1/Collection(id) → Update an entity
- Update: PUT
  - v1.1/Collection(id) → Replace an entity
- Delete: DELETE
  - v1.1/Collection(id) → Remove an entity

Query URL patterns: Index

Get Service Index

- GET http://ogctest.docker01.ilt-dmz.iosb.fraunhofer.de/v1.1
  - Response:

```
{
  "value" : [
    {
      "name" : "Datastreams",
      "url" : "http://server.de/SensorThingsService/v1.0/Datastreams"
    },
    {
      "name" : "FeaturesOfInterest",
      "url" : "http://server.de/SensorThingsService/v1.0/FeaturesOfInterest"
    },
    {
    …
    …
    },
    {
      "name" : "Things",
      "url" : "http://server.de/SensorThingsService/v1.0/Things"
    }
  ],
  "serverSettings": {}
}
```

Query URL patterns: Get Collection

Get All Things

- GET http://ogctest.docker01.ilt-dmz.iosb.fraunhofer.de/v1.1/Things
  - Response:

```
{
  "value" : [
    {
      "name" : "My camping lantern",
      "description" : "camping lantern",
      "properties" : {
        "property1" : "it's waterproof",
        "property2" : "it glows in the dark"
      },
      "Locations@iot.navigationLink" : "Things(1)/Locations",
      "HistoricalLocations@iot.navigationLink":
"Things(1)/HistoricalLocations",
      "Datastreams@iot.navigationLink" : "Things(1)/Datastreams",
      "@iot.id" : 1,
      "@iot.selfLink" : "/SensorThingsService/v1.0/Things(1)"
    },
    {
      a second thing…
    }, { … }, { … }, { … }
  ]
}
```

Query URL patterns: Get one Entity

Get Specific Thing

- GET http://ogctest.docker01.ilt-dmz.iosb.fraunhofer.de/v1.1/Things(1)
  - Response:

```
{
  "name" : "My camping lantern",
  "description" : "camping lantern",
  "properties" : {
    "property1" : "it’s waterproof",
    "property2" : "it glows in the dark"
  },
  "Locations@iot.navigationLink" : "Things(1)/Locations",
  "HistoricalLocations@iot.navigationLink" : "Things(1)/HistoricalLocations",
  "Datastreams@iot.navigationLink" : "Things(1)/Datastreams",
  "@iot.id" : 1,
  "@iot.selfLink" : "/SensorThingsService/v1.0/Things(1)"
}
```

Query URL patterns: Get related Entities

Get all Datastreams of a specific Thing

- GET http://…/v1.1/Things(1)/Datastreams
  - Response:

```
{
  "value" : [
    {…},
    {…},
    {…}
  ]
}
```

Query URL patterns: Pagination

GET only 4 Observations and the total count of Observations

- GET … /v1.1/Observations?
  $top=4&
  $count=true
  - Response:

```
{
  "@iot.count" : 16,
  "@iot.nextLink" : "/SensorThingsService/v1.0/Observations?$top=4&$skip=4",
  "value" : [
    { … },
    { … },
    { … },
    { … }
  ]
}
```

Query URL patterns: $select

Get only *description* and *id* for all Things

- GET … /v1.1/Things?
  $select=@iot.id,description
  - Response:

```
{
  "value" : [
    {
      "description" : "camping lantern",
      "@iot.id" : 1
    },
    {
      "description" : "camping stove",
      "@iot.id" : 2
    }
  ]
}
```

Query URL patterns: Sorting

GET all Observations sorted by phenomenonTime, newest first

- GET … /v1.1/Observations?

  $orderby=phenomenonTime desc

- Functions work for Ordering

  GET … /v1.1/Datastreams?

  $orderby=length(name) desc

Query URL patterns: Filtering

GET only Observations with result (value) > 5

- GET … /v1.1/Observations?
  $filter=result gt 5
  - Response:

```
{
  "@iot.nextLink" : "/v1.0/Observations?$filter=result gt 5&$top=4&$skip=4",
  "value" : [
    {
      "phenomenonTime" : "2016-06-22T13:21:31.144Z",
      "resultTime" : null,
      "result" : 10,
      "@iot.id" : 34,
      "@iot.selfLink" : "/SensorThingsService/v1.0/Observations(34)"
    },
    { … },
    { … },
    { … }
  ]
}
```

Query URL patterns: Functions 1

Lots of Choice

- Comparison:
  - gt: >   ge: >=
  - Eq: =   le: <=
  - lt: <   ne: !=
- Logical:
  - and / or / not
- Mathematical:
  - add / sub / mul / div / mod
  - round(n1)
  - floor(n1) / ceiling(n1)

- String Functions:
  - substringof(p0, p1)
  - endswith(p0, p1)
  - startswith(p0, p1)
  - substring(p0, p1)
  - indexof(p0, p1)
  - length(p0)
  - tolower(p0)
  - toupper(p0)
  - trim(p0)
  - concat(p0, p1)

Query URL patterns: Functions 2

Even more choice!

- Geospatial:
  - geo.intersects(g1, g2)
  - geo.length(l1)
  - geo.distance(g1, g2)
  - st_equals(g1, g2)
  - st_disjoint(g1, g2)
  - st_touches(g1, g2)
  - st_within(g1, g2)
  - st_overlaps(g1, g2)
  - st_crosses(g1, g2)
  - st_intersects(g1, g2)
  - st_contains(g1, g2)
  - st_relate(g1, g2)
- Date and Time:
  - now()
  - mindatetime()
  - maxdatetime()
  - date(t1)
  - time(t1)
  - year(t1)
  - month(t1)
  - day(t1)
  - hour(t1)
  - minute(t1)
  - second(t1)
  - fractionalseconds(t1)
  - totaloffsetminutes(t1)

Query URL patterns: Encoding in URLs

Tricky Bits!

- Strings: in single quotes
  - $filter=name eq 'Living room'
  - Single quotes are doubled: 'Hylke''s Living room'
- ISO 8601 DateTimes: not quoted
  - $filter=phenomenonTime gt 2018-03-09T08:14:54+00:00
  - Don’t forget to URLEncode:
    2018-03-09T08:14:54%2B00:00
- ISO 8601 Durations
  - duration'<…>'
  - duration'P1WT1H'

Query URL patterns: Filtering examples

Find me a Thing

- All observations with an even result
  - Observations?$filter=result mod 2 eq 0
- Observations of the last hour
  - Observations?$filter=phenomenonTime gt now() sub duration’PT1H’
  - https://en.wikipedia.org/wiki/ISO_8601#Durations
- Datastreams that measure temperature (ObservedProperty id 1)
  - Datastreams?$filter=ObservedProperty/@iot.id eq 1
- Filtering on JSON properties
  - Things?$filter=properties/style eq 'Cozy'
  - Observations?$filter=result gt Datastream/properties/max

Query URL patterns: $expand

GET the Thing with id=17 and its Datastreams

- GET …/v1.1/Things(17)?
  $expand=Datastreams
  - Response:

```
{
  "name" : "My camping lantern",
  "description" : "camping lantern",
  "Datastreams" : [
    { … },
    { … },
    { … }
  ],
  "@iot.id" : 17
}
```

Query URL patterns: $expand(…)

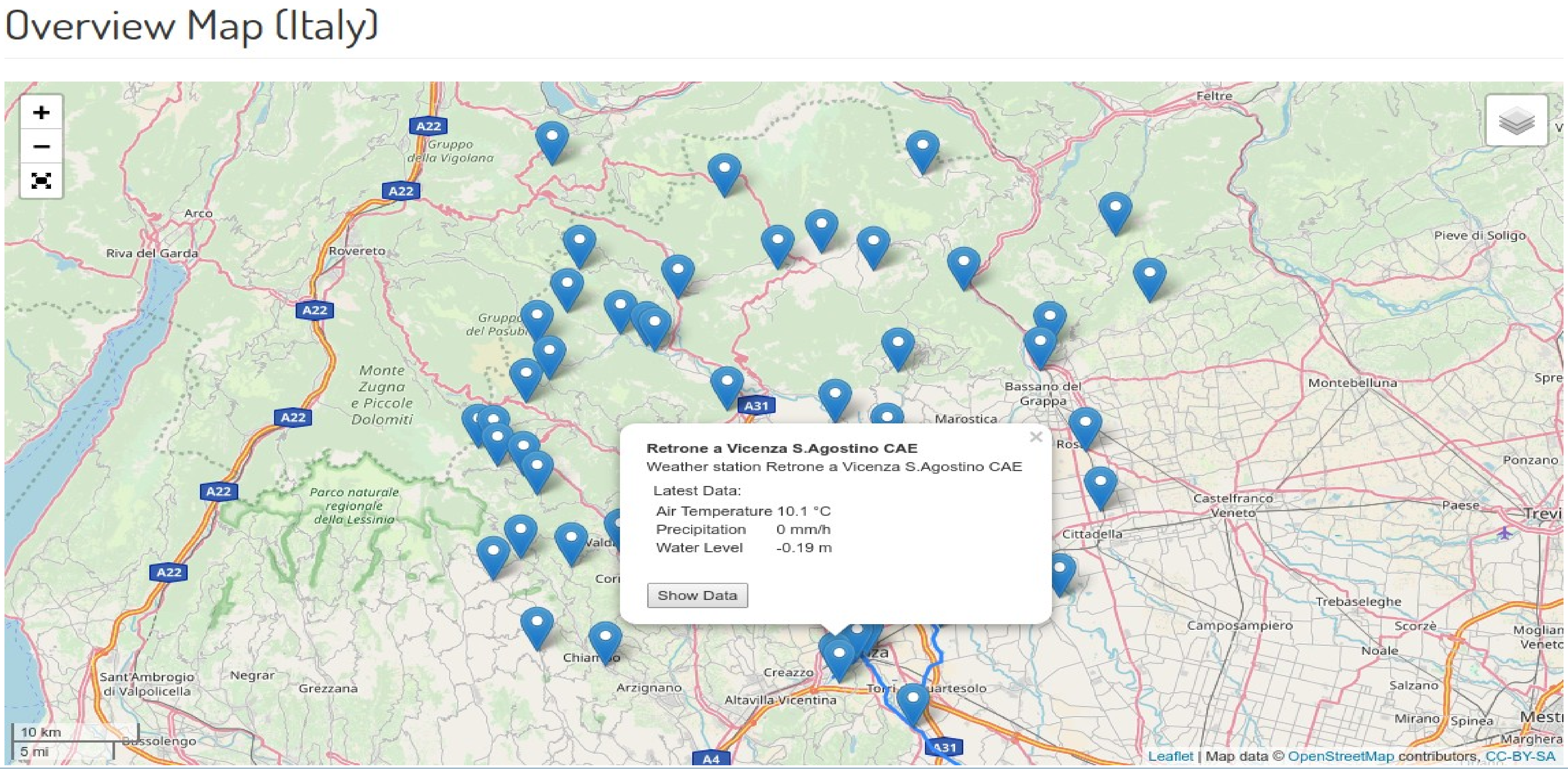
GET only description, id and Datastreams for Thing 17 and for the Datastreams only id and description

- GET …/v1.1/Things(17)?
  $select=@iot.id,description&
  $expand=Datastreams($select=@iot.id,description)
  - Response:

```
{
  "description" : "camping lantern",
  "@iot.id" : 17,
  "Datastreams" : [
    {
      "description" : "Temperature measurement",
      "@iot.id" : 19
    },
    {
      "description" : "Humidity measurement",
      "@iot.id" : 21
    }
  ]
}
```

Query URL patterns: $expand example

Get everything for my map in 1 request



Retrone a Vicenza S.Agostino CAE
Weather station Retrone a Vicenza S.Agostino CAE

Latest Data:

Air Temperature	10.1 °C
Precipitation	0 mm/h
Water Level	-0.19 m

Show Data

Query URL patterns: $expand example

Get everything for my map in 1 request

```
…/v1.1/Things?
   $select=id,name,description,properties
   &$top=1000
   &$filter=properties/type eq 'station'
   &$expand=
      Locations,
      Datastreams(
         $select=id,name,unitOfMeasurement
         ;$expand=
            ObservedProperty($select=name),
            Observations(
               $select=result,phenomenonTime
               ;$orderby=phenomenonTime desc
               ;$top=1)
      )
```

Creating new Entities

Create a new Thing

- POST ... /v1.1/Things

```
Content-Type: application/json;charset=UTF-8

{
  "name" : "Office",
  "description" : "My Work Room",
  "properties" : {
    "style" : "Business",
    "balcony" : false
  },
  "Locations" : [
    {
      "@iot.id" : 1
    }
  ]
}
```

  - Response:

```
Location: http://localhost:8080/FROST-Server/v1.1/Things(2)
```

Creating new Entities

POST a new Thing with a new Location

- POST … /v1.1/Things

```
{
  "name" : "Office",
  "description" : "My Work Room",
  "properties" : {
    "style" : "Business",
    "balcony" : false
  },
  "Locations" : [
    {
      "name" : "My Office",
      "description" : "The office room of Fraunhoferstr. 1",
      "encodingType" : "application/vnd.geo+json",
      "location" : {
        "type":"Point",
        "coordinates":[8.425548,49.015196]
      }
    }
  ]
}
```

  - Response:

```
Location: http://localhost:8080/FROST-Server/v1.1/Things(2)
```

Creating new Observations

Create a new Observation

- POST … /v1.1/Observations

```
{
  "result" : 123,
  "Datastream" : {
    "@iot.id" : 1
  }
}
```

- POST … /v1.1/Datastreams(1)/Observations

```
{
  "result" : 123
}
```

- phenomenonTime and FeatureOfInterest are generated automatically if not provided.

Creating new Observations – HTTP vs MQTT

Options, options …

	+	–
MQTT	• Efficient for subsequent Observations	• Connection management • Can only create Observations • Persistent connection makes load balancing tricky
HTTP	• Simple requests • Can create all entities • Works through firewalls & proxies • Simple load balancing	• New connection per request

Changing Entities
Update an existing Thing

- PATCH … /v1.1/Things(1)

  ```
  {
    "description" : "A new description"
  }
  ```

  Changes only the specified fields

- PUT … /v1.1/Things(1)

  ```
  {
    "name" : "A new name",
    "description" : "A new description"
  }
  ```

  Replaces all fields.
  Fields that are not set are removed (properties in this case)!

Deleting Entities
Delete a Thing

- DELETE … /v1.1/Things(1)
- Deletes the Thing and all objects depending on the thing
  - Datastreams
    - Observations

Managing your data
Summary

- Base URL: http://server.de/FROST-Server/v1.1
- Read: GET
  - v1.1 → Get index
  - v1.1/Collection → Get all in a set
  - v1.1/Collection(id) → Get one from a set
- Create: POST
  - v1.1/Collection → Create a new entity
- Update: PATCH
  - v1.1/Collection(id) → Update an entity
- Update: PUT
  - v1.1/Collection(id) → Replace an entity
- Delete: DELETE
  - v1.1/Collection(id) → Remove an entity

Extensions

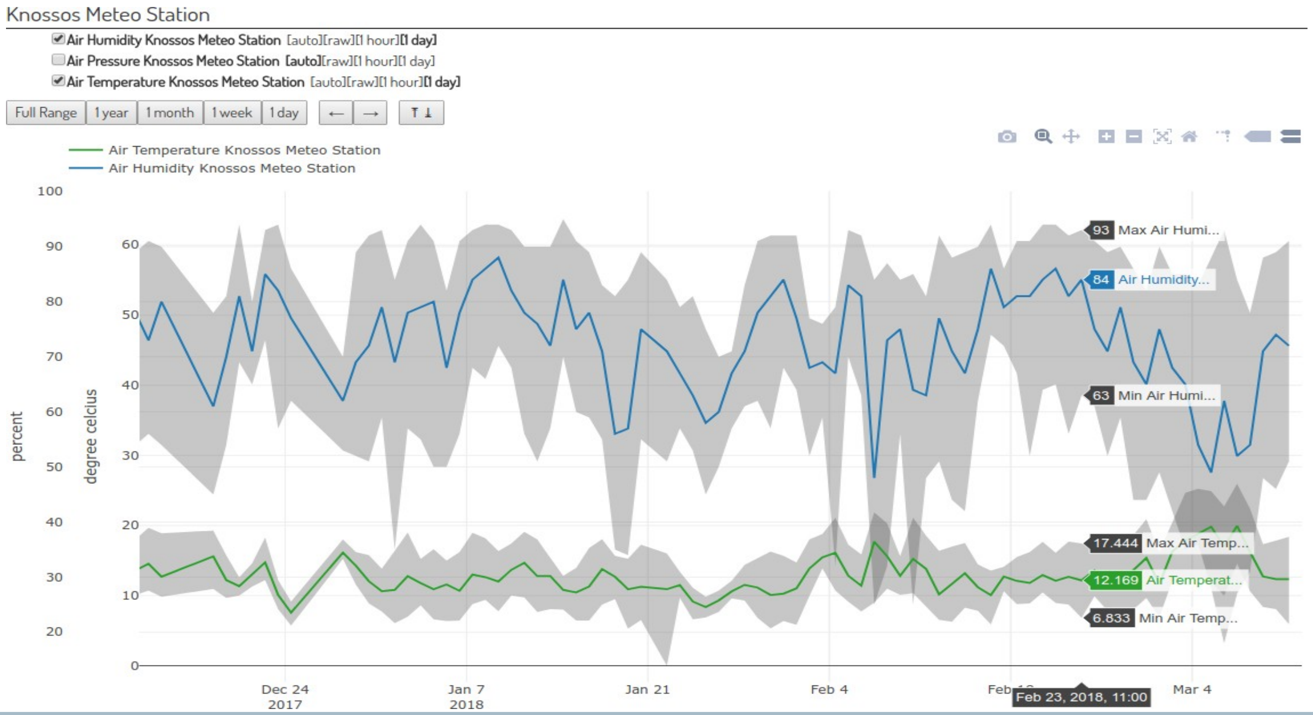
Wait, there is More!

- MQTT
  - Receive push notification on Entity create or update
    - Subscriptions as in urls
      - v1.1/Datastreams
      - v1.1/Datastreams(1)
      - v1.1/Datastreams(1)/name
      - v1.1/Datastreams(1)/Observations
    - ?$select to reduce message size
    - For all entity types
  - Create Observations using MQTT messages

Extensions

Wait, there is More!

- MultiDatastream
  - Datastream with multiple ObservedProperties
  - Observations with multiple result values
    - Observation/result is a JSON Array
  - Useful for
    - Wind:
      Speed / direction
    - Aggregates
      Average / Minimum /
      Maximum / std-deviation



Extensions

Wait, there is even More!

- Data Array
  - More efficient Observation encoding
  - GET … /v1.1/Observations?$resultFormat=DataArray
  - For Get & POST
- Batch requests
  - Multiple actions in 1 request

FROST Extensions

More, more, more!

- Filtered Delete
  - If you can GET it, you can DELETE it
  - DELETE v1.1/Observations?$filter=phenomenonTime lt now() sub period'P1D'
- Time interval functions
  - before / after / starts / finishes / overlaps / contains
- CSV ResultFormat (Thank you BRGM!)
  - For easier export to spreadsheets
  - GET … /v1.1/Observations?$resultFormat=CSV
- GeoJSON ResultFormat
  - For easier display in mapping software
  - GET … /v1.1/Things?$expand=Locations&$resultFormat=GeoJSON

Contact

Dr. Hylke van der Schaaf
Information Management and Production Control
hylke.vanderschaaf@iosb.fraunhofer.de

Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB
Fraunhoferstraße 1
76131 Karlsruhe, GERMANY
www.iosb.fraunhofer.de